SIGNATURE BASED DETECTION FOR CLOUD BASED LOG MANAGEMENT USING MACHINE LEARNING TECHNIQUE

[1]Ogili Solomon Nnaedozie

[1]Department of Electrical and Electronic Engineering, Faculty of Engineering,
Enugu State University of Science and Technology (ESUT)

Article Info

Article history:

Received: 11/01/ 2023
Revised: 27/01/2023
Accepted 02/02/2023

Corresponding Author Email:
pricesoloedozie@yahoo.com

Corresponding Author Tel:
+234 8169033442

ABSTRACT

This paper presents signature based detection mechanism for cloud based log management using machine learning technique. The study aimed at detecting unauthorized log entry files into the cloud server and isolate from the network. To achieve this, data was collected from INFN-Tier data center during the hadron collider experiments and then used to train a neural network algorithm after processing using service-specific procedures. The performance was evaluated using accuracy and loss parameters and the result reported a training accuracy of 0.94188 and loss of 0.385 respectively. Finally after cross validation, the accuracy recorded was 0.915. The neural network was further compared with other state of the art algorithms such as Naïve Bayes, K-mean and Isolation forest. The Neural Network algorithm emerged as the most accurate among the tested algorithms, with an accuracy of 0.9155, indicating that it correctly predicted outcomes with an approximate success rate of 91.55%.

Keywords: Cloud Log Entry, Machine Learning, Accuracy, Log Entry Files, Data Processing

1. INTRODUCTION

Cloud computing service is a modern computing paradigm that allows users to pay for the usage of services without the need to purchase physical hardware (Viera et al., 2010). This has led to rapid development of cloud computing along with the growing demand for Information Technology (IT) services. Cloud computing is efficient and cost-effective for consumers as they can use computing resources and services as needed from cloud computing providers (Hodo et al., 2016). One of the key factors that makes cloud computing attractive is its robust capacity to manage and store large amounts of resources. As a result, many enterprises in both the public and private sectors have adopted cloud computing for data and resource management (Bertero et al., 2017). The IT infrastructure of cloud computing consists of three major models: software as a service, infrastructure as a service, and platform as a service, which work collaboratively for effective data management. However, due to the sensitive and confidential nature of the information stored in the cloud, it has become a prime target for hackers and criminals (Viera et al., 2010; Amirreza, 2012; Anthony et al., 2010).

With the advancements in technology, cloud-based platforms have evolved into multi-mesh distributed and service-oriented paradigms with multiple domains, multi-tenancies, and multiple user autonomous administrative infrastructures. Unfortunately, these characteristics have also exposed the cloud-based platforms to numerous threats that question the integrity, confidentiality, and availability of its resources (Dhage et al., 2011; Axelsson, 1999).

Over the years, cloud-based infrastructures have experienced various types of attack models, such as wormhole, black hole, denial of service, man-in-the-middle, IP spoofing, among others, and have been exploited for ransomware attacks (Jun et al., 2011; Kento et al., 2009; Mkuzhalisai and Gayathri, 2012), where organizations are forced to pay a ransom to restore their services. This problem has been a major challenge in global cybersecurity studies and remains unresolved. Many solutions have been proposed, including cryptographic techniques (Olofin, 2021), signature-based approaches (Sonu, 2020; Mazzariello et al., 2010; Bakshi and Yogesh, 2010), and anomaly detection approaches (Bharadweja et al., 2011). However, these solutions often focus on securing the packet without considering the security of the cloud-based server infrastructure, leaving it vulnerable to intruders. To address this problem, there is a need for intrusion detection systems that can monitor the servers and prevent hackers from gaining unauthorized access.

Machine Learning (ML) has gained increased attention in the field of cyber security in recent years. ML, which is a branch of artificial intelligence, utilizes mathematical algorithms to learn from data and solve regression or pattern recognition problems (Olofin, 2021). ML has been used successfully in solving issues such as cloud intrusion detection (Hodo et al., 2016). However, despite the success of ML in addressing these challenges, cloud services still face issues such as unauthorized access, data breaches, unauthorized data modifications, and vulnerability to manipulation by malicious actors for various purposes such as covering their tracks or creating false evidence (Bertero et al., 2017). Ensuring the integrity of log entries, i.e., that they have not been altered, is crucial for accurate auditing, forensic investigations, and compliance requirements. Implementing tamper-proof log storage and verification mechanisms can be challenging due to the distributed nature of cloud log entries across multiple locations and different cloud service providers.

