International Research Journal of Multidisciplinary Scope (IRJMS), 2024; 5(2): 723-731

Original Article | ISSN (O): 2582-631X DOI: 10.47857/irjms.2024.v05i02.0629

Cloud Intrusion Detection Using Hybrid Convolutional Neural Networks

Mahalakshmi SB[1], Lokanayaki Karnan[2*], Nayana KV[3], Thirunavukkarasu V[2]

[1]Department of Artificial Intelligence & Machine Learning, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India. Department of Computer Science, Christ University, Bangalore-, Karnataka, India. [3]Department of Computer Science, St Francis de Sales College, Bangalore, Karnataka, India. *Corresponding Author's Email: loka.karnan@rediffmail.com

Abstract

Instead of storing data on a hard drive, cloud computing is seen as the best option. The Internet is used to deliver three different kinds of computing services to users all over the world. One advantage that cloud computing provides to its customers is greater access to resources and higher performance while at the same time increasing the risk of an attack. Intrusion detection systems that can handle a large volume of data packets, analyse them, and generate reports based on knowledge and behaviour analysis were developed as part of this research. As an added layer of protection, the Convolution Neural Network Algorithm is used to encrypt data during end-to-end transmission and to store it in the cloud. Intrusion detection increases the safety of data in the cloud. In this paper demonstrates the data is encrypted and decrypted using a model of an algorithm and explains how it is protected from attackers. It's important to take into account the amount of time and memory required to encrypt and decrypt large text files when evaluating the proposed system's performance. The security of the cloud has also been examined and compared to other existing encoding methods.

Keywords: CNN, Cloud Security, DDoS, Intrusion Detection, k-NN Cloud-Based Intrusion Detection, Spam.

Introduction

It is becoming more and more commonplace to use cloud computing today (1). The cloud is being adopted by all businesses. With the cloud, users can store and access their data from anywhere for no additional cost. Distributed systems and cloud computing are closely related in that data is dispersed across multiple locations and can be accessed from any location on the planet. As a result, cloud computing allows users to access all of their data from any location.

Client-server computing was the norm before the advent of cloud computing. If you're using a client-server model, you can only access data from clients that are although the server has access to its resources, it is still connected to it. A server-to-PC key makes it simple for a customer's PC to connect to the server and access the data therein. In the early days of cloud computing, distributed systems and client-server architectures served as the foundation. There are three types of cloud computing available to users: private, public, and hybrid.

A public model is one that is available to anyone, anywhere in the world, for no charge. There is always a third party in charge of running and maintaining the cloud server infrastructure and infrastructure services. Because it is specifically tailored for a single organisation and can only be accessed from within that organisation, a private cloud is far more secure and reliable. Private cloud computing has become a popular option for many businesses. The hybrid cloud combines the best features of both cloud computing models. Organizations benefit from the protection and flexibility provided by hybrid cloud intrusion detection, which connects at least one private cloud with one or more public cloud services. To keep the cloud safe from malware, cloud computing uses two types of security measures (2).

(Received 16th February 2024; Accepted 21st April 2024; Published 30th April 2024)

There are two types of security: data security and network security. Numerous threats in the cloud data centre attempt to harm the cloud or steal data from it. A number of studies have shown that the likelihood of an attack is greatly reduced if an intrusion detection system and other security measures are implemented on every cloud device. In order to steal data from cloud servers, hackers devise a variety of attack algorithms, encryption methods, and decryption methods. The cloud server's data can be destroyed or corrupted by this intrusive method. In order to protect the cloud server, you need a secure architecture.

Data storage in a cloud server necessitates additional security measures after the data has been placed there. Only a set of implementations, methods, and techniques can ensure the security of a cloud. Different researchers have devised a variety of cloud-security methods, procedures, and algorithms. Intrusion Detection System is the best of the bunch. Detection of intrusions and policy violations from a network is the goal of an Intrusion Detection System (IDS) (3).

