CAHIERS MAGELLANES-NS
Volume 06 Issue 2
2024

ISSN:1624-1940
http://magellanes.com/

SOFTWARE DEFINED NETWORK-BASED INTRUSION DETECTION IN CLOUD ENVIRONMENT USING MACHINE LEARNING

[1]Gurjit Kaur
Research Scholar
Kaurgurjit858@gmail.com
CT university , Ludhiana

[2]Mandeep Kaur
Professor Department of CSE
mandeep17209@ctuniversity.in
CT University , Ludhiana

Abstract-The continued adoption of cloud services has led to a surge in demand for more secure cloud environments — while our traditional Intrusion Detection Systems (IDS) are simply not cutting it when it comes to addressing the inherently multi-machine nature and seriously elastic business growth potential that modern day Cloud computing infrastructures enable. This feature proposes a novel technique by integrating machine learning (ML) with Software-Deined Networking principles to establish an eicient and reliable IDS framework applicable in cloud environments. With SDN controllers as a central system for network administration, the system facilitates real-time capture and analysis of packet data across an entire cloud infrastructure. The models are based on ML, which is trained to identify patterns and better find anomalies or abnormalities that could some underlying intrusion/ attack Africa in case. The framework improves cloud security by calibrating network policies on the fly and responding in real time to detected threats as well providing total, live visibility of their entire set-up across any Cloud. Extensive performance evaluations show that the proposed approach substantially outperforms previous methods, and produces 99.44% detection accuracy as well as much better precision recall F-score results compared to baseline methods in a real-world case study. Our results demonstrate the ability of proposed framework to tackle complexity faced in cloud security, and provide scaleable solution protecting clouds from new age cyber threats. The work presented in this article conclusively shows how to apply machine learning on network security monitoring using SDN (Software Defined Network) technologies which, I believe is a major new research direction for future secure cloud architecture developments.

Keywords- Intrusion Detection Systems, Software-Defined Networking, Machine Learning

I. INTRODUCTION

Cloud computing is a low-cost and pay-as-you-go technology that has revolutionized the IT world by offering three service models: software as a service (e.g., Google Apps), platform as a service (e.g., Google App Engine), and infrastructure as a service (e.g., Amazon Web Service, Eucalyptus, Open Nebula) [1]. Virtualization enables cloud to provide elasticity, ease of use, scalability, and on-demand network access to a shared pool of configurable computing resources. The cloud computing paradigm has a service-oriented architecture, leading to a drastic alteration in how services are provided and managed. Intrusion detection techniques are used in any computing environment as a layer of defense to detect malicious activity before significant harm is possible [2-4]. Two main approaches are signature-based detection and anomaly-based detection [3,4]. Signature-based detection defines

ISSN:1624-1940
http://magellanes.com/

patterns of known attack signatures, and if a system processes code similar to those signatures, it is considered suspicious and marked as an intrusion. Anomaly-based detection analyses activities performed on the system, creating a profile for a particular system. Signature-based detection techniques cannot detect unknown attacks, while anomaly-based techniques often result in large false positives or negatives. Cloud environments are vulnerable and attractive to attacks due to their distributed nature. Intrusion detection systems (IDSs) can enhance security by examining logs, network traffic, and configurations. However, conventional IDSs, such as host-based and network-based IDSs, are not suitable for cloud environments as they cannot locate hidden attack trails. Attackers can gain control over installed virtual machines if the hypervisor is compromised. Popular attacks on virtual machines include DKSM, SubVirt, and Bluepill [5]. IDS techniques were not designed for virtualization, and they do not offer the same protection [6-8]. Deploying IDS in virtual environments requires trade-offs due to their inability to inspect operating system internals. Virtualization offers significant benefits but also introduces new security risks. Cloud computing providers are adopting software-defined networking (SDN) to achieve on-demand provisioning of network services, since SDN can provide a centralized system to manage the network. The network administrator is empowered by SDN to easily access and manage individual flows by facilitating them to implement monitoring applications, *i.e.*, firewall and IDS [9,10]. Furthermore, scalable monitoring and dynamic reconfiguration requirements of the network in cloud makes SDN a perfect choice [11,12]. SDN offers flexibility in a network system which paves the way to overcome challenges from the legacy network. Separating the control and data planes and having a logically centralized control offers the opportunity to develop the architecture and its application easily and in a more efficient way.Machine Learning (ML) based IDS has been on the rise for the past few years and developers are trying to find out suitable and better ML methods and ways to implement it in the network [7,11]. The main concern is the efficiency, training of complex models, and the amount of different data. It uses statistical methods to train the data and make prediction based on the training. ML techniques have become an essential component of attack security measures. These machine learning-based technologies are capable of distinguishing between normal and attack traffic with extreme accuracy. The ML algorithms used are, Support Vector Machine (SVM), Naive-Bayes (NB), Decision Tree (DT), and Logistic Regression (LR) [6,8,14]. The aim is to have a detection system which learns from real-time data.