Addressing these issues and challenges of cloud log entry security requires a combination of technical, organizational, and policy measures. This will be achieved in this paper implementing a robust machine learning model training and validation techniques, ensuring data quality and consistency, optimizing for scalability and performance, and providing interpretability and explainability in machine learning models.

2. LITERATURE REVIEW

Belavagi and Muniyal (2016) presented a performance evaluation of supervised machine learning algorithms for intrusion detection. The study presents the application of machine learning classification algorithms like gaussian naïve bayes, logistic regression, support vector machine and random forest for the classification and prediction of intrusion on a wireless network system. It used the NSL-KDD datasets for testing the system and the experimental result shows that the random forest classifier has the best performance among the rest in threat identification and prediction.

Smys et al., (2020) presented a study on hybrid intrusion detection system for internet of things. The study uses a hybrid convolutional neural network model for the detection of intrusion on an IoT platform which is suitable for a wide range adoption of the technology. The study further used the conventional machine learning and deep learning model for the validation of the technique applied in this study. The result of the demonstration shows that the proposed hybrid model is very sensitive to attacks in IoT networks and performs better than the other machine learning models presented.

Alruhaily and Ibrahim (2021) researched on a multi-layer machine learning-based intrusion detection system for wireless sensor networks. The work adopted a defense-in-depth approach for applying a multilayered intrusion detection on a wireless sensor network with the integration of various machine learning algorithms like naïve bayes and random forest multiclass classifier. The result of the implementation shows that the system was effective for detection of normal, blackhole, grayhole, flooding and scheduling attacks. But despite the success, there is still room for improvements.

Anthi (2022) presented a study on detecting and defending against cyber attacks in a smart home internet of things ecosystem. The study presents the implementation of a novel secure hub for IoT devices which consists of security properties like confidentiality, access control and authentication. The aim of the study is to classify the IoT device encountered, identify malicious network packets and identify the type of attack that has occurred on the platform. Adversarial machine learning attacks were implemented for the experimentation of the system which is capable of exploring the weakness of the system. The result of the experiment identified that the system has a limitation in data labeling and feature extraction engineering.

Alaparthy and Morgera (2018) researched on a multi-level intrusion detection system for wireless sensor networks based on immune theory. This work considered the application of a type of immune theory called Danger theory for securing and mitigation of attack on a wireless sensor network. Wireless sensor network parameters like energy, frequency of data transfer, volume of data and developing an output based on concentrations and weight were put on constant monitoring for the security measures on the IDS. However, the work did not consider implementation on other adhoc networks.

3. METHODOLOGY

The methodology include data collection of cloud log entry files from the INFN-Tier data center and then processed using data replacement approach which ensured that all values of missing attributes were replace. The data was feed into a feed forward neural network algorithm for training using back-propagation algorithm and then generate the signature based threat detection model. This model was implemented with python programming language and evaluated considering accuracy. The performance of the model was validated considering existing state of the art cloud log security algorithm and the percentage improvement was recorded.

4. Data collection

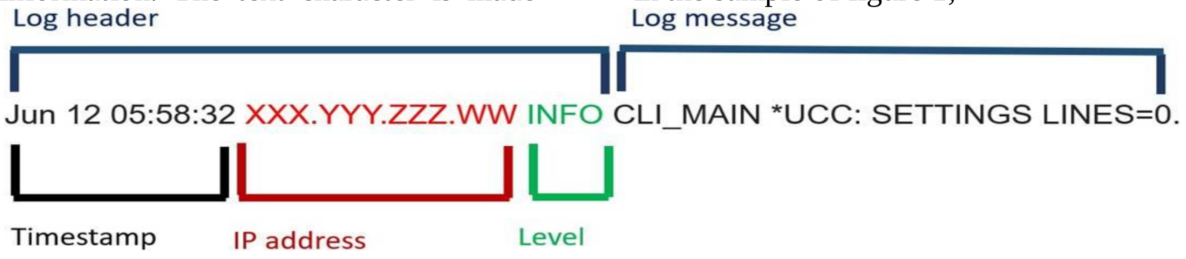
This work considers a similar log dataset used by Viola et al. (2022), which was collected from the INFN-Tier data center during the hadron collider experiments Dell et al. (2019]. The data is composed of four classes: storage, which includes tape data transfer services, storage, and disk devices; farming handles, which are responsible for data computations; network, which takes care of security regulations and access control; and user support, which manages authentication, account creation, and configuration. Each log entry consists of software utility crond, main transfer agent postfix, and standard message login syslog (Wang et al., 2011). The sample size of the collected data is 3,562,758 log entries, consisting of log suffixes and file types such as .gz and .txt, respectively. Table 1 presents the first 30 columns of the collected data.