Anti-malware can be detected in the cloud by an intrusion detection system. When an intruder tries to attack the cloud data centre, it sends an alert to the administrator. Intrusion Detection System (IDS) benefits primarily from monitoring network traffic and determining whether or not it is legitimate. When malware is detected, some Intrusion Detection Systems are so advanced that they can take action immediately. There are a wide variety of Intrusion Detection Systems (IDS) in antivirus software that can detect intrusions from cloud servers. Intrusion Detection Systems that are most commonly used include (4).

Interference from the network can be detected using NIDS. Monitors all the devices on the network, as well. HIDS, like NIDS, is critical to the network's operation. HIDS is a tool for spotting and keeping track of suspicious activity on host PCs and devices. Host PCs run HIDS, and network devices run NIDS, but they are not interchangeable. The Intrusion Detection System is divided into two parts: Anomaly-Based IDS and Signature-Based IDS. Using a signature in signature-based IDS is common. Its job is to keep track of all network activity, whether it's genuine or not. On the behaviour of the system, Anomaly-Based is based. Anti-malicious detection systems are alerted when an attacker tries to alter system behaviour in an abnormal way.

In order to skew the results by country, we'll use a unique port number for each router, which we'll design and implement in this article. In order to detect an attack on the router, we'll link IDS with each country's IDS. Finally, we'll apply some rules and come to the conclusion that the likelihood of detecting a cloud-based attack is high if an effective algorithm is designed.

Related Work

This paper, written by Narendra et al., examines cloud security issues and offers recommendations for mitigating them on an elementary level (5). There are several attacks, threats, and models discussed on the cloud server in this paper by the authors. The authors claim that while the cloud can store and manage any type of data, it comes with a number of security risks. Cloud computing is plagued by security breaches and cyberattacks. Each new technology has two stages. The first stage is marked by difficulties, while the second stage is marked by success. As in the real world, there are phases to cloud computing, with one providing advantages and the other posing difficulties for the service provider. An insider attack or malware threat is just one example among a number of challenges. An important security issue is addressed in the article's conclusion, as well as a few dangers of destroying clouds.

Network administrators are alerted if suspicious activity is detected in the network by the Intrusion Detection System, according to Jyoti Snehi (6). If a network were to engage in malicious activity, both data breaches and user loyalty would be jeopardized. A cloud server must be protected from both internal and external threats such as password cracking when data is uploaded. To keep a network safe, you need more than just an anti-malware solution and a firewall. Interference across the network can be detected using IDS in this article, which discusses the datasets used in other articles and concludes.

An efficient tool called snort was used to implement the architecture that Arya chandra designed using a network (7). There were two cloud servers in this system, and there were two ports that could interfere with each other. VMBR1 and VMBR2 are the names of the virtual machine ports, while cloud

server-1 and cloud server-2 are the names of the virtual machines. Intrusion Detection System (IDS) is used three times in this article by the author. As a result of this, IDS was first installed outside of the cloud server. Inside of the cloud server, an Intrusion Detection System (IDS) was installed. IDS (Intrusion Detection System) was installed on both sides of the cloud servers for the third time. Three truth tables were set up following the attack on the network to indicate the likelihood of another attack. Even though RAM affects 0.25 percent of execution, Snort found that CPU has a much greater impact on performance. CPU, on the other hand, has no effect on the execution.

A host-based framework for intrusion detection was proposed by Gassais et al. (8). Machine learning methods are used in this article to detect smart device intrusions. Another is a tracking method that uses machine learning algorithms to handle devices and data automatically and generate alerts. Deep learning is one of many machine learning used in this article that can detect interruptions. The host-oriented approach was used to demonstrate how this solution can be tailored to a variety of devices and how it functions effectively.

