INTEGRATING MACHINE LEARNING ALGORITHMS AND BLOCKCHAIN IN EDGE DEVICE

Ms. T. Sathya[1], Mr. K. Ramkishore[2], Mr. J. Sanjay Kumar[3], Mr. M. Saravanan[4]

Professor[1], UG Scholar[2],[3],[4], Department of Cyber Security, SRM

Valliammai Engineering College

 DOA:15.05.2025 / DOP: 20.05.2025 ISSN: 3048-619X
Volume 2 Issue 2 [Apr – July 2025] DOI:

ABSTRACT:

The Blockchain-Machine Learning Integration (BMLI) proposes a decentralized approach to data processing and security for edge devices by integrating machine learning (ML) algorithms and blockchain technology. The Blockchain mechanism provides a immutable decentralized ledger that helps in storing various data and are linked together by cryptographic hashes. The utilization of blockchain in edge devices is to store the data that are sensitive to any intrusion that occur due to unauthorized access from an unauthenticated user. This paper examines the use of blockchain mechanism which are machine learning programs are used at various fields that helps in protecting and securing the data from any external modification or unauthorized access by the people outside the organization to enable integrity and confidentiality. This paper also discusses about the various algorithms that are used in the identification and classification of the data that is labelled under various categories that involves private sensitive data as-well-as typical data that doesn't need any high alert. Furthermore, it discusses the various methods that are used in integrating machine learning algorithms with blockchain.

Keywords *– Machine Learning, Blockchain, Edge devices, Federated Learning, Smart contract, Decentralized network,*

INTRODUCTION

The integration of machine learning (ML) algorithms and blockchain technology within edge devices heralds a profound shift for edge computing, offers comprehensive services for multifaceted hurdles inherent to this burgeoning field. In the forefront of this convergence is the imperative need for efficient data processing within resource-constrained environments characteristic of edge devices. By deploying lightweight ML models directly onto these devices, we unlock the potential for real-time processing capabilities, thereby mitigating latency concerns and minimizing reliance on centralized data processing infrastructures. Moreover, the utilization of smart contracts further augments this framework by automating data sharing agreements and access control policies, thereby facilitating secure and transparent data exchange while preserving user privacy and ownership rights. This amalgamation of ML algorithms and blockchain technology not only optimizes data processing efficiency and security but also fosters a newfound trust and resilience within edge computing ecosystems. As such, it represents a pivotal advancement towards realizing the full potential of edge computing across diverse applications and industries.

RELATED WORKS:

The convergence of machine learning (ML) algorithms with blockchain technology has sparked significant interest owing to its potential to tackle a myriad of issues concerning data privacy, security, and decentralized cooperation. While blockchain data mining, as elucidated in [1], provides valuable insights into the functioning of systems and participant behaviour, it encounters obstacles in managing extensive and intricate datasets. Conversely, federated learning (FL), as outlined in [3], facilitates collaborative model training while safeguarding data privacy, positioning it as a promising avenue for decentralized ML. Nevertheless, ensuring the privacy and security of federated learning systems remains a formidable challenge, as observed in [3].

In response to these challenges, researchers have put forth innovative frameworks like HealthFed, as delineated in [6], which harness the capabilities of both federated learning and blockchain technologies to enable privacy-preserving and distributed learning in healthcare. HealthFed fosters secure collaboration among multiple clinician collaborators while upholding data privacy through blockchain-driven mechanisms. Similarly, as discussed in [10], a blockchain-based federated learning approach is proposed for the Internet of Medical Things (IoMT), facilitating collaborative model training without centralized data storage, thus mitigating privacy concerns in healthcare applications.

Furthermore, blockchain technology offers avenues to fortify the security and credibility of federated learning systems. For instance, [12] advocates for a decentralized blockchain-based framework for federated learning, ensuring mutual trust among users during the exchange of local learning models. By leveraging the transparency and immutability of blockchain, this framework bolsters the security of federated learning systems, addressing apprehensions regarding the reliability of shared models.