Table 1: first 30 column of the data collection (Viola et al. (2022)

Filename	Frequency	Filename	Frequency

sudo.log	378,782	systemd.log	107,701
puppet-agent.log	368,531	mmfs.log	72,621
run-parts.log	365,735	rsyslogd.log	70,211
crontab.log	348,894	kernel.log	65,937
crond.log	347,706	logrotate.log	62,532
sshd.log	304,914	syslog.log	47,331
anacron.log	287,427	yum.log	43,300
postfix.log	175,565	fusinv-agent.log	42,124
auditd.log	120,473	root.log	37,343
smartd.log	109,441	gpfs.log	31,001
cvmfs_x509_helper.log	5396	userhelper.log	21,381
srp_daemon.log	4939	nslcd.log	20,543
edg-mkgridmap.log	4082	neutron_linuxbridge.log	8571
libvirtd.log	3329	runuser.log	6858
dbus.log	3302	cvmfs_x509_validator.log	6032

The table 1 presents samples of the data collection with different amount files containing alpha-numeric characters modeling the system operation information. The text character is made of dynamic strings like the run time and date. While the static text of the log entries includes message has log header, host and service name process as shown in the sample of figure 1;



```
Jun 12 05:58:32 XXX.YYY.ZZZ.WW INFO CLI_MAIN *UCC: SETTINGS LINES=0.
```

Figure 1: Sample of the log entry data (Viola et al., 2022)

The figure 1 showed a sample of data log entry which is characterized with alpha-numeric attributes which are not a standard defining message format and hence difficult to decode.

Data Processing and Machine learning model

The data processing focused on the missing data replacement strategy through the application of service-specific procedures. According to Viol et al. (2022), this procedure addresses missing data such as internet protocol (IP) addresses, service names, process identifiers, and splitting service names and component names. This is achieved by converting the original data format into a CSV file."Having processed the data, it was loaded into neural network algorithm for training. The algorithm of the neural network was presented as;

Pseudocode of ANN Algorithm (Algorithm 1)

```
1. Initialize the neural network with input size, hidden size, and output size.
2. Initialize weights (W1, W2) and biases (b1, b2) randomly.
3. Define a sigmoid activation function to introduce non-linearity in the network.
4. Implement the forward pass, which computes the weighted sum of inputs (X) and biases (b1), applies the sigmoid activation function, then computes the weighted sum of hidden layer activations (a1) and
```

biases (b2), and applies the sigmoid activation function again to obtain the final output (a2).

5. Compute the prediction error (loss) using binary cross-entropy.
6. Implement the backward pass, which computes the gradients of the weights (W1, W2) and biases (b1, b2) using the chain rule of calculus and updates them using a specified learning rate.
7. Train the neural network by repeating the forward pass, backward pass, and weight/bias updates for a specified number of epochs.
8. Monitor the loss during training for evaluation.
9. Implement a prediction function that uses the trained weights and biases to make binary predictions (0 or 1) based on input data (X).

High level algorithm for the ANN training (Algorithm 2)

1. Load and pre-process the training data, including features (input variables) and target variable (output variable).
2. Call algorithm 1 and define the architecture with table (see table 2)
3. Compile the model by specifying the optimizer, loss function, and evaluation metrics to be used during training.
4. Train the model using the training data by passing the training data through the neural network in batches, calculating the loss, and updating the model weights using back-propagation algorithm [Bertero et al., 2017].
5. Monitor and record the training progress, such as loss and accuracy, during training to evaluate the model's performance.
6. Evaluate the trained model using the validation data to measure its performance on unseen data and fine-tune hyper-parameters if needed.
7. Repeat steps 4-6 for multiple epochs until the model converges or reaches a certain stopping criterion.
8. Analyze the training results with accuracy
9. Stop

Table 2: Neural network parameters

Parameters	Values
Epoch	100
The network hidden layers	10
Max epoch values	100
No. delayed reference input	1.0
Maximum feature output	3.0
Number of non hidden layers	2.0
Maximum interval per sec	2.0
No. delayed output	1.0
No. delayed feature output	4.0
Minimum reference value	-0.7
Maximum reference value	0.7

Neural network parameters Cont.