As the amount of data stored on a cloud server grows, so does the number of attacks on that server (9). Right now, the most pressing issue is one of cyber security. The reputation of security services can be tarnished if they fail to stop the intrusion. Various Intrusion Detection methods were examined by Jang Jacquard et al. The Signature-Based IDS and the Anomaly-Based IDS are two of these technologies. The cloud also made use of data resource techniques such as these. Conclusions are drawn from recent papers and new models are sought to improve AIDS performance as a solution to Cloud Intrusion Detection System (CIDS) issues in the final section of the paper.

To detect anomalies, Mehrnaz Mazini employs the artificial bee colony technique and develops an AdaBoost algorithm to detect large exposure charges with cheap specific charges. Artificial Bay-Colony assisted in the selection of several features. Classification and testing of features are carried out through the Ada Boost algorithm. The Ada Boost algorithm utilised the meta-algorithm. Accuracy of an advanced approach for organising various attack groups is a factor in this issue. At the end of the paper, an Artificial Bee-colony meta-algorithm was used to optimise the problem of IDS detection.

When Vishal and Vasudha looked at various papers on DOS attacks, they found that there are numerous possible causes. DDoS attacks can target a variety of targets, including cloud servers, websites, OSI model layers, and more. A DDoS attack, on the other hand, employs a swarm of malicious machines or computers to flood a single server with malicious traffic, rendering the website inaccessible to users or providing them with only a partial service. It's difficult to track down the IP addresses used in a DDOS attack because so many requests are sent to the server at the same time. It is difficult to locate the IP addresses used in a DDOS attack because the number of requests sent to the server is unlimited. An OSI model was developed and all possible OSI model attack methods were examined after it had been peer-reviewed. Final results show that Random Forest and Cat Boost algorithms have an accuracy of 99.99 percent or higher.

The cloud has been made safer by a number of researchers. Algorithms developed by different researchers are different. In order to protect the architects, many architectural structures were built. The findings of some researchers were based on a review of previous studies and the development of new detection methods. While the authors of paper (10) researched which OSI model layers might be vulnerable to DDoS attacks and also compiling examples of various DDoS attacks from a variety of sources, makes no mention of ways to guard against or detect DDoS attacks in the cloud. DDoS and brute force attacks, as well as pattern matching and brute force attacks, will target the cloud server. Various types of brute force prevention methods will be collected to safeguard the cloud against pattern matching attacks of DDoS attacks from various papers, this article does not show any approach to protect the cloud from DDoS or discuss DDoS attack detection techniques. The cloud server will be the target of DDoS and brute force attacks, as well as pattern matching and brute force attacks. While some fuzzy based prevention methods will be discussed in order to guard against pattern matching attacks on the cloud, HIDS will be used. The cloud flare service will be implemented as a defence against DDoS attacks. Some mathematical rules will

then be applied to the data and conclusions drawn as a result.

Methodology

K-Nearest Neighbour

Cloud computing necessitates the maintenance of a large number of databases. Processing such massive amounts of data necessitates a high level of temporal and spatial complexity. Datasets should be reduced in order to save time and space while maintaining accuracy. Classification accuracy may be hampered if the training dataset contains some noisy examples. This means that the main goal is to reduce the number of training examples while maintaining or even improving accuracy, and to minimise as much noise as possible in the training set. A method for reducing the size of the training set by removing boundary instances and reducing the number of noisy examples is presented in this paper. We first used our Penalty-Reward based method to remove noise from the original dataset in our proposed model. After that, relevant records from the dataset are collected for training using the k-Nearest Neighbor (kNN) method.

Prior to applying the proposed method, the datasets are first processed and normalised. The NSL-KDD dataset contains some features with non-numeric values, so pre-processing is required to convert those non-numeric values to numeric values before we begin our analysis. Secondly, the NSL-KDD dataset contains features with discrete and continuous values, causing the feature values to have a different range. This dataset is normalised using min-max normalisation to address this issue. The datasets can be used for experiments after normalisation.

