International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Bridging the Security Gap: Preventing Reentrancy Attacks in Blockchain eVault Systems

Tarnam Goel[1], Sourav Singh [2], Anushka Singh[3], Mr. Yaduvir Singh[4]

[1]B.Tech CSE(AI) Noida Institute of Engineering and Technology Greater Noida, Uttar Pradesh tarnamgoel@gmail.com
[4] Assistant Professor, B.Tech CSE(AI) Noida Institute of Engineering and Technology Greater Noida, Uttar Pradesh yaduyash@gmail.com
[2]B.Tech CSE(AI) Noida Institute of Engineering and Technology Greater Noida, Uttar Pradesh ss96496636958@gmail.com
[3]B.Tech CSE(AI) Noida Institute of Engineering and Technology Greater Noida, Uttar Pradesh anushkasingh00567@gmail.com

ABSTRACT—

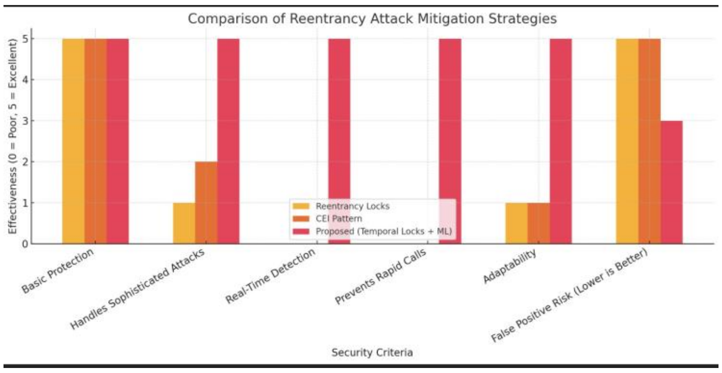
Blockchain platforms are susceptible to reentrancy assaults, which poses a risk to digital asset storage in eVaults. Before the contract changes its information, a hacker repeatedly invokes a function, such as a withdrawal, in this attack. This enables them to withdraw more money than they had planned. Although they help lower the risk, current solutions like reentrancy locks and the

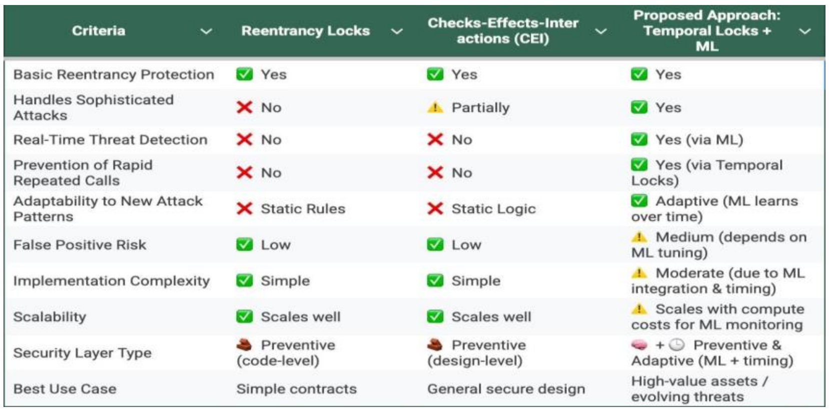
Checks-Effects-Interactions (CEI) pattern might not completely stop more sophisticated assaults. To improve security, we suggest integrating temporal locks with machine learning (ML) anomaly detection. In order to identify odd trends, like several quick calls, the machine learning system would track transactions in real time. By imposing a brief waiting interval between calls, the temporal locks will prevent more transactions if suspicious activity is identified.Through threat identification, this two-step method improves protection.

Keywords—Reentrancy Attack, Blockchain Security, eVaults, Smart Contract Vulnerability, Machine Learning (ML) Anomaly Detection, Temporal Locks, Reentrancy Guards