3.1 Development of the multi level intrusion detection system

The model of the multi level intrusion detection system was developed to monitor, detect and control the penetration of threat on the log manager. This was achieved using the threat detection algorithm and the control algorithm to develop a multi level intrusion detection system. The table 2 presents the neural network training parameters and algorithm of the intrusion detection, while the flow chart was presented in figure 2;

Intrusion detection Algorithm (3)

1. Start
2. Identify input log entry from user

```
3.  Feed to algorithm 2
4.  Train for threat detection
5.  If
6.      Threat is detected = true
7.          Log Server Rejects Acknowledgement of packet
8.          Send alert notification
9.          Deny data throughput to cloud
10. Else
11.         Acknowledge request
12.         Allow throughput to cloud
13. End if
14. Return to algorithm 1
15. End
```

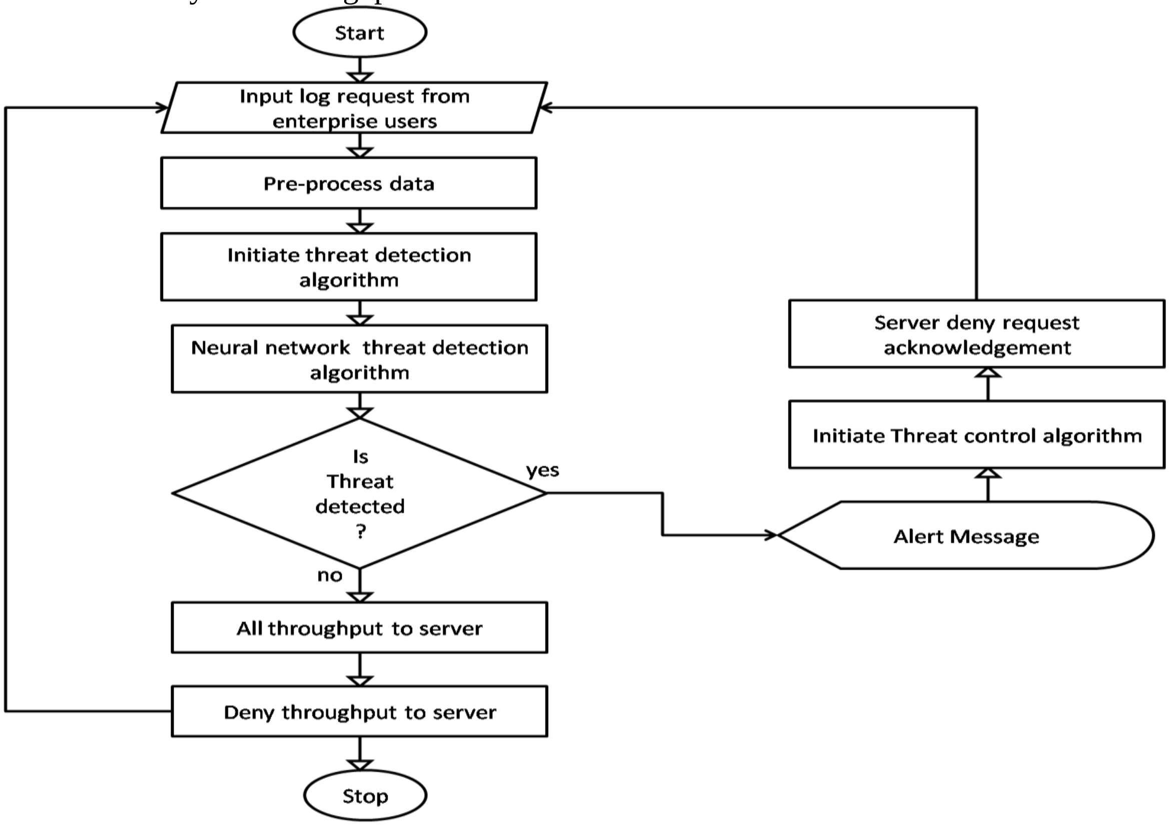

Start
Input log request from enterprise users
Pre-process data
Initiate threat detection algorithm
Neural network threat detection algorithm
Is Threat detected ?
yes
no
All throughput to server
Deny throughput to server
Stop
Alert Message
Initiate Threat control algorithm
Server deny request acknowledgement