Two additional datasets, Safe and Temp, are used for experimentation in the proposed method. Provisional datasets should contain all occurrences that appear to be noisy, while the Safe datasets should contain all other instances. As a result of using penalties and rewards, we are able to keep track of all instances and identify which ones are the most disruptive. Figure 2 depicts the entire process of discovering the KDD Train+ dataset's Reduced training set.

This is done by first calculating a set of K-Nearest Neighbours for all of the instances in the dataset. We used the Euclidean distance function to figure out how far apart two points are. According to this definition, it is:

For each input attribute, the input values are defined as the two vectors defined as inputs, with the total number of input attributes (m) being equal to the number of input vectors (x, y). In our dataset, we use this function because all of the input attributes are numeric and they fall within an approximately equal range in width.

We need to make a decision about which dataset to put each individual instance in. It's as simple as that: For each instance *Xa*,

1. It is necessary to find the closest k- Nearest Neighbour of an object in the set of all objects in which *Xa* among {*X*1,*X*2,...,*XN*}. Index number *a* is the number of instances in the dataset, while N is the total number of instances.
2. For each k-Nearest Neighbour, count how many instances of Ca there are and store that number in Ta. The class Ca is associated with Xa, while the temporary variable Ta is used in conjunction with Xa.
3. Place Xa in the Safe dataset if Ta , otherwise put Xa in Temporary dataset. To put it another way, is a threshold number.
4. Increase t by 1. If t is equal to S terminate the algorithm, otherwise go to Step 3.

Additionally, the dataset's effectiveness must be determined for each individual instance. Rewards are given for instances that fall into the same class as that reward and penalties are applied for instances that fall into a different class than that reward. After that, the following equation should be used to determine the effectiveness of each instance.

Examples that are ineffective are expected to be noisy. This means that we can select all of the noisy instances using our Penalty-Reward-based method. It is then necessary to remove from both Safe and Temp datasets all selected noisy instances. The NSL-KDD-Train+ dataset and Temp dataset should be deleted after this step if the Temp dataset is empty. Then, the algorithm should be terminated. Only the Safe dataset should be retained for future use. NSL-KDD-Train+ dataset should be deleted from NSL-KDD-Train+ dataset if the Temp is not empty All of the situations in Temp. need to be dealt with in a proper manner. Some instances may be moved from Temp to Safe as a result of this procedure. There is no longer any risk 15of incorrect classification due to

the removal of previously noisy instances from the dataset.

Temp should delete both datasets after this procedure is finished. In addition, the clean, noise-free Safe dataset should be kept for future operations. In this case, no instances receive Penalty or Reward because the noisy instances have been removed from the dataset.

The dataset is cleaned up using the aforementioned algorithm, which eliminates all noisy and boundary instances. As a next step, the Safe dataset is partitioned into as many partitions as there are classes in the dataset (C), with each partition named "Safe1", "Safe2",..., SafeC". Each dataset's instances are grouped together into a single class. This is followed by a process called k-Nearest Neighbour (kNN). There are a number of ways to determine a single instance's k-Nearest Neighbour (kNN) This algorithm selects a subset of the class's members that can be used to model other members of the class. As a result, a lot of instances will be eliminated from each dataset. Reduced sets from each dataset should be merged together to create a final dataset that is more manageable.

After reducing the dataset, any existing classification method can be used to test the accuracy of the classification. As a classifier, we employed a Convolutional Neural Network (CNN).

Neural Network

Neural networks that use convolutional calculations are known as convolutional neural networks. Traditional fully-connected neural networks have a large number of weights, but Convolutional Neural Networks have a smaller number of weights and are more efficient. A variety of fields now make use of Convolutional Neural Networks, image classification, object recognition, just a few of the features include: pose estimation, text detection, action recognition, scene labelling and speech and natural language processing available (11, 12). Images can be classified using Convolutional Neural Networks, which are particularly effective (13). There are four layers in a typical Convolutional Neural Network: an input, a convolutional layer, and a pooling layer (14).