Moreover, the amalgamation of blockchain and federated learning holds promise in tackling privacy and security concerns across diverse domains beyond healthcare. For instance, [8] presents AI-Bazaar, a blockchain-based computing-power trading framework, aimed at optimizing computing resource utilization and management in AI services. By integrating federated learning with blockchain, as discussed in [10], collaborative model training is facilitated while preserving data privacy in various applications of the Internet of Everything.

In essence, the integration of machine learning algorithms and blockchain technology offers

viable solutions to challenges pertaining to data privacy, security, and decentralized collaboration. Through the synergistic utilization of both technologies, pioneering frameworks and methodologies are being devised to address these challenges across a spectrum of domains, as evidenced by the research cited in [6] and [8].

SYSTEM ARCHITECTURE DIAGRAM

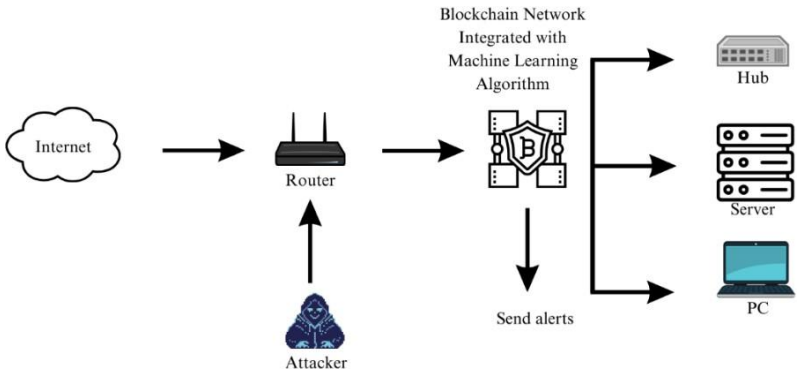

Fig 1: The architecture diagram that represent the integration of machine learning algorithms with the blockchain

The architecture diagram represent the usage of machine learning datasets that are used in the model training and traffic detection in the network that are entering from the internet which enters a private network. Upon finding a intrusion it sends alert to the user regarding the intrusion that is occurring.

MODULES:

1 MODULE: BLOCKCHAIN CREATION

This module focusses on creation of the blockchain that serves as a distributed ledger hold the dataset that is to be used in this work. These blocks are deployed at each node at the devices that are present in the network.

1.1 Block creation

This section involves the creation of simple blocks that are to be used as storage medium. Note that at this point the blocks are still mutable. The involvement of various algorithms make this a

irrevocable source of storage.

1.2 Consensus algorithms

The usage of consensus algorithm is done here. The integrity, security, and de-centralization of blockchain networks are all dependent on these algorithms.

```
# generates random string
def random_char(y):
    return ''. join(random. choice(string. ascii_letters) for x in range(y))


# this is method add transactions to the block on the fly make it more realistic
def add_transaction(block):

   global transactions_length
   global transactions

   for i in range(transactions_length):


    if(random.random() > 0.9):


        name = random_char(random.randint(0,20))
        file_name = random_char(random.randint(0,20))
        file_data = random_char(random.randint(0,200))

        t = {

            "user" : name,
            "v_file" : file_name,
            "file_data" : file_data,
            "file_size" : random.randint(0,1000)
        }

        block.add_t(t)
```

Fig 2: The code that is used in the backend of the creating consensus

```
C:\Windows\System32\cmd.e
4.966902600004687
------------Proof of Work with Random Nounce ------------
Difficulty  2  Time :  0.00159
Difficulty  3  Time :  0.03121
Difficulty  4  Time :  0.11252
Difficulty  5  Time :  3.22629
------------Proof of Work with Iterative Nounce ------------
Difficulty  2  Time :  0.0004
Difficulty  3  Time :  0.04379
Difficulty  4  Time :  0.01627
Difficulty  5  Time :  4.9669
------------Proof of Work with Random Nounce ------------
0.00159
0.03121
0.11252
3.22629
------------Proof of Work with Iterative Nounce ------------
0.0004
0.04379
0.01627
4.9669
```

Fig 3: Represent the differences in the consensus algorithms

2 MODULE: DATASET PREPARATION

In this module the dataset is created and the model is trained to detect the intrusion passing through the network. The dataset training involves various steps that are used to convert raw data into a data that is used to train a machine learning model. The following steps are used in the conversion of raw data into the pre-processed data. This data is then used to train the model.