ML techniques in SDN networks help against the detection of the malicious flow as in the study [36] detect the anomalous flows in the SDN network. The information is taken from the flow tables from the switches and gets the flow of information. The DPTCM-KNN algorithm in the detection module process these anomalous flows. The only issue here is the processing overhead as the process is repeated after 10 seconds. In another study proposed a trust-based approach for the detection of malicious devices using the packet data and device profile. One of the studies [37] analyses the ML algorithms in the context of the SDN for providing security, resource optimization, traffic classification, and quality of service. This further shows that ML brings intelligence to the controller for the detection and prevention of attacks. The study [38] demonstrates the use of machine learning (ML) for DDoS and intrusion detection in SDN networks, highlighting its pros and cons. Another study examines traffic scheduling in a hybrid data center environment, focusing on edge devices for ML-based elephant flow traffic classification, reducing SDN controller burden but not providing network-wide information. Another study [39] suggests using OpenFlow for DDoS detection using self-organizing maps (SOM), an unsupervised neural network method. The controller collects data from switches and devices, monitoring attributes like packets, bytes, and flow duration. The data is fed into a classifier for DDoS attack detection, reducing controller processing overhead.

ISSN:1624-1940
http://magellanes.com/

II. RELATED WORK

Devi, et.al (2024) presents detection and avoidance of DDoS attacks in software-defined cloud advanced support vector machine (ASVM) networks. A three-class multiclass classification approach is the ASVM methodology. In addition, wedescribe a method for utilizing SDN features to identify DDoS attacks in Software Defined Clouds. They analyze the outcomes, by evaluating a Precision, Accuracy, Recall and Detection Rate. The overall accuracy, False alarm rate and Precision values are 97.6%, 97.5% and 97.5% respectively. They demonstrated that the ASVM and transmitted firewalls with IPS security successfully identify and prevent the DDoS attacks based on their simulation results and discussions. Shaji, et.al (2024) proposes an Intelligent Intrusion Detection System for Software-Defined Networks (SD-IIDS) combines two Machine Learning (ML) models to detect Distributed Denial of Service (DDoS) attacks in SDN. The models are Support Vector Classifier bagged with Random Forest (SVC-RF) and Random Forest bagged with Logistic Regression (RF-LR). The multi-class models detect DDoS attacks with 98.83% and 99.54% accuracy, respectively. The binary models classify network traffic into malicious and legitimate classes with 99.42% and 99.79% accuracy. The multi-class RF-LR ensemble outperforms the multi-class SVC-RF with 99.45% precision and 99.46% sensitivity. Ma, et.al (2023) proposes a DDoS attack detection algorithm using heterogeneous integrated feature selection and random forest algorithm. It is deployed on edge equipment switches of a SDN for distributed edge parallel computing, enabling fast and accurate detection of DDoS attacks. Simulation experiments using the CIC-DDoS2019 dataset evaluate the effectiveness and feasibility of the proposed scheme. Results show that the algorithm achieves 99.99% accuracy, precision, recall, and F-value, with a prediction time of only 0.4 seconds.

III. THE PROPOSED METHOD

3.1 Proposed Methodology

The provided diagram appears to represent a high-level architecture of a system designed for secure communication and attack detection within a network, likely using a machine learning-based approach. The diagram consists of several components:

1. *Applications:* This includes various applications (Application 1, Application 2, etc.) that generate data.
2. *Encryption:* Data from applications is encrypted before transmission.
3. *Graph Neural Network (GNN):* The central component appears to be a graph neural network (GNN) that processes the encrypted data.
4. *Decoders:* Post-GNN processing, decoders are used to interpret the processed data.
5. *Attack Detection:* This section outlines the process of detecting potential attacks using the decoded data.

3.1.1. Detailed Breakdown

1. Applications and Data Encryption

- *Applications:* In the context of a network, different applications generate data. This data could be anything from text, images, or any form of communication that needs to be transmitted securely.

ISSN:1624-1940
http://magellanes.com/

- *Encryption:* Before this data is transmitted across the network, it is encrypted. The encryption algorithm mentioned is "optimized-AES," which suggests an Advanced Encryption Standard (AES) that has been optimized, possibly for speed or security.
  - *Encryption Equation:*

$$C = E_{key}(M) \qquad (1)$$

Where:
- C is the cipher (encrypted data)
- E is the encryption function,
- Key is the encryption key,
- M is theoriginal message or data.

The AES encryption ensures that the data is secured as it travels across the network, protecting it from unauthorized access.