Figure 2: Flow chart of the multi level intrusion detection system

The flowchart in figure 2 presented the workflow of the intrusion detection system, showing the logical relationships between the various modules which interacted to achieve the complete system. The data input from the user enterprises as logged into the server, the adopted processing algorithm removed noise with image formats which is common with such data and then the algorithm was used to train and detect threat for control via access denial to the target using the threat control algorithm, else when threat was not detected, throughput is allowed to the server.

PERFORMANCE EVALUATION

The performance of the algorithm developed was evaluated using accuracy as shown in the equation 1, considering the true positive (TP), false positive (FP), true negative (TN), false negative (FN), an true negative (TN) as;

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \qquad 2$$

RESULTS AND DISCUSSION

This section presents the performance of the neural network and also its accuracy for threat detection in the log server. The neural network during the training used the loss function and accuracy to evaluate the performance at every epoch step of the training process. The result was also validated and when consistencies were recorded in the series of training output result, then its stop and generate the model. The result of the training and also the accuracy of threat detection is presented in figure 3;

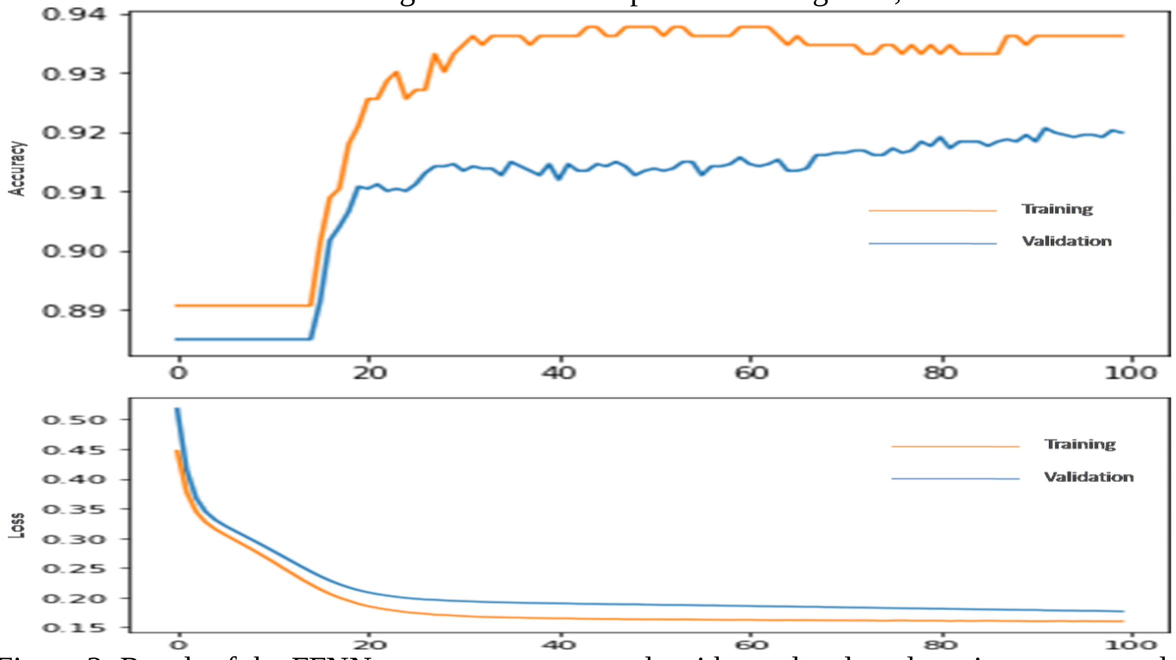


Figure 3: Result of the FFNN

The figure 3 presented the neural network training performance evaluation model which used accuracy and loss to determine the learning progress of the algorithm. The table 3 was used to record the training results and also the validation.