2.1 Data Collection

This part involves the collection of data that is needed to be used for the model to be trained. The real time data is generally collected through the network sniffing tools like Wireshark or tcpdump in a pcap format. The that is used in this work is obtained from the Kaggle domain. The dataset used is CICIDS2017.

2.2 Data Cleaning

The obtained data is then cleaned by certain steps involves the removal of null values, duplicate values and the combining same data at different instances. This is done in-order to increase the robustness of the ML algorithm.

2.3 Data Sampling

The sampling of data is the process of removing the repeated data and quantizing it as single data. This reduces the number of unwanted data that is required to be present in the dataset.

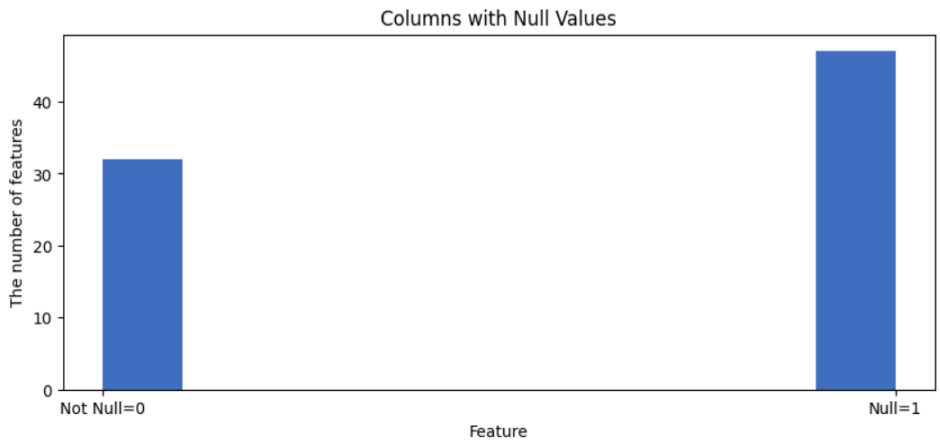

Fig 4.1: The number of null and non null values before the data preprocessing

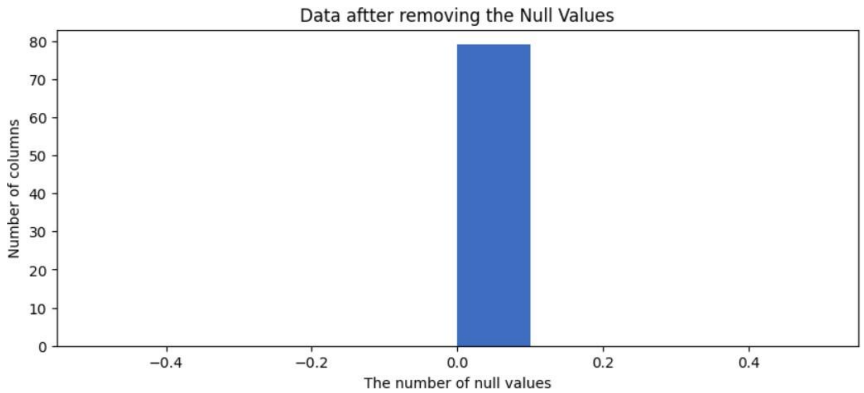

Fig 4.2 The number of null values that are present after the data preprocessing

2.4 Data Splicing

The next process involves the separation of data for training the ML model. The dataset is dataset is divided into training and test set in the ratio of 60:40. The data are labelled with the values that indicate the type of traffic.

```
# Split data into features and target variable
X = data_f.drop('Label', axis=1)
y = data_f['Label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

print("The train dataset size = ",X_train.shape)
print("The test dataset size = ",X_test.shape)
```

```
The train dataset size =  (46365, 78)
The test dataset size =  (19871, 78)
```

Fig 5: The dataset that is used is split into the training and test set

2.5 Model Training

The ML model is trained with the given training set. The training set has the network packet details that can be used to detect the attack pattern that is used to detect the traffic for any intrusions.