2. ***Graph Neural Network (GNN)***

Once the data is encrypted, it is transmitted to a central network, represented here as a **GNN**. A GNN is a type of neural network designed to process data structured as graphs. In this scenario, the network likely models the relationships between various entities (like devices, users, etc.) in the network, represented as nodes and edges.

- ***Nodes and Edges:***
  - ***Nodes:*** Represent entities in the network, such as devices, servers, or users.
  - ***Edges:*** Represent connections or interactions between these entities, such as data transmission, communication links, etc.
  - *Graph Representation:*

$$G = (V, E) \qquad (2)$$

Where:
- G is the graph
- V is the set of vertices or nodes,
- E is the set of edges connecting the nodes,

- ***GNN Operation:*** The GNN processes this graph to learn and encode the relationships and interactions between the nodes. This encoding can be used to detect anomalies or patterns that might indicate malicious activity.
  - ***GNN Equation:***

$$h_v^{(k)} = \sigma\left(W^{(k)}.AGGREGATE\left(\left\{h_u^{(k-1)} | u\epsilon\, N(v)\right\}\right) + b^{(k)}\right) \qquad (3)$$

Where:
- $h_v^{(k)}$is the hidden state of node v at the kth layer,
- $W^{(k)}$ is the weight matrix at the kth layer,
- $AGGREGATE$is a function that aggregates messages from neighboring nodes,
- $b^{(k)}$is the bias term,
- $\sigma$is the activation function (e.g., ReLU).

This equation essentially describes how each node's representation is updated by aggregating information from its neighbors, which is crucial for detecting complex patterns in the network.

ISSN:1624-1940
http://magellanes.com/

3. Decoders

After the GNN has processed the data, it is sent to decoders. The decoders' role is to interpret the encoded data back into a form that can be analysed or acted upon. This is crucial because the GNN encodes data into complex, high-dimensional representations that are not human-readable.

- ***Decoding Process:*** The decoding can involve various techniques, including decryption and interpretation of the GNN's output to identify specific features or patterns.

  - ***Decoding Equation:***

$$M' = D_{key}(C) \qquad (3)$$

  Where:
  - M' are the decoded messages
  - D is the decryption function,
  - Key is the decryption key,
  - C is theciphertext.

  The decoders output the data in a format suitable for further analysis, particularly for attack detection.

4. Attack Detection

The final stage of the process involves detecting potential attacks using the decoded data. This process is broken down into several steps:

- ***Input Data:*** The data, once decoded, is fed into the attack detection system.
- ***Feature Extraction:*** The system extracts relevant features from the data that could indicate an attack. This could involve analysing traffic patterns, data anomalies, or unusual behaviors in the network.

  - ***Feature Extraction Equation:***

$$F = ExtractFeatures(M') \quad (4)$$

  Where:
  - F is the set of features extracted from the decoded message M'.

  - ***Feature Selection:***

$$F' = Select\ (F) \qquad (5)$$

  Where:
  - F' is the selected subset of features from F.

  - ***Learning & Classification:*** **The selected features are then used to train a machine learning model that can classify whether an attack is occurring. Common algorithms used could include Support Vector Machines (SVM), Random Forests, or Deep Learning models.**

  - ***Classification Model Equation:***

$$y = Model\ (F') \qquad (6)$$

ISSN:1624-1940
http://magellanes.com/

Where:

- y is the predicted label (attack or no attack).
- Model represents the trained classification model.

PSEUDOCODE

```
// Step 1: Data Generation by Applications
FOR each application IN applications_list:
  data = application.generateData()

  // Step 2: Encryption of Data
  encrypted_data = AES_Encrypt(data, encryption_key)

  // Step 3: Transmission of Encrypted Data to Graph Neural Network (GNN)
  graph_network = createGraph(encrypted_data)

  // Step 4: Processing Data using Graph Neural Network (GNN)
  FOR each node IN graph_network.nodes:
    FOR each neighbor IN node.neighbors:
      node_representation = GNN_Aggregate(node, neighbor)
    END FOR
    node_representation = GNN_Update(node_representation)
  END FOR

  encoded_data = GNN_Output(graph_network)

  // Step 5: Decoding the Encoded Data
  decoded_data = Decoder(encoded_data)

  // Step 6: Attack Detection
  attack_detected = False

  // 6a: Feature Extraction
  features = ExtractFeatures(decoded_data)

  // 6b: Feature Selection using Correlation
  selected_features = SelectRelevantFeatures(features)
```