I. INTRODUCTION

Blockchain technology has altered the digital financial sector by increasing openness, security, and efficiency. However, the inadequacies inherent in smart contracts provide major issues that must be addressed. One of the most serious dangers is the **reentrancy attack**, which occurs when an attacker uses a smart contract's ability to call itself before the previous execution is complete. This might lead to unauthorized withdrawals, depleting cash and ultimately jeopardizing the contract's integrity [1][2].
Existing defensive techniques, such as
Checks-Effects-Interactions (CEI) patterns and reentrancy guards, provide some protection against these attacks by requiring state changes before external calls are performed [3][4]. However, these strategies frequently prove ineffective in the face of developing and increasingly sophisticated attack vectors. Attackers continuously adapt, rendering traditional security measures less effective.
To enhance security, we propose a hybrid security framework that includes machine learning (ML) anomaly detection and temporal locks in a blockchain-based eVault system. The ML component will examine transaction
patterns for abnormalities that may indicate an attempted attack, providing a proactive protection mechanism. [5]. Machine learning algorithms can identify abnormal activities that deviate from established criteria by analyzing previous transaction data.
Meanwhile, temporal locks will impose time-based access restrictions, allowing interactions with smart contracts only for certain durations. This method not only improves security but also reduces risk by limiting unwanted entrance outside of designated windows. [6]. The incorporation of these sophisticated technologies into the eVault architecture aims to create a safe environment for sensitive legal information while also improving transaction reliability within the blockchain ecosystem.
By combining these unique techniques, the proposed framework aims to provide a more robust defense for blockchain-enabled eVaults, addressing smart contract vulnerabilities while ensuring digital asset integrity and safety. The combination of machine learning, temporal locks, and blockchain technology presents a strong barrier to attacks, enabling a safe and trustworthy environment for legal and financial transactions [7]. [8].





Criteria	Reentrancy Locks	Checks-Effects-Interactions (CEI)	Proposed Approach: Temporal Locks + ML
Basic Reentrancy Protection	✅ Yes	✅ Yes	✅ Yes
Handles Sophisticated Attacks	❌ No	⚠️ Partially	✅ Yes
Real-Time Threat Detection	❌ No	❌ No	✅ Yes (via ML)
Prevention of Rapid Repeated Calls	❌ No	❌ No	✅ Yes (via Temporal Locks)
Adaptability to New Attack Patterns	❌ Static Rules	❌ Static Logic	✅ Adaptive (ML learns over time)
False Positive Risk	✅ Low	✅ Low	⚠️ Medium (depends on ML tuning)
Implementation Complexity	✅ Simple	✅ Simple	⚠️ Moderate (due to ML integration & timing)
Scalability	✅ Scales well	✅ Scales well	⚠️ Scales with compute costs for ML monitoring
Security Layer Type	🧱 Preventive (code-level)	🧱 Preventive (design-level)	🧠 + 🕒 Preventive & Adaptive (ML + timing)
Best Use Case	Simple contracts	General secure design	High-value assets / evolving threats

To better illustrate the comparative effectiveness of existing and proposed security measures against reintrusion attacks, the following diagram shows visual comparisons via key evaluation criteria with visual comparisons. This includes basic protection, the ability to carry out sophisticated attacks, real-time threat detection, preventing rapid repeat calls, adaptability to attack patterns that occur, and risk of false positives. In contrast, the proposed hybrid framework combines machine learning-based anomaly detection with time locking to demonstrate excellent performance in all critical areas. The ability to monitor real-time and adaptive learning allows him to actively recognize and mitigate the behavior of the abnormalities, and the temporary bulky mechanism serves as a buffer to stop rapid malicious calls. This integrated approach greatly improves the robustness of intelligent contractual security in blockchain-based Evault Systems.

II. LITERATURE REVIEW

G. A. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger" (2014)

Wood's research paper suggests the concept of decentralized ledger that enables smart contracts to set the groundwork for the future work and advancements of blockchain technology. This research paper explores Ethereum's architectural framework and tells how its unique design promotes security and transparency. Author also emphasizes on the need of strong consensus mechanism and the role of ethereum to enable Decentralized Applications, while solving the current security issues. It is critical to understand the context in which smart contract vulnerabilities emerge.

A. Christidis & M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things" (2016)

The study by Christidis and Devetsikiotis explores how blockchain and smart contract technology can be used with the Internet of Things (IoT) technology. According to there research paper, blockchain technology can be used to boost network confidence and protect IoT networks. While the focus is broader than smart contracts alone, it highlights the importance of security measures in the interrelated landscape of IoT, where security holes can have far-reaching consequences. The authors stress the need for robust security frameworks to deal with the unique challenges posed by this integration.