2.6 Accuracy checking

The model is later tested for the accuracy using the test set. This indicates the detection ability of the ML model. The higher the accuracy of the model indicates the efficiency and the false positives obtained during the execution.

```
# Evaluate Random Forest
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_precision = precision_score(y_test, rf_pred)

print('\nRandom Forest Metrics:')
print(f'Accuracy: {rf_accuracy:.4f}')
print(f'Precision: {rf_precision:.4f}')
```

```
Random Forest Metrics:
Accuracy: 0.9995
Precision: 1.0000
```

Fig 6: The accuracy and precision of the model used is obtained

3 MODULE: BLOCKCHAIN INTEGRATION

This module involves the creation of blockchain and hosting it in the local network to store the dataset that is used to train the ML model. The blockchain is integrated with the dataset to ensure the integrity of the dataset used.

3.1 Storage medium generation

The storage medium developed using the Python to store the dataset. The dataset is stored in this medium to be used to protect against any attack that occurs.

```
def mine(self):
    if(len(self.pending) > 0): #if there is atleast one pending transaction
        last_block = self.last_block() #get last block
        # Creates a new block to be added to the chain
        new_block = Block(last_block.index + 1,self.pending,last_block.hash)

        # runs the our proof of work and gets the consensus.
        hash1 = self.p_o_w(new_block)
        self.add_block(new_block, hash1)
        self.pending = []
        return new_block.index
    else:
        return False
```

Fig 7: The code used to mine a block for storing the data

3.2 Block hashing

The hash is created for each block to be used as a header for the consecutive blocks. This hash value is used as a connecting medium between the blocks to connect with each other. The hashing method used is SHA256.

```
# creates a hash for the block
def generate_hash(self):
    #  generates hash code using the values stored in block instance. completely random
    all_data_combined = str(self.index) + str(self.nonce) + self.prev_hash + str(self.transactions)
    return sha256(all_data_combined.encode()).hexdigest()
```

Fig 8: The generation of hash values

3.3 New block creation

The new block is created to constantly store live pre-processed data that is captured in the network. This allows the user to store the data in the blockchain for future model training to prevent the modification of data.

```
# Add a block to the chain after verfying and validating it.
def add_block(self, block, hashl):
    prev_hash = self.last_block().hash
    #check the validity of the block
    if (prev_hash == block.prev_hash and self.is_valid(block, hashl)):
        block.hash = hashl
        self.chain.append(block)
        return True
    else:
        return False
```

Fig 9: The module that allows the generation of new block

3.4 Front-end Development

The front-end for the blockchain is generated using the JavaScript. The front-end is used for the easy understandability of the functions.

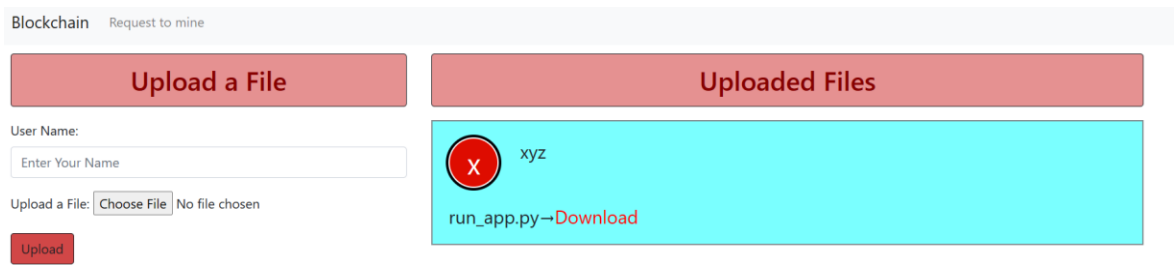

Fig 10: The dashboard displaying the blockchain file upload

4 MODULE: DEPLOYMENT

4.1 Network scan

The network scanning is done by using the tools imported in the python program using the tcpdump tool. This is done to scan network for the live traffic.