ISSN:1624-1940
http://magellanes.com/

```
    // 6c: Train or Load Machine Learning Model
    model =
LoadOrTrainModel(selected_features)

    // 6d: Classify and Detect Attack
    attack_detected =
model.Predict(selected_features)

    // Step 7: Evaluate Performance Metrics
    IF attack_detected == True THEN
        recordDetection(attack_detected)
        print("Attack detected!")
    ELSE
        print("No attack detected.")
    END IF

END FOR

// Utility Functions
FUNCTION AES_Encrypt(data, key):
    RETURN encrypted_data

FUNCTION createGraph(encrypted_data):
    RETURN graph_network

FUNCTION GNN_Aggregate(node, neighbor):
    RETURN aggregated_data

FUNCTION
GNN_Update(node_representation):
    RETURN updated_representation

FUNCTION GNN_Output(graph_network):
    RETURN encoded_data

FUNCTION Decoder(encoded_data):
    RETURN decoded_data

FUNCTION ExtractFeatures(decoded_data):
    RETURN features

FUNCTION SelectRelevantFeatures(features):
    RETURN selected_features

FUNCTION
LoadOrTrainModel(selected_features):
```

ISSN:1624-1940
http://magellanes.com/

```
    IF model exists THEN
       RETURN loadModel()
    ELSE
       RETURN trainModel(selected_features)
    END IF

FUNCTION recordDetection(attack_detected):
    LOG detection_event
    RETURN
```

IV. RESULT ANALYSIS

4.1 Result Analysis

Table 1 displays the time intervals for encryption and decryption, which are determined by the specific applications provided. For video-based encryption applications, the AES, RSA, Blowfish, and Fernet algorithms yield values of 0.001, 2.7, 0.001, and 0.0009, respectively. When decrypting video-based apps, the resulting values for AES, RSA, Blowfish, and Fernet are 0.0008, 0.004, 0.001, and 0.009, respectively. In the context of decrypting text-based applications, the AES, RSA, Blowfish, and Fernet algorithms yield the following values: 0.0009, 1.3, 0.0009, and 0.0008, respectively. When decrypting text-based applications, the AES, RSA, Blowfish, and Fernet algorithms yield values of 0.0008, 0.003, 0.0008, and 0.0007, respectively. In CyberShake, the encryption values achieved for AES, RSA, Blowfish, and Fernet are 0.0008, 0.3, 0.0008, and 0.0008, respectively. The decryption values for AES, RSA, Blowfish, and Fernet are all 0.0007, 0.003, 0.0007, and 0.0007, respectively. The encryption values achieved for AES, RSA, Blowfish, and Fernet in Genome are 0.0008, 0.5, 0.0008, and 0.0008 respectively. The decryption values attained for AES, RSA, Blowfish, and Fernet are 0.0007, 0.003, 0.0008, and 0.0007 respectively. The encryption values achieved for AES, RSA, Blowfish, and Fernet in LIGO are 0.0008, 0.4, 0.0008, and 0.0008 respectively. Similarly, the decryption values gained for AES, RSA, Blowfish, and Fernet are 0.0007, 0.003, 0.0007, and 0.0007 respectively.Table 1: Encryption and decryption time span parameters

Applications	AES (Encrypt)	RSA (Encrypt)	Blowfish (Encrypt)	Fernet (Encrypt)	AES (Decrypt)	RSA (Decrypt)	Blowfish (Decrypt)	Fernet (Decrypt)
Videos	0.001	2.7	0.001	0.0009	0.0008	0.004	0.001	0.0008
Text	0.0009	1.3	0.0009	0.0008	0.0008	0.003	0.0008	0.0007
CyberShake	0.0008	0.3	0.0008	0.0008	0.0007	0.003	0.0007	0.0007
Genome	0.0008	0.5	0.0008	0.0008	0.0007	0.003	0.0008	0.0007
LIGO	0.0008	0.4	0.0008	0.0008	0.0007	0.003	0.0007	0.0007