M. A. G. El-Mawardy, "Smart Contracts Security Issues and Solutions," Journal of Computer Virology and Hacking Techniques (2020)

A thorough assessment of security concerns (connected to smart contracts) was provided by El-Mawardy. This helped to find weakness other than reentrancy attacks. e.g: Gas Limit Issue, Timestamp reliance and Incorrect Access Control. The study gives us the common solutions and underlines the significance of rigorous testing and formal verification of the smart contract. Author has recommended to adopt the industry standards and best practices to prevent vulnerabilities. This will also encourage the developers to be aware about potential hazards in the quickly expanding blockchain technology.

H. Hu, Z. Huang, & Y. Zhang, "Anomaly Detection in Smart Contracts Using Machine Learning" (2020)

Hu et al. examined machine learning algorithms for detecting anomalies and improving smart contract security. The authors propose a system that leverages previous transaction data to identify trends and flag unusual activity that could indicate a security breach. This unique technology facilitates real-time monitoring and alarms, allowing for proactive risk management. The paper focuses on the evolving relationship between AI and blockchain technology, emphasizing the necessity for adaptive security methods in smart contracts.

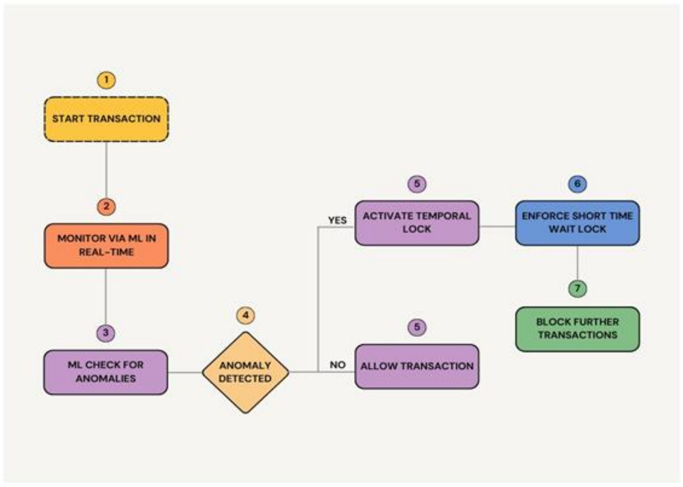
A. Khalid & B. S. A. M. Abdul-Sada, "A Hybrid Machine Learning Approach to Smart Contract Security" (2021)

A hybrid approach of making smart contracts safer was suggested by Khalid and Abdul-Sada. Author suggested to combine regular security protection with machine learning algorithms. The paper suggests a way of how this method could make it easier to find and response to threats in order to increase security. The authors say that this approach might make smart contracts more secure against different types of attacks, this would also make people more confident in blockchain apps.

6) P. R. L. Gomes et al., "Temporal Access Control in Smart Contracts" (2021)

A unique technique was suggested improve the smart contract security was provided by Gomes et al. They suggested the use of temporal access control mechanism to improve security. Author highlights how time-based constraints can prevent unauthorized access and minimize certain attacks, such as reentrancy attack. The paper suggests that allowing smart contracts to execute only within set time windows, we would get a possible path for improving overall security posture of blockchain systems.