$$hl_i^{fm}=\tanh(w^{fm}xi:i+f-1+b) \quad [1]$$

where b ∈ R a bias term is denoted by b. The filter hl is applied to each TCP/IP { x1:f,x2:f+1,...,xn−f+1} connection record in order to generate feature maps for each set of features in a TCP/IP connection record.

$$hl=[hl1,hl2,...,hln-f+1] \quad [2]$$

where hl ∈ R^{n-f+1} and We then perform hl = maxhl on each feature map, which is a variation on the max-pooling operation. For example, the most important features are obtained by selecting one of the most valuable features. Although multiple features are fed to the fully connected layer, these new features have more than one feature. The probability distribution over each class is provided by the softmax function in a fully connected layer. The mathematical definition of a fully connected layer.

$$ot=softmax(whohl+bo) \quad [3]$$

Input layer: Convolutional Neural Nets are trained using input data from the input layer. First, we use 8 8 grayscale images as input instead of the raw training data. Image recognition is used as input data and grayscale images are used as input. FOA's weights were used to resample training data in the training phase, which we used.

Convolutional layer: Because they have convolutional layers, Convolutional Neural Networks differ significantly from the BP Neural Networks. We can extract higher-dimensional features and reduce calculation time by using the convolution kernel in the convolutional layer (15). We use three convolutional layers in our Convolutional Neural Networks in this paper. ReLU Function is used as the activation function for each convolutional layer's convolution calculations.

Pooling layer: In order to reduce the number of parameters and prevent over-fitting, a pooling layer is placed between two continuous convolutional layers. The max-pooling method is the most widely used for pooling resources. Here we use the max-pooling method as well. The three convolutional layers are separated by two pooling layers.

Fully connected layer: In a fully connected layer, every neuron communicates with every other neuron in the preceding layer. The fully connected layer of a Convolutional Neural Network is typically located at the end of the network. The Convolutional Neural Network presented in this paper has two layers, both of which are fully interconnected with each other. To avoid overfitting, a random dropout is inserted into the first fully connected layer. The second fully connected layer SoftMax Function is the activation function for classification (16).

Our proposed algorithm is given below:

1. Set up the dataset from scratch.
2. Then, for each instance Xa in the dataset, find the K-nearest neighbour (Ka).
3. Determine the number of neighbours in the same category as Xa for each instance of Xa.
4. Insert Xa into S if Ta , otherwise into T.
5. For each instance Xa, calculate the effectiveness.
6. All instances that have a negative effect on S and T should be deleted.
7. If T is empty, then proceed to step 9; otherwise, proceed to step 7.
8. Calculate kNN for all the instances in T after deleting all of the selected instances from NSL-KDD-Train
9. Calculate Ta as the number of neighbours in T that belong to the same class as Xa for each instance of Xa in T.
10. If T$a \geq \theta$, insert Xa into S.
11. Compute function (1) and (2) apply to function (3)
12. End.