4.2 Trigger generation

This involves the generation of notification triggers that are used to alert the users. The trigger is generated when the model detect the incoming traffic.

```
def trigger_notification():
    title = "Alert"
    message = "Warning! Intrusion is detected."
    time.sleep(3)
    notification.notify(
        title=title,
        message=message,
        app_name="Alert"
    )
```

Fig 11: The trigger code used to generate a trigger

4.3 Front-end of scan

The front to start a scan is done using the Python. This triggers the scanning tool to initiate and sniff the packets in the network.

Fig 11: The front-end to start an network scan

4.3 Scan output

The output of the scan is shown in this part. The scan output describes the details of the packet that enters the network.

```
Frame 101: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: AzureWaveTec_fa:89:d0 (50:5a:65:fa:89:d0), Dst: IPv4mcast_7f:ff:fa (01:00
:5e:7f:ff:fa)
Internet Protocol Version 4, Src: 192.168.1.42, Dst: 239.255.255.250
Internet Group Management Protocol

No.     Time           Source                Destination           Protocol Length Info
102 74.383082      fe80::16a7:2bff:feea:fd6d ff02::1               ICMPv6   90     Multicas
t Listener Query
```

Fig 12: The scan output obtained after the start of the scan

4.4 Front-end of trigger

The trigger that is generated after the detection of any intrusion that enters the network.

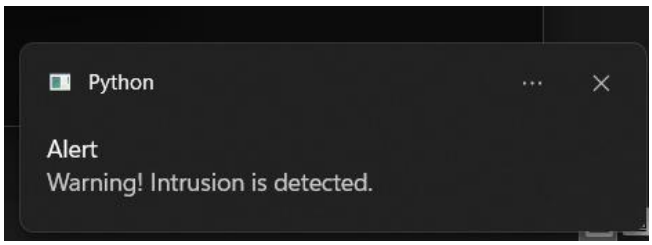

Fig 13: The trigger generated during the packet identification

CONCLUSION

The machine-learning (ML) algorithms and block-chain technologies within edge devices represents a compelling frontier with immense potential to redefine the landscape of edge computing. Through our exploration, it becomes evident that future enhancements in this domain hold the key to unlocking unprecedented capabilities and driving innovation across various facets of edge computing environments. The ongoing refinement and optimization of ML algorithms tailored for edge devices, alongside the infusion of advanced cryptographic techniques, promise to fortify security, enhance privacy, and instill trust in decentralized edge networks.

FUTURE ENHANCEMENT

On edge devices with constrained resources, it would be advantageous to focus on optimizing the blockchain and machine learning algorithm integration process. This optimization may include developing lightweight machine learning models tailored for edge computing environments and devising effective consensus techniques specifically for edge devices. Investigating novel approaches, such as secure multi-party computation or differential privacy, to enhance the security and privacy of data handled by edge devices may also be beneficial. If blockchain and machine learning solutions were to be successfully implemented on edge devices, it would be imperative to look into ways to address the scalability and interoperability problems that come up and ensure a seamless integration with the current infrastructure.

REFERENCES

[1] Yuxin Qi , JunWu , Senior Member, IEEE, Hansong Xu , Member, IEEE, and Mohsen Guizani , 'Blockchain Data Mining With Graph Learning' , IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 2, pp 729-748, February, 2024.

[2] Liwei Ouyang , Fei-Yue Wang , Fellow, IEEE, Yonglin Tian , Xiaofeng Jia, Hongwei Qi, and Ge Wang, 'Artificial Identification: A Novel Privacy Framework for Federated Learning Based on Blockchain', IEEE Transactions on Computational Social Systems, vol. 10, no. 6, pp 3576-3585, December, 2023.

[3] Qinbin Li , Zeyi Wen , Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li , Xu Liu, and Bingsheng He, 'A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection', IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 4, pp 3347-3366, April, 2023.

[4] Hao Guo , Member, IEEE, Collin Meese , Student Member, IEEE, Wanxin Li , Member, IEEE,Chien-Chung Shen , Member, IEEE, and Mark Nejad , Senior Member, IEEE, 'B2SFL: A Bi-Level Blockchained Architecture for Secure Federated Learning-Based Traffic Prediction', IEEE Transactions on Services Computing, vol. 16, no. 6, pp 4360-4374, November-December, 2023.