Table 3: Result of Neural Network Training

Epoch	Loss	Training Accuracy	Validation Accuracy
1	0.511	0.9357	0.9107
2	0.413	0.9422	0.9146
3	0.365	0.9394	0.9167
4	0.333	0.9451	0.9185
5	0.342	0.9444	0.9171
6	0.345	0.9445	0.9156
Avg.	0.3848	0.941883	0.915533

The table 3 presented the result of the neural network training for the threat log detection system. The result reported that the training accuracy achieved is 0.94188 and loss of 0.385 respectively. When the results were validated, the accuracy became 0.915. What this mean is that the threat detection algorithm developed with neural network will correctly detect false log entry into the cloud server with 92% accuracy. The given result indicates the performance of a threat detection algorithm developed using a neural network. The training accuracy of the algorithm is reported as 0.94188, which means that during the training process, the algorithm correctly predicted the threat status of the data with an accuracy of 94.188%.

The loss value of 0.385 represents the error between the predicted and actual outputs, with lower values indicating better performance. After training, the algorithm was validated, and the validation accuracy was reported as 0.915, which means that the algorithm correctly predicted the threat status of new, unseen data with an accuracy of 91.5%. This suggests that the algorithm is performing well on data it has not been trained on, indicating its ability to generalize to new data. Additionally, the neural network is capable of correctly identifying false log entries into the cloud server with an accuracy of 92%. This implies that the algorithm has the potential to effectively detect and identify potential threats or false log entries, making it a promising solution for threat detection in the context of cloud server security.

To validate the results, other machine learning algorithms used in training similar cloud log entry data in O'luranti et al. (2020), such as Naïve Bayes, were selected. Additionally, K-Means and Isolation Forest algorithms from Sri and Akhila (2021) were also chosen for comparison with the new neural network algorithm, as shown in Table 4.

Table 4: Comparative analysis

ML algorithm	**Accuracy**
Naive Bayes	0.8564
Neural network	0.9155
K-mean	0.8873
Isolation forest	0.8835

Based on the given data, which shows the accuracy of different machine learning algorithms, we can make the following observations: Naive Bayes: The Naive Bayes algorithm has an accuracy of 0.8564, which means it correctly predicts the outcomes approximately 85.64% of the time. Neural Network: The Neural Network algorithm has the highest accuracy among the given algorithms, with a value of 0.9155, which means it correctly predicts the outcomes approximately 91.55% of the time. Neural networks are known for their ability to learn complex patterns in data, and in this case, it seems to be performing well. K-means: The K-means algorithm has an accuracy of 0.8873, which means it correctly predicts the outcomes approximately 88.73% of the time. K-means is a popular clustering algorithm used for unsupervised learning tasks, and its accuracy is relatively high in this case. Isolation Forest: The Isolation Forest algorithm has an accuracy of 0.8835, which means it correctly predicts the outcomes approximately 88.35% of the time. Isolation Forest is an anomaly detection algorithm, commonly used for identifying outliers or anomalies in data. Its accuracy is also relatively high in this case. Overall, based on the given data, the Neural Network algorithm has the highest accuracy, followed by Naive Bayes, K-means, and Isolation Forest. Figure 4 presents the graph of the result.