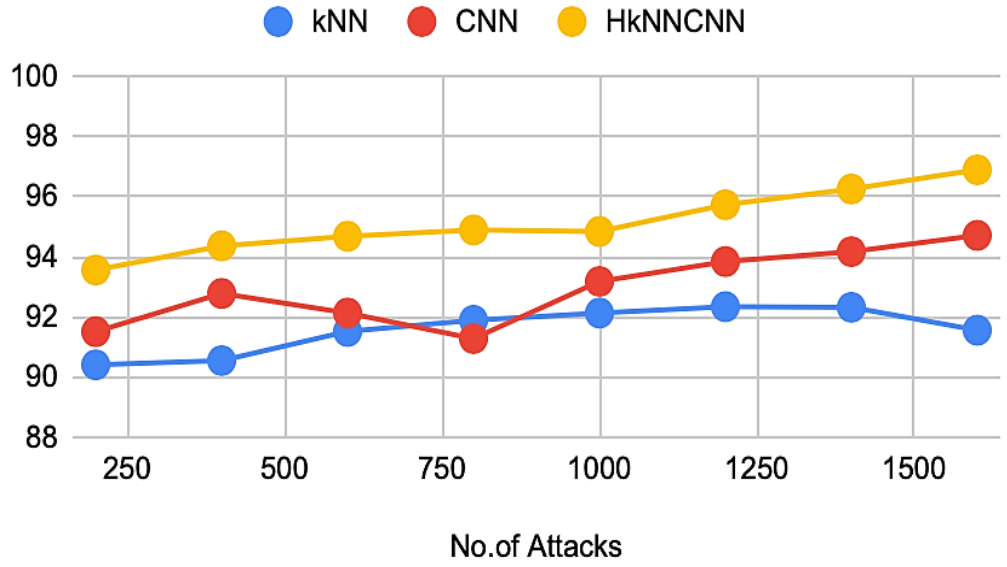

Figure1: Proposed method accuracy comparison

Dataset Description

For the purposes of our experiments, we consulted the NSL-KDD dataset. The original KDD dataset has been reworked into this new dataset. In the NSL-KDD dataset, there is a four-component structure. KDD Train+, KDD Test+, 20% KDD Training+, and KDDTest-21 are the components. We used one dataset for training (KDD Train+) and one dataset for testing (KDD Test+), each containing 125,973 instances. The dataset is divided into two categories: normal and attack, each with its own set of 41 features. There are four categories of features that can be broken down into:

- **Basic features:** TCP/IP has been used as a basis for many of the features.
- **Content features:** Domain expertise is put to use by content features to get at the original TCP packets payload.
- **Time-based traffic features:** The features that take two seconds to mature are captured by traffic-related features that are time-based.
- **Host-based traffic features:** Traffic based on a host's IP address shared authorised access to all attacks lasting more than a few minutes.

Many records are contained in the KDD-Train+ dataset. Due to the time it will take to analyse that dataset, it cannot be used for training purposes. In our paper, we propose a smaller training dataset in order to reduce training time while still maintaining accuracy.

Result and Discussion

Evaluation

Evaluation of the classification models is done in this section. For each model, we calculated the Accuracy

and Sensitivity and Precision of each model using the results from the confusion matrix.

The following is a definition and formula for Accuracy, Sensitivity, and Precision:

Accuracy

As a measure of a model's accuracy, we can count the number of times it correctly identified both normal and attacking traffic.

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

Prediction accuracy is measured by dividing the number of True Positive predictions by the number of True Positive predictions.

Precision (SE)

Depending on how many True Positive predictions there are, a sensitivity ratio (also called Recall or the True Positive rate) can be calculated.

$$SE = \frac{TP}{TP + FN}$$

Specificity (SP) is defined as, a measure of capacity to test the negative samples.

$$SP = \frac{TN}{TN + FP}$$

Table 1: Recognition Time

No. of Attacks	Recognition time(ms)		
	kNN	CNN	HkNNCNN
200	446	356	289
400	797	548	318
600	1158	798	777
800	1783	1189	791
1000	1900	1426	998
Average	1216.8	863.4	634.6