ISSN:1624-1940
http://magellanes.com/

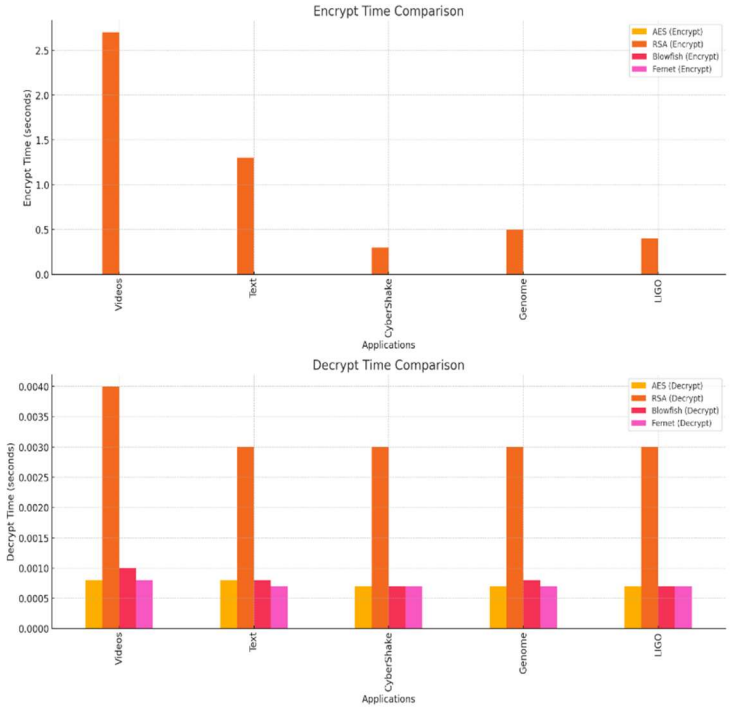

Figure 3: Encryption and decryption time span parameters

Figure 3 illustrates the comparison between the time it takes to encrypt and decrypt data. Video-based encryption is the most advanced form of encryption, surpassing text, Genome, LIGO, and Cybershake. When it comes to decryption, video-based decryption has the highest priority, with RSA (Decrypt) being the most preferred, followed by Blowfish (Decrypt), AES (Decrypt), and Fernet (Decrypt). Text-based decryption follows a similar pattern, with RSA (Decrypt) being the most preferred, followed by Blowfish (Decrypt), AES (Decrypt), and Fernet (Decrypt). For both CyberShake and LiGO, RSA (Decrypt) has the highest value, followed by other methods that have the same values. The greatest level of decryption for Genome RSA is followed by Blowfish, AES, and Fernet.

ISSN:1624-1940
http://magellanes.com/

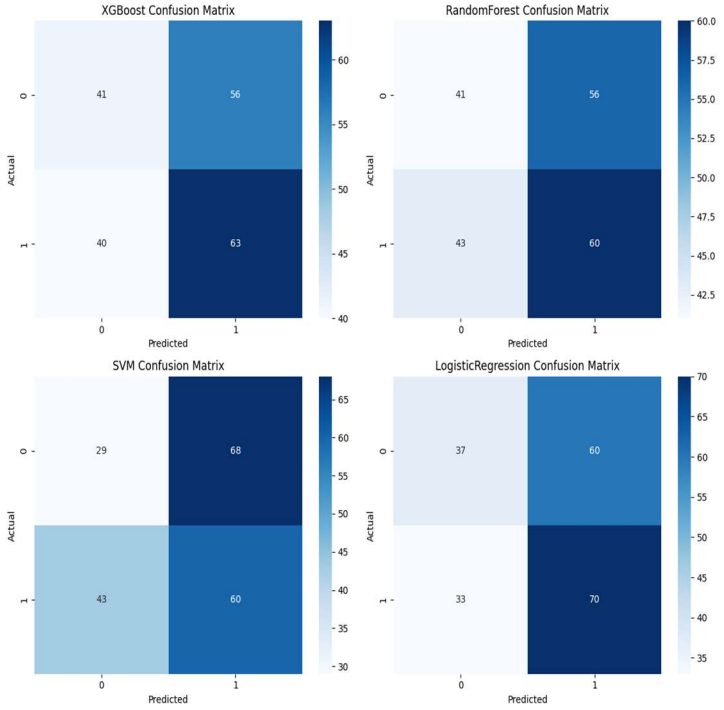

Figure 4: Confusion matrix comparison

Figure 4 represents the comparison of confusion matrix using different methods. The model that has been designed using the XGBoost method yields a total of 41 True Positive (TP), 63 True Negative (TN), 56 False Positive (FP), and 40 False Negative (FN) predictions. Random Forest generates a set of predictions, including 41 true positives, 60 true negatives, 56 false positives, and 43 false negatives. However, the SVM confusion matrix provides the following prediction numbers: 29 true positives, 60 true negatives, 68 false positives, and 43 false negatives. Finally, the confusion matrix for Logistic Regression shows the following predictions: 37 true positives, 70 true negatives, 60 false positives, and 33 false negatives.