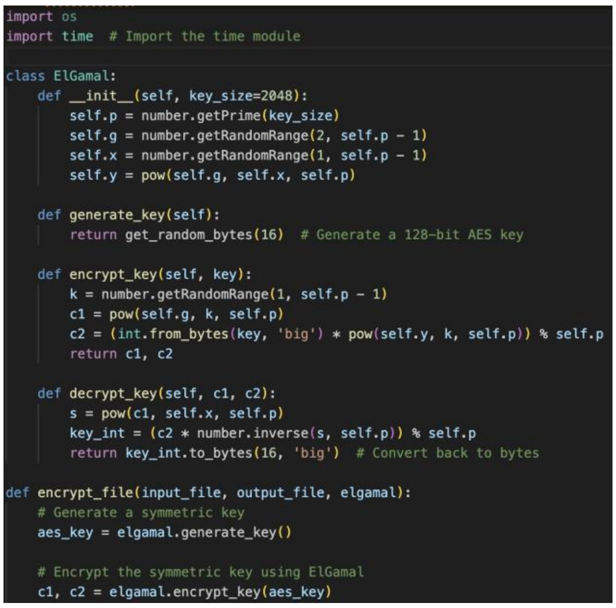
III. METHODOLOGY



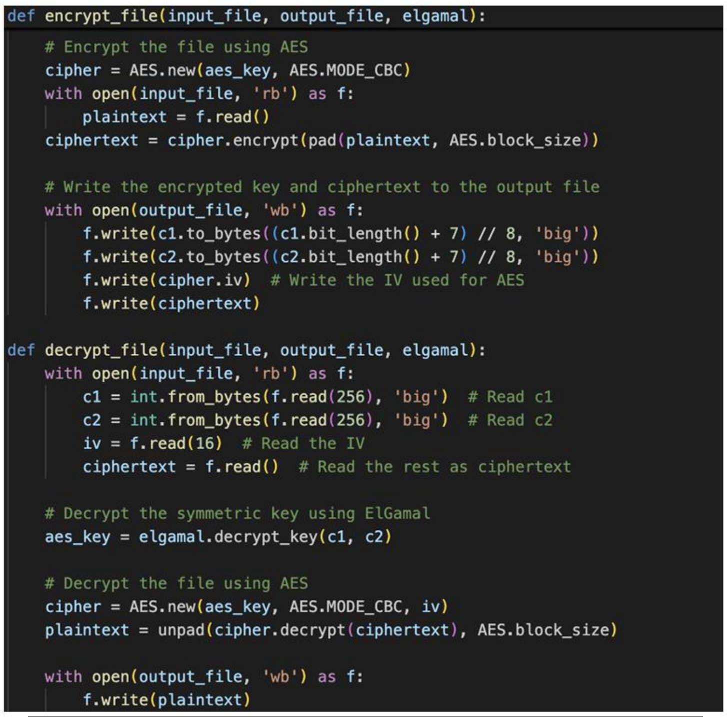
We propose a strategy of creating blockchain based eVault system. This system must include Machine Learning and Temporal Locks that focuses on improving the security, dependability and accessibility of the legal data. This method combines multiple technologies that are proven for security. The technologies include: Blockchain, Machine Learning and Temporal Locks Access Control Techniques to build a powerful product to handle all the transactions and sensitive information.

- **Blockchain Framework**: A permission based Blockchain (e.g: Hyperledger Fabric or Ethereum) will be used to manage the access and adaption. The Framework will be chosen based on the needs, security, privacy and speed [1][2].
- **Smart Contracts**: Smart Contracts will be made to implement access controls and manage user interactions with the eVault. These contracts will contain the logic of document storage, retrieval, security and implementation of temporal locks for time sensitive access [3][4].