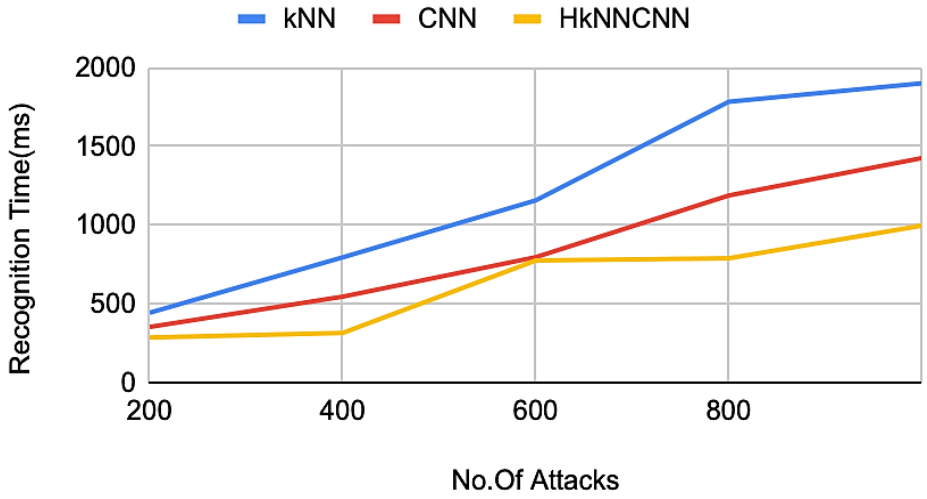

Figure 2: Execution-time comparison

Error rate is the total number of incorrectly classified instances among all web pages available during the test.

The proposed method to existing methods such as CNN and kNN in Fig. 1, you will see that it has a classification accuracy of 96.89%, which is 2.08 percentage points and 4.45 percentage points

higher. The proposed CNN-based classification reduces the classifier's errors.

In Figure 2 and Table 2, there are many existing methods, such as CNN and kNN, that can be compared in terms of execution time. There is less execution time for the proposed hybrid HkNNCNN approach, which is 0.44ms less than CNN 1.44ms less than the existing kNN methods, respectively.

From Figure 3 the Error Rate(ER) value is plotted for existing kNN, CNN and proposed HkNNCNN algorithm. The ER value gradually increases if the number of word counts decreases with the range of 200-1000 for all methods. The proposed HkNNCNN algorithm provides lesser results of 1.11% (1600 attacks) which significantly lesser 3.08% and 4.04% when compared with other existing methods are discussed in Table 2.

Table 2: Error Rate

No of attacks	Methods and Error rate (%)		
	kNN	CNN	HkNNCNN
200	7.59	5.49	4.44
400	7.45	6.22	3.64
600	6.48	5.87	3.32
800	6.11	5.12	3.11
1000	5.72	4.82	3.16
1200	5.55	4.19	2.27
1400	5.55	3.83	1.75
1600	5.15	3.19	1.11
Average (%)	6.2	4.84125	2.85

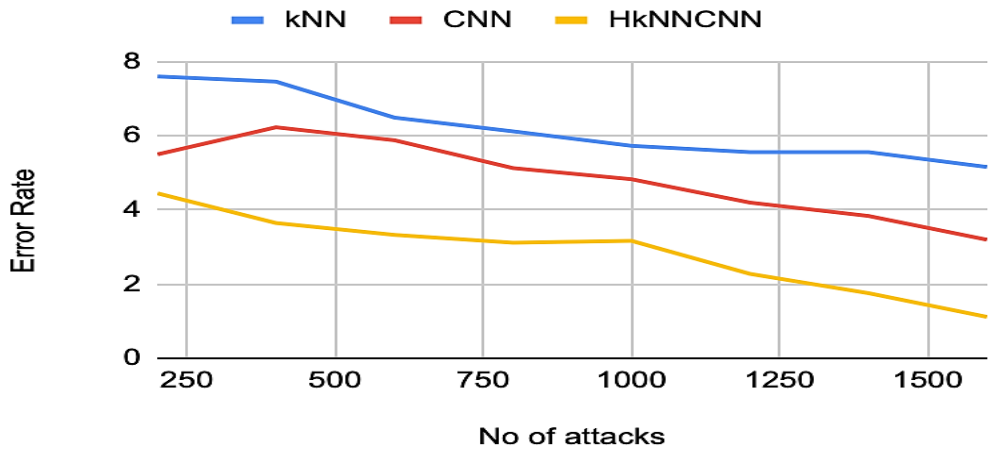

Figure 3: Error - rate comparison

Conclusion

In the end, we found that using an intrusion detection system to detect cloud-based intrusions was an excellent strategy. Secure cloud computing algorithms can be developed to prevent intrusion attacks on the cloud computing infrastructure. A secure architecture is required for the cloud to be secure. An attacker can gain immediate access to the cloud by installing spyware and then stealing keys. An alert and signature system can help protect the cloud. If an attacker repeatedly enters the wrong keys, he or she may receive a harsh warning. The