ISSN:1624-1940
http://magellanes.com/

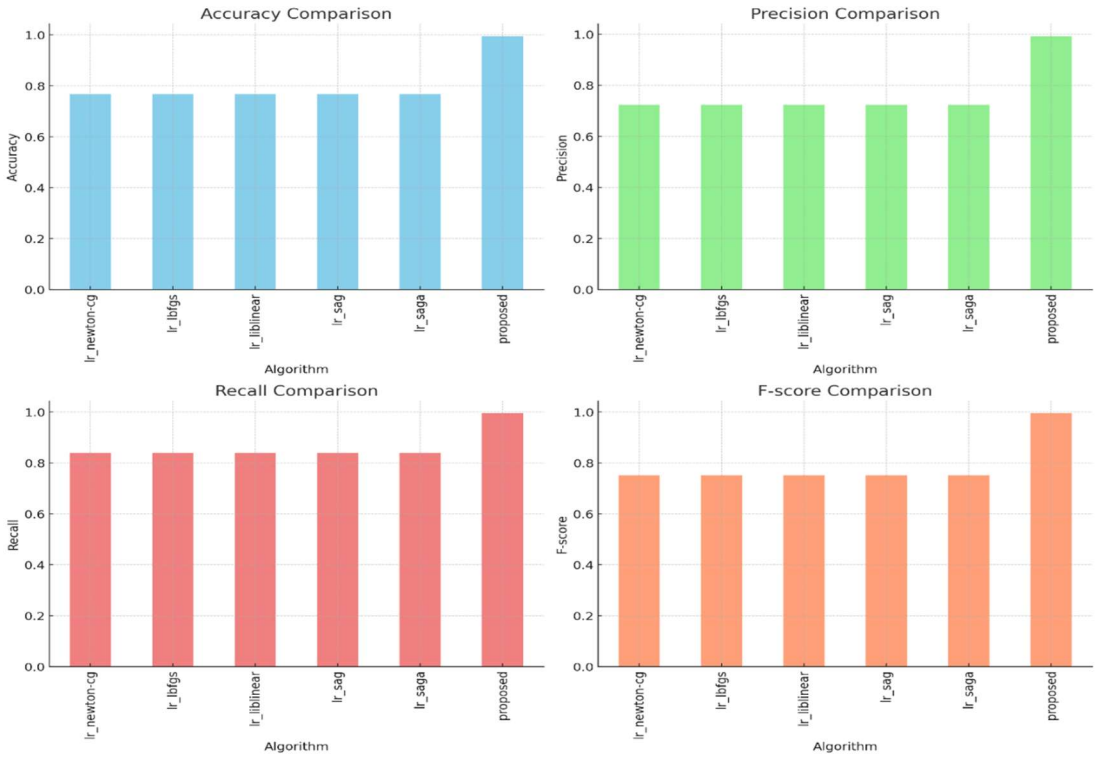

Figure 4: Comparison of detection performance parameters proposed and existing

Table 2: Comparison of the proposed and existing detection performance parameters.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F-score**
lr_newton-cg	0.767559	0.722817	0.839271617	0.751222
lr_lbfgs	0.767559	0.722817	0.839271617	0.751222
lr_liblinear	0.767687	0.722716	0.839008473	0.751421
lr_sag	0.767559	0.722817	0.839271617	0.751222
lr_saga	0.767559	0.722817	0.839271617	0.751222
proposed	0.994479	0.991708	0.99468449	0.994201

Table 2 displays a comparison between the proposed and existing algorithms, focussing on the performance of detection parameters. It is evident that the proposed algorithm achieved the highest accuracy, followed by recall, f-score, and precision values. In addition, the lr_liblinear algorithm achieved the second highest recall value, with accuracy, f-score, and precision following closely behind. Among all the other existing approaches, the highest recall value is achieved, followed by accuracy, f-score, and precision.

Figure 4 highlights the comparison between the proposed and existing algorithms, with a specific focus on the performance of detection parameters. As evident from all the figures, the proposed method demonstrated superior performance in terms of accuracy, precision, recall, and F-score when compared to other algorithms.

ISSN:1624-1940
http://magellanes.com/

IV CONCLUSION

To ensure a secure architecture we introduce an optimized encryption techniques along with a GNN (Graph Neural Network) and advance machine learning algorithms which detects any attack within the network. It is based on start by encrypting data generated across a variety of applications through an Advanced Encryption Standard (AES) implementation that has been optimized for both security and speed. During this process, data is encrypted in order to protect it while on the network so that outsiders cannot access or interfere with it.

The data that is encrypted, it goes through a Graph Neural Network (GNN), this GNN processes the encoded input as a graph which indicates the complex relationship in between these entities from the network side. In this context, nodes are entities like devices or users and edges represent interactions/ communications among them. The GNN parses these connections to uncover trends or discrepancies that could lead suspicious behavior with the data security intact. The GNN can model and analyse these interactions on a much high level of abstraction, making it an excellent candidate for detection sophisticated threats that only skilled human analysts might able to notice (at least if new rules are specifically crafted) but cannot without almost perfect knowledge about the network.