[5] Chao Lin , Xinyi Huang , Jianting Ning , and Debiao He , 'ACA: Anonymous, Confidential and Auditable Transaction Systems for Blockchain', IEEE Transactions on Dependable and Secure Computing, Vol. 20, No. 6, November/December 2023.

[6] Zakaria Abou El Houda , Member, IEEE, Abdelhakim Senhaji Hafid , Member, IEEE, Lyes Khoukhi , Senior Member, IEEE, and Bouziane Brik , Member, IEEE, 'When Collaborative Federated Learning Meets Blockchain to Preserve Privacy in Healthcare' IEEE Transactions on Network Science and Engineering, vol. 10, no. 5, pp 2445-24456, September-October, 2023.

[7] Aitizaz Ali , Muhammad Fermi Pasha , Antonio Guerrieri , Member, IEEE, Antonella Guzzo , Member, IEEE, Xiaobing Sun , Aamir Saeed, Amir Hussain, and Giancarlo Fortino , Fellow, IEEE, 'A Novel Homomorphic Encryption and Consortium Blockchain-Based Hybrid Deep Learning Model for Industrial Internet of Medical Things', IEEE Transactions on Network Science and Engineering, vol. 10, no. 5, pp 2402-2418, September-October,2023.

[8] Xiaoxu Ren, Chao Qiu ,Xiaofei Wang ,Zhu Han , Ke Xu, Haipeng Yao, and Song Guo, ‘ AI-Bazaar: A Cloud-Edge Computing Power Trading Framework for Ubiquitous AI

Services', IEEE Transactions on Cloud Computing, Vol. 11, No. 3,pp. 2337-2348, July-September 2023.

[9] Yingtao Ren , Xiaomin Zhu , Kaiyuan Bai , and Runtong Zhang, 'A New Random Forest Ensemble of Intuitionistic Fuzzy Decision Trees', IEEE Transactions on Fuzzy Systems, VOL. 31, No. 5,pp. 1729-1741, May 2023.

[10] Zhuotao Lian ,Graduate Student Member, IEEE, Qingkui Zeng ,Weizheng Wang ,Graduate Student Member, IEEE, Thippa Reddy Gadekallu , Senior Member, IEEE, and Chunhua Su, 'Blockchain-Based Two-Stage Federated Learning With Non-IID Data in IoMT System',IEEE Transactions on Computational Social Systems, vol. 10, no. 4, pp 1701-1710, August, 2023.

[11] Haoxiang Luo, Gang Sun, Hongfang Yu, Bo Lei, and Mohsen Guizani, 'An Energy-Efficient Wireless Blockchain Sharding Scheme for PBFT Consensus', IEEE Transactions on Network Science and Engineering, VOL. 26, No. 3, August, 2023.

[12] Ali Riahi , Amr Mohamed , Senior Member, IEEE, and Aiman Erbad , Senior Member, IEEE, 'RL-Based Federated Learning Framework Over Blockchain (RL-FL-BC)', IEEE Transactions on Network and Service management, vol. 20, NO. 2, pp 1587-1599, June, 2023 .

[13] Li Ruan , Yu Bai, Shaoning Li , Jiaxun Lv , Tianyuan Zhang, Limin Xiao , Haiguang Fang, Chunhao Wang, and Yunzhi Xue, 'Cloud Workload Turning Points Prediction via Cloud Feature-Enhanced Deep Learning', IEEE Transactions on Cloud Computing, VOL. 11, No. 2,pp. 1719-1732, April-June 2023.

[14] Aditya Pribadi Kalapaaking, Ibrahim Khalil, Mohammad Saidur Rahman, Mohammed Atiquzzaman , Senior Member, IEEE, Xun Yi, and Mahathir Almashor, 'Blockchain-Based Federated Learning With Secure Aggregation in Trusted Execution Environment for Internet-of-Things' IEEE Transactions on industrial informatics, vol. 19, NO. 2, pp 1703-1714, February, 2023.

[15] Dr.B.Chidhambararajan K.Elaiyaraja, Dr.M.Senthil Kumar ," Deep Learning-Based BDMSF Resource Sharing—A Systematic Approach for Analysis and Visualization "-Disruptive Technologies for Big Data and Cloud Applications, Springer, August 2022