security of cloud servers can be improved by using various authentication methods, such as intrusion detection, two-step authentication, and digital signatures. The numerous routers and public cloud in this article's architecture are protected by HIDS, SIDS, and NIDS. Three more cyberattacks were launched against the cloud server in the days that followed. According to the first scenario, we were going after router R1. The second scenario involves an attack on the router R2, while the third involves an attack on the cloud server. There was a different table set up for every possible outcome after the assault had taken place. implemented various laws into tables to verify the accuracy of the table results. Better detection and protection methods could help prevent an attack on the cloud.

This tool will be compared with other intrusion detection systems (IDS) like Snort, Suricata, and OSSEC in the future to develop better algorithms and techniques to protect the cloud from intrusions.

Abbreviation

Nil

Acknowledgment

Nil

Author Contributions

All authors are equally contributed.

Conflict of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

Ethics Approval

Not applicable

Funding

Nil

References

1. Ibawi T, Mohammed A, Al-Zawi S. Understanding of a convolutional neural network. In: International Conference on Engineering and Technology (ICET'17), 2017; 1–6.
2. Krizhevsky A, Sutskever I, Hinton G E. Image classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, 2012; 1097–1105.
3. Li Z, Qin Z, Huang K, Yang X, Ye S. Intrusion detection using convolutional neural networks for representation learning. In: International Conference on Neural Information Processing, 2017; 858–866.
4. Shea KO, Nash R. An introduction to convolutional neural networks. In: Journal of Neural and Evolutionary Computing, 2015; 1511.08458.
5. Tadapaneni NR, Hussaini SSR. Cloud computing security challenges. In: SSRN Electron. J., 2020; 7:1–6.
6. Snehi J. Diverse methods for signature-based intrusion detection schemes adopted. In: Int. J. Recent Technol. Eng., 2020; 9:2, 44–49.
7. Aryachandra AA, Arif YF, Anggis SN. Intrusion detection system (IDS) server placement analysis in cloud computing. In: Proc. 4th Int. Conf. Inf. Communication Technology (ICoICT), 2016; 1–5.
8. Tadapaneni NR. Cloud computing: Opportunities and challenges. In: SSRN Electron. J., 2018; 6:9,122–143.
9. Gassais R, Ezzati Jivan N, Fernandez JM, Aloise D, Dagenais MR. Multi-level host-based intrusion detection system for Internet of Things. In: Journal of Cloud Computing, 2020; 9:1, 1–16.
10. Jang Jaccard J, Nepal S. A survey of emerging threats in cybersecurity. In: Journal Computer System, 2014; 80:5, 973–993.
11. Mazini M, Shirazi B, Mahdavi I. Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. In: Journal King Saud University Computer, 2019; 31:4, 541–553.
12. Vishal V, Vasudha K. DoS/DDoS attack detection using machine learning: A review. In: Proceeding International Conference Innovation Computer Communication (ICICC), 2021; 1–7.
13. Rahmad F, Suryanto Y, Ramli K. Performance Comparison of Anti-Spam Technology Using Confusion Matrix Classification. In: IOP Conf. Ser.: Mater. Sci. Eng., 2020; 879:012076.
14. Revathi S. Cloud Data security based on Fuzzy Intrusion Detection system with Elliptic Curve Cryptography (ECC) using Non-Adjacent Form (NAF) Algorithm. In: International Journal of Engineering Research & Technology (IJERT), 2021; 10:11.
15. Lou Y, He Y, Wang L, Chen G. Predicting network controllability robustness: A convolutional neural network approach. In: Systems and Control, 2019.
16. Lou Y, He Y, Wang L, Tsang R, Chen G. Knowledge-based prediction of network controllability robustness. In: Physics and Society, 2020.