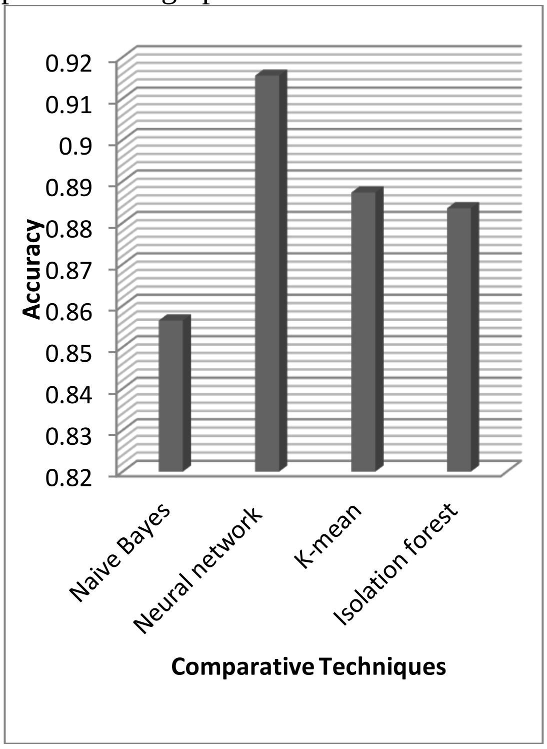


Figure 4: Comparative analysis of ML algorithms

5. CONCLUSION

In conclusion, based on the given data, we can observe that the Neural Network algorithm has the highest accuracy among the given machine learning algorithms, with a value of 0.9155. This indicates that the Neural Network algorithm correctly predicts the outcomes approximately 91.55% of the time, showcasing its ability to learn complex patterns in the data. The Naive Bayes algorithm also performs well with an accuracy of 0.8564, followed by the K-means algorithm with an accuracy of 0.8873, and the Isolation Forest algorithm with an accuracy of 0.8835. These results suggest that all of these algorithms are effective in making accurate predictions in this context. It is important to consider the specific requirements and characteristics of the problem at hand when choosing the most suitable machine learning algorithm for a particular task. Further experimentation and analysis may be necessary to validate the performance and suitability of these algorithms in real-world scenarios.

5.1 Recommendation

Based on the results presented for the neural network training for the threat log detection system, the following recommendations can be made:

a) Utilize the Trained Neural Network Algorithm: The trained neural network algorithm, with a training accuracy of 0.94188 and a loss of 0.385, can be utilized in the threat log detection system. The high training accuracy indicates that the algorithm has learned the patterns in the training data and can accurately predict the threat status of the data during the training process.

b) Validate the Algorithm: The validation accuracy of 0.915 indicates that the algorithm is performing well on new, unseen data, with an accuracy of 91.5%. This suggests that the algorithm has the potential to generalize to real-world scenarios and can effectively predict the threat status of new data. It is recommended to thoroughly validate the algorithm on diverse and representative datasets to ensure its reliability and performance in different scenarios.

c) Consider False Log Entry Detection: The accuracy of 92% in detecting false log entries into the cloud server indicates that the trained neural network algorithm has the potential to effectively identify potential threats or false log entries. This makes it a promising solution for detecting and mitigating security threats in the context of cloud server security. Consider incorporating the algorithm as part of a comprehensive threat detection system to enhance the overall security of the cloud server environment.

d) Monitor Performance and Fine-tune: Continuously monitor the performance of the neural network algorithm in the real-world environment and fine-tune the hyperparameters or model architecture as needed to optimize its performance. This may involve periodic updates to the training data, refining the algorithm's accuracy, and minimizing false positives or false negatives to improve the overall threat detection accuracy.

e) Consider Other Evaluation Metrics: While accuracy and loss are important metrics, consider evaluating the algorithm's performance using other relevant metrics such as precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve to gain a more comprehensive understanding of the algorithm's performance and its effectiveness in real-world scenarios.

6. Acknowledgement

Special thank to Viola, L.; Ronchieri, E. and Cavallaro, C who their study inspired this paper. The authors also thank INFN Tier-1 site administrators who provide log files.

6.1 Conflicts of Interest: The authors declare no conflict of interest.

7. REFERENCES

Alaparthy V., &Morgera S., (2018) "A Multi-Level Intrusion Detection System for Wireless Sensor Networks Based on Immune Theory", 2169-3536 2018 IEEE.Digital Object Identifier 10.1109/ACCESS.2018.2866962

Alruhaily N., & Ibrahim D., (2021) "A Multi-layer Machine Learning-based Intrusion Detection System for Wireless Sensor Networks", (IJACSA) International Journal of Advanced Computer Science and

Applications, Vol. 12, No. 4, 2021http://www.ijacsa.thesai.org/

AmirrezaZ., (2012). "Research on Internet Intrusion Detection System Service in a Cloud", appear in International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012, ISSN (Online): 1694-0814

Anthi E (2022) "Detecting and Defending against Cyber Attacks in a Smart Home Internet of Things Ecosystem", Cardiff University School of Computer Science & Informatics

Anthony T., Toby J., and Robert E. (2010). "Cloud computing — A Practical Approach", 2010.

Axelsson S, (1999). "Research in Intrusion-Detection Systems: A Survey",tech. report TR-98-17, Dept. Computer Eng.,Chalmers Univ. of Technology, 1999.

Bakshi A., &Yogesh B., (2010)"Securing cloud from DDOS Attacks using Intrusion Detection System in Virtual Machine", Second International Conference on Communication Software and Networks, pp. 260-264.