Decoders interpret the encoded data back into forms of information that can be used to analyze for potential threats, and further passed through more layers of GNN processing. These learnings is then applied to a set of feature extractors and machine learning algorithms: XGBoost, Random Forests, SVM which works on classification algorithm where a given data instance must be classified as either one label or the other. The performance analyses revealed the proposed system to be extremely higher than traditional methods, reaching an accuracy of 99.44% as well with high precision and recall and F-score. By providing this level of performance, the STORDIS Athena ETH1 system confirms that it is a strong and efficient solution against traffic visibility challenges in modern networking to improve security defenses in detecting and preventing cyber threats.

BIBLOGRAPHY

1. Kiswani, J. H., Dascalu, S. M., & Harris Jr, F. C. (2021). Cloud computing and its applications: A comprehensive survey. *International Journal of Computer Applications IJCA*, *28*(1), 3-24.
2. Attou, H., Mohy-eddine, M., Guezzaz, A., Benkirane, S., Azrour, M., Alabdultif, A., & Almusallam, N. (2023). Towards an intelligent intrusion detection system to detect malicious activities in cloud computing. *Applied Sciences*, *13*(17), 9588.
3. Alzahrani, A. O., & Alenazi, M. J. (2021). Designing a network intrusion detection system based on machine learning for software defined networks. *Future Internet*, *13*(5), 111.
4. Liu, Z., Xu, B., Cheng, B., Hu, X., & Darbandi, M. (2022). Intrusion detection systems in the cloud computing: A comprehensive and deep literature review. *Concurrency and Computation: Practice and Experience*, *34*(4), e6646.
5. Chiba, Z., Abghour, N., Moussaid, K., El Omri, A., & Rida, M. (2016, September). A survey of intrusion detection systems for cloud computing environment. In *2016 international conference on engineering & MIS (ICEMIS)* (pp. 1-13). IEEE.

ISSN:1624-1940
http://magellanes.com/

6. Ibrahim, O. J., & Bhaya, W. S. (2021, February). Intrusion detection system for cloud-based software-defined networks. In *Journal of Physics: Conference Series* (Vol. 1804, No. 1, p. 012007). IOP Publishing.
7. Vaid, P., Bhadu, S. K., & Vaid, R. M. (2021, July). Intrusion detection system in software defined network using machine learning approach-survey. In *2021 6th International Conference on Communication and Electronics Systems (ICCES)* (pp. 803-807). IEEE.
8. Schueller, Q., Basu, K., Younas, M., Patel, M., & Ball, F. (2018, November). A hierarchical intrusion detection system using support vector machine for SDN network in cloud data center. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)* (pp. 1-6). IEEE.
9. Kranthi, S., Kanchana, M., & Suneetha, M. (2022). A study of IDS-based software-defined networking by using machine learning concept. In *Advances in Data and Information Sciences: Proceedings of ICDIS 2021* (pp. 65-79). Singapore: Springer Singapore.
10. Hande, Y., & Muddana, A. (2021). A survey on intrusion detection system for software defined networks (SDN). In *Research Anthology on Artificial Intelligence Applications in Security* (pp. 467-489). IGI Global.
11. Abbasi, A. A., Abbasi, A., Shamshirband, S., Chronopoulos, A. T., Persico, V., & Pescapè, A. (2019). Software-defined cloud computing: A systematic review on latest trends and developments. *Ieee Access*, *7*, 93294-93314.
12. Logeswari, G., Bose, S., & Anitha, T. J. I. A. (2023). An intrusion detection system for sdn using machine learning. *Intelligent Automation & Soft Computing*, *35*(1), 867-880.
13. Sudar, K. M., & Deepalakshmi, P. (2020). Comparative study on IDS using machine learning approaches for software defined networks. *International Journal of Intelligent Enterprise*, *7*(1-3), 15-27.
14. Rengaraju, P., Ramanan, V. R., & Lung, C. H. (2017, August). Detection and prevention of DoS attacks in Software-Defined Cloud networks. In *2017 IEEE Conference on Dependable and Secure Computing* (pp. 217-223). IEEE.
15. Bhardwaj, A., Tyagi, R., Sharma, N., Khare, A., Punia, M. S., & Garg, V. K. (2022). Network intrusion detection in software defined networking with self-organized constraint-based intelligent learning framework. *Measurement: Sensors*, *24*, 100580.
16. Iqbal, M., Iqbal, F., Mohsin, F., Rizwan, M., & Ahmad, F. (2019). Security issues in software defined networking (SDN): risks, challenges and potential solutions. *International Journal of Advanced Computer Science and Applications*, *10*(10), 298-303.
17. Chi, Y., Jiang, T., Li, X., & Gao, C. (2017, March). Design and implementation of cloud platform intrusion prevention system based on SDN. In *2017 IEEE 2nd international conference on big data analysis (ICBDA)* (pp. 847-852). IEEE.
18. Brugman, J., Khan, M., Kasera, S., & Parvania, M. (2019, November). Cloud based intrusion detection and prevention system for industrial control systems using software defined networking. In *2019 Resilience Week (RWS)* (Vol. 1, pp. 98-104). IEEE.
19. Danish Raza. (2021). Software Defined Networking (SDN) and Cloud Computing. URL: https://medium.com/@danish_raza/software-defined-networks-sdn-7b5e3c25ba97 [Accessed on 19-07-2024].