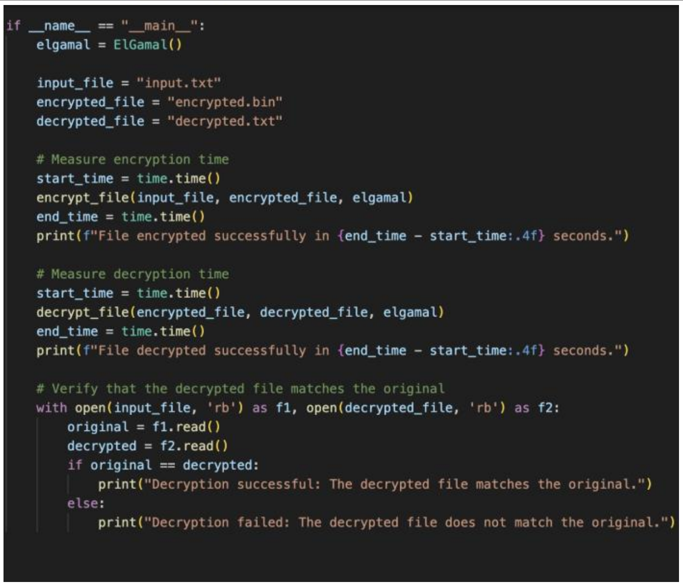
Image below shows the logic of temporal locks implementation and ML-based anomaly detection..



```python
import os
import time  # Import the time module

class ElGamal:
    def __init__(self, key_size=2048):
        self.p = number.getPrime(key_size)
        self.g = number.getRandomRange(2, self.p - 1)
        self.x = number.getRandomRange(1, self.p - 1)
        self.y = pow(self.g, self.x, self.p)

    def generate_key(self):
        return get_random_bytes(16)  # Generate a 128-bit AES key

    def encrypt_key(self, key):
        k = number.getRandomRange(1, self.p - 1)
        c1 = pow(self.g, k, self.p)
        c2 = (int.from_bytes(key, 'big') * pow(self.y, k, self.p)) % self.p
        return c1, c2

    def decrypt_key(self, c1, c2):
        s = pow(c1, self.x, self.p)
        key_int = (c2 * number.inverse(s, self.p)) % self.p
        return key_int.to_bytes(16, 'big')  # Convert back to bytes

def encrypt_file(input_file, output_file, elgamal):
    # Generate a symmetric key
    aes_key = elgamal.generate_key()

    # Encrypt the symmetric key using ElGamal
    c1, c2 = elgamal.encrypt_key(aes_key)
```



```
def encrypt_file(input_file, output_file, elgamal):
    # Encrypt the file using AES
    cipher = AES.new(aes_key, AES.MODE_CBC)
    with open(input_file, 'rb') as f:
        plaintext = f.read()
    ciphertext = cipher.encrypt(pad(plaintext, AES.block_size))

    # Write the encrypted key and ciphertext to the output file
    with open(output_file, 'wb') as f:
        f.write(c1.to_bytes((c1.bit_length() + 7) // 8, 'big'))
        f.write(c2.to_bytes((c2.bit_length() + 7) // 8, 'big'))
        f.write(cipher.iv)  # Write the IV used for AES
        f.write(ciphertext)

def decrypt_file(input_file, output_file, elgamal):
    with open(input_file, 'rb') as f:
        c1 = int.from_bytes(f.read(256), 'big')  # Read c1
        c2 = int.from_bytes(f.read(256), 'big')  # Read c2
        iv = f.read(16)  # Read the IV
        ciphertext = f.read()  # Read the rest as ciphertext

    # Decrypt the symmetric key using ElGamal
    aes_key = elgamal.decrypt_key(c1, c2)

    # Decrypt the file using AES
    cipher = AES.new(aes_key, AES.MODE_CBC, iv)
    plaintext = unpad(cipher.decrypt(ciphertext), AES.block_size)

    with open(output_file, 'wb') as f:
        f.write(plaintext)
```



```
if __name__ == "__main__":
    elgamal = ElGamal()

    input_file = "input.txt"
    encrypted_file = "encrypted.bin"
    decrypted_file = "decrypted.txt"

    # Measure encryption time
    start_time = time.time()
    encrypt_file(input_file, encrypted_file, elgamal)
    end_time = time.time()
    print(f"File encrypted successfully in {end_time - start_time:.4f} seconds.")

    # Measure decryption time
    start_time = time.time()
    decrypt_file(encrypted_file, decrypted_file, elgamal)
    end_time = time.time()
    print(f"File decrypted successfully in {end_time - start_time:.4f} seconds.")

    # Verify that the decrypted file matches the original
    with open(input_file, 'rb') as f1, open(decrypted_file, 'rb') as f2:
        original = f1.read()
        decrypted = f2.read()
        if original == decrypted:
            print("Decryption successful: The decrypted file matches the original.")
        else:
            print("Decryption failed: The decrypted file does not match the original.")
```

2. Temporal Access Control Mechanism

- **Definition of Temporal Locks**: Temporal locks are used to limit the access to the data or document based on time parameters. These also include defining of access based roles within the window during which user can interact with the eVault. e.g: Lawyers and Advocates may be permitted to access specific papers for a short time only. This can be done by implementing time parameter in Temporal Locks [5].
- **Smart Contract Integration**: Temporal Locks logic is integrated with smart contracts to ensure that the document access request must be evaluated against the time parameters (made in temporal locks). This would help in minimizing illegal access and the danger of data breaches [6].

3. Machine Learning Integration

- **Anomaly Detection**: Using machine learning methods we will track the user activity and detect the anomalies. This might involve employing supervised learning algorithms to recognize frequent and suspicious activities and potential threats in real time [7]. [8]
- **Training Data**: Collect historical information on document access trends, including both successful and unsuccessful attempts. This information will be used to train machine learning algorithms to recognize legitimate usage and identify anomalies that may indicate security threats. [9].
- **Adaptive Learning**: Regularly update ML models with new patterns and data to provide resilience to future attack vectors [10].

4. Implementation Phases

- **Prototype Development**: Build an eVault prototype that includes the blockchain infrastructure, smart contracts, and basic ML models. This phase will concentrate on functionality, such as document uploading, retrieval, and access tracking [11].
- **Testing and Validation**: Conduct extensive testing of the eVault system, including penetration testing, to detect weaknesses. Validate the temporal locks and ML anomaly detection techniques' efficacy using simulated assaults and real-world scenarios [12].
- **User Feedback and Iteration**: Engage potential users (legal professionals, clients, etc.) and solicit input on the system's usability and security. Iterate on design and functionality in response to customer feedback to improve overall effectiveness [13].