Belavagi M., &Muniyal B., (2016) "Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection", Published by Elsevier B.VTwelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)doi: 10.1016/j.procs.2016.06.016

Bertero, C.; Roy, M.; Sauvanaud, C.; Trédan, G. Experience report: Log mining using natural language processing and application to anomaly detection. In Proceedings of the 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), Toulouse, France, 23–26 October 2017; pp. 351–360.

Bharadwaja S., Sun W., Niamat M., Shen F., (2011) "Collabra: A Xen Hypervisor based Collaborative Intrusion Detection System", Eighth International Conference on Information Technology: New Generations, pp. 695-700.

Dell'Agnello, L.; Boccali, T.; Cesini, D.; Chiarelli, L.; Chierici, A.; Dal Pra, S.; Girolamo, D.; Falabella, A.; Fattibene, E.; Maron, G.; et al. INFN Tier–1: A distributed site. *EPJ Web Conf*. **2019**, *214*, 8002

Dhage S., Meshram B., Rawat R., Padawe S., Paingaokar M., Misra A., (2011) "Intrusion Detection System in Cloud Computing Environment", International Conference and Workshop on Emerging Trends in Technology (ICWET), pp. 235-239.

HodoX., A. Hamilton P.,Dubouilh E., Tachtatzis C., and Atkinson R., (2016) "Threat analysis of IoT networks Using Artificial Neural Network Intrusion Detection System," in 2016 3rd International Symposium on Networks, Computers and Communications (ISNCC), 2016, pp. 1–6.

Jun H., Min W., and Jung H., (2011) "Multi-level Intrusion Detection and Log Management in Cloud computing", IEEE computer society, pp 552-555, Feb.2011.

Kento S, Hitoshi S., Satoshi M, (2009). "A Model-based Algorithm for Optimizing I/O Intensive Applicationsin Clouds using VM-Based Migration", 9th IEEE/ACM International Symposium, Cluster Computing and Grid, 2009.

Mazzariello C., Bifulco R. and Canonico R. (2010) "Integrating a Network IDS into an Open-Source Cloud Computing Environment", Sixth International Conference on Information Assurance and Security, pp. 265-270.

Mkuzhalisai and Gayathri G., (2012). "Research on Enhanced Security In Cloud With Multi-Level Intrusion Detection System", appear in International Journal of Computer and Communication Technology (IJCCT) ISSN (ONLINE): 2231 - 0371 ISSN (PRINT): 0975 –7449 Vol-3, Iss-3, 2012.

Olofin B., (2021) “A review of recent trends in intelligent agent technology” IJICST; Vol 6; Issue 2; pp. 90-98.

Oluranti Jonathan, Sanjay Misra, Victor Osamor (2020)”Comparative Analysis of Machine Learning techniques for Network Traffic Classification” 4th International Conference on Science and Sustainable Development (ICSSD 2020) IOP Conf. Series: Earth and Environmental Science 655 (2021) 012025 IOP Publishing doi:10.1088/1755-1315/655/1/012025

Smys S., Basar A., & Wang H., (2020) “Hybrid Intrusion Detection System for Internet of Things (IoT)”, Journal of ISMAC (2020) Vol.02/ No.04 Pages: 190-199 http://irojournals.com/iroismac/ DOI: https://doi.org/10.36548/jismac.2020.4.002

Sonu D., (2020) “Understanding of Intrusion Detection System for Cloud Computing with Networking System” International Journal of Computer Science and Mobile Computing, Vol.9 Issue.3, March- 2020, pg. 19-25

Sri Sai Manoj Kommineni, Akhila Dindi (2021)”Automating Log Analysis” Master of Science in Computer Science; Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

Vieira K., Schulter A., Carlos B. Westphall, and C. Westphall M. (2010), “Intrusion Detection for Grid and Cloud Computing”, IEEE Computer Society, pp. 38-43.

Viola, L.; Ronchieri, E.; Cavallaro, C. Combining Log Files and Monitoring Data to Detect Anomaly Patterns in a Data Center. Computers 2022, 11, 117. https://doi.org/10.3390/computers11080117 Academic Editor: Leandros Maglaras

Wang, C.; Viswanathan, K.; Choudur, L.; Talwar, V.; Satterfield, W.; Schwan, K. Statistical techniques for online anomaly detection in data centers. In Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, Dublin, Ireland, 23–27 May 2011; pp. 385–392.