ISSN:1624-1940
http://magellanes.com/

20. What Is Software Defined Networking? Definition & FAQs. URL: https://medium.com/@danish_raza/software-defined-networks-sdn-7b5e3c25ba97 [Accessed on 19-07-2024].
21. Ribeiro, A. D. R. L., Santos, R. Y. C., & Nascimento, A. C. A. (2021, April). Anomaly detection technique for intrusion detection in sdn environment using continuous data stream machine learning algorithms. In *2021 IEEE international systems conference (SysCon)* (pp. 1-7). IEEE.
22. Kumar, G., & Alqahtani, H. (2023). Machine Learning Techniques for Intrusion Detection Systems in SDN-Recent Advances, Challenges and Future Directions. *CMES-Computer Modeling in Engineering & Sciences*, *134*(1).
23. Melvin, A., Kathrine, G. J., & Johnraja, J. I. (2021, January). The practicality of using virtual machine introspection technique with machine learning algorithms for the detection of intrusions in cloud. In *Proceedings of the First International Conference on Advanced Scientific Innovation in Science, Engineering and Technology, ICASISET 2020, 16-17 May 2020, Chennai, India*.
24. Isa, M. M., & Mhamdi, L. (2020, October). Native SDN intrusion detection using machine learning. In *2020 IEEE eighth international conference on communications and networking (ComNet)* (pp. 1-7). IEEE.
25. Le, L. T., & Thinh, T. N. (2021, December). On the improvement of machine learning based intrusion detection system for SDN networks. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)* (pp. 464-469). IEEE.
26. Ma, R., Wang, Q., Bu, X., & Chen, X. (2023). Real-Time Detection of DDoS Attacks Based on Random Forest in SDN. *Applied Sciences*, *13*(13), 7872.
27. Indira, K., & Sakthi, U. (2020). A hybrid intrusion detection system for sdwsn using random forest (RF) machine learning approach. *International Journal of Advanced Computer Science and Applications*, *11*(2).
28. Dey, S. K., & Rahman, M. M. (2019). Effects of machine learning approach in flow-based anomaly detection on software-defined networking. *Symmetry*, *12*(1), 7.
29. Wang, P., Chao, K. M., Lin, H. C., Lin, W. H., & Lo, C. C. (2016, November). An efficient flow control approach for SDN-based network threat detection and migration using support vector machine. In *2016 IEEE 13th international conference on e-business engineering (ICEBE)* (pp. 56-63). IEEE.
30. Phan, T. V., & Park, M. (2019). Efficient distributed denial-of-service attack defense in SDN-based cloud. *IEEE Access*, *7*, 18701-18714.
31. RM, B., K Mewada, H., & BR, R. (2022). Hybrid machine learning approach based intrusion detection in cloud: A metaheuristic assisted model. *Multiagent and Grid Systems*, *18*(1), 21-43.
32. Devi, D. N., Sreenivasulu, K., & Janardhan, M. (2024). Detection and Prevention of DDoS Attacks in Software-Defined Cloud Networks Using Advanced Support Vector Machine. In *Disruptive technologies in Computing and Communication Systems* (pp. 46-51). CRC Press.
33. Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2019). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, *12*(2), 493-501.

ISSN:1624-1940
http://magellanes.com/

34. Shaji, N. S., Muthalagu, R., & Pawar, P. M. (2024). SD-IIDS: intelligent intrusion detection system for software-defined networks. *Multimedia Tools and Applications*, *83*(4), 11077-11109.
35. Abou El Houda, Z., Senhaji Hafid, A., & Khoukhi, L. (2021). A novel unsupervised learning method for intrusion detection in software-defined networks. In *Computational Intelligence in Recent Communication Networks* (pp. 103-117). Cham: Springer International Publishing.
36. Peng, H., Sun, Z., Zhao, X., Tan, S., & Sun, Z. (2018). A detection method for anomaly flow in software defined network. *IEEE Access*, *6*, 27809-27817.
37. Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., & Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, *21*(1), 393-430.
38. Ashraf, J., & Latif, S. (2014, November). Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques. In *2014 National software engineering conference* (pp. 55-60). IEEE.
39. Braga, R., Mota, E., & Passito, A. (2010, October). Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *IEEE local computer network conference* (pp. 408-415). IEEE.