5. Deployment and Monitoring

- **Deployment**: Once verified, install the eVault on a secure server with all essential compliance safeguards in place. Ensure that all legal and regulatory requirements are satisfied, notably data protection legislation [14].
- **Continuous Monitoring**: Set up a monitoring system to oversee eVault access, using the ML model for real-time anomaly detection and automatic notifications for suspicious activity. This helps protect the integrity and security of legal records over time [15] [16].

III. PROPOSED APPROACH

1. MACHINE LEARNING-BASED ANOMALY DETECTION

This module focuses on detecting anomalous transaction behaviors in real time on the blockchain.

- **Training Model**: The machine learning model is created by studying previous transaction data from the blockchain, allowing it to learn and recognize legal transaction patteThe machine learning model is created by studying previous transaction data
  from the blockchain, allowing it to learn and recognize legal transaction patterns [6]. [7]..
- **Deployment**: The machine learning model works on-chain with a lightweight architecture to reduce transaction delays and gas costs that results in the efficient performance in the blockchain context [10] [11].

2. Temporal Locks on Smart Contracts

To improve security, temporal locks provide a purposeful waiting interval between subsequent function calls.

- **Lock Mechanism**: Once a function is performed, it enters a lock state for a defined amount of time, prohibiting any future calls [12].
- **Mitigating Recursive Exploits**: This feature successfully restricts attackers from performing additional function calls during the lock period, thereby avoiding recursive reentrancy attacks that might undermine the system's integrity [13] [14].
- **User Impact**: Although temporal locks cause a minor delay in transaction processing, they considerably improve security without compromising user experience [15].

3. Integration with Blockchain-based eVaults

The security components are simply incorporated into the eVault's blockchain architecture.

eVault Architecture: The eVault smart contracts combine both the ML anomaly detection system and the temporal lock approach [16].

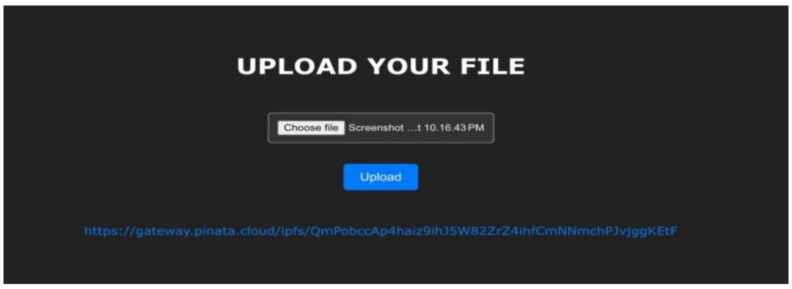
Execution Flow:

A. A user initiates an eVault transaction.
B. The ML model examines the transaction in real time for irregularities.
C. If the transaction is marked as normal, it will proceed while activating temporal locks to avoid repetitive calls.
D. If the transaction is considered suspicious, it is paused, and an alert is produced for further investigation [17]. [18].

IV. IMPLEMENTATION

To show the eVault's capability, we incorporate a basic IPFS-based file storage system. A step-by-step overview of the implementation is provided below, with two screenshots for reference.

1. Uploading Files on eVault (Frontend)

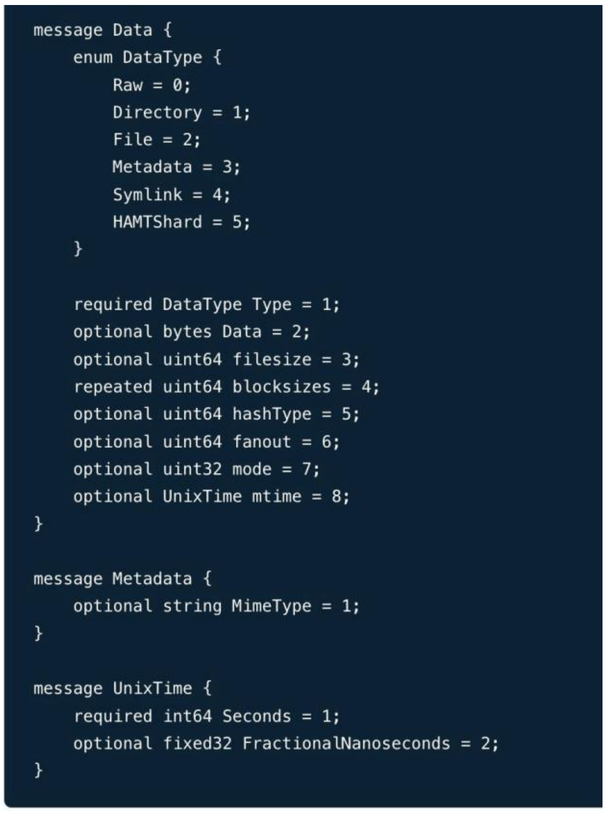


https://gateway.pinata.cloud/ipfs/QmPobccAp4haiz9ihJ5W82ZrZ4ihfCmNNmchPJvjggKEtF

The first screenshot shows the file upload interface. Users pick a file from their local system and upload it to IPFS (InterPlanetary File System) using Pinata, a gateway service. This enables a decentralized mechanism to store and access data, which is consistent with eVault's objective of secure blockchain-based storage.

- **Process**:
  - **Frontend**: A simple file input interface was created using React.js or a similar framework.
  - **Upload Mechanism**: When a user selects "Upload," the file is sent to the Pinata IPFS gateway. When the file is successfully uploaded, a unique hash (CID) is generated that corresponds to its IPFS address.
  - **Link Generation**: The CID is used to generate the IPFS link, which is shown on the interface (as seen in the image), allowing users to retrieve their files at any time.

2. File Metadata and Structure (Backend)



```
message Data {
    enum DataType {
        Raw = 0;
        Directory = 1;
        File = 2;
        Metadata = 3;
        Symlink = 4;
        HAMTShard = 5;
    }

    required DataType Type = 1;
    optional bytes Data = 2;
    optional uint64 filesize = 3;
    repeated uint64 blocksizes = 4;
    optional uint64 hashType = 5;
    optional uint64 fanout = 6;
    optional uint32 mode = 7;
    optional UnixTime mtime = 8;
}

message Metadata {
    optional string MimeType = 1;
}

message UnixTime {
    required int64 Seconds = 1;
    optional fixed32 FractionalNanoseconds = 2;
}
```

The second graphic shows the **Protobuf file structure**, which specifies the format of the stored data. This structure ensures that all files and their metadata are securely maintained.

- **Components**:
  - **Data Type**: specifies whether the uploaded item is a raw file, directory, metadata, or symlink.
  - **Metadata**: Stores information such as MIME type for simple retrieval and identification.
  - **Temporal Lock Configuration**: Implemented using timestamps (UnixTime), which specify the time period during which data is locked or limited.
  - **Transaction Security**: If any abnormalities are discovered during upload (using the ML module), the transaction is

flagged or suspended. **3. Anomaly Detection Using Machine Learning** Once a user initiates a file upload:

- **Training & Deployment**: A pre-trained machine learning algorithm examines transactions in real time to ensure that only valid uploads occur. This model examines common transaction patterns to detect suspicious activity (such as unexpected data volumes or frequent downloads).
- **Action on Suspicious Activity**: If a transaction is flagged, the system suspends further activities until it is manually reviewed or approved.

4. Temporal Locks Implementation

As soon we upload the data, temporal locks are activated and now it can be used for the protection of the system against unauthorized access and other attacks. If any attacker tries to access the data, then it gets blocked by Temporal Locks.

- **Smart Contract Code**:
  - Implements time-based constraints.
  - The blockchain records the latest transaction time.
  - Ensures that the next transaction for the same file or address should only occur once the set cooldown period has expired.

5. Integrated Flow of eVault

- **Transaction Execution**:
  1. User will visit the website and upload there files.
  2. The transaction will be analyzed by the system using Machine Learning Anomaly Detection System.
  3. If the uploaded file is authentic then the file's CID will be produced and temporal locks are enabled.
  4. In other case, if the transaction appears to be suspect, then the system halts it and shows a warning.

This eVault is implemented using Blockchain, so that it enhances security by adding decentralized storage (IPFS), machine learning and temporal locks. This results in a strong solution for safe digital asset management.

V. CONCLUSION

The proposed strategy to improve security and dependability of Blockchain Based eVaults using Machine Learning and Temporal Locks offers significant improvement in the management of sensitive data and transactions. The real time anomalies can be detected and highlighted using a machine learning model based on past transaction records. This also adverts future security breaches. Use of the temporal locks introduces an important delay between function calls, effectively limiting the risks of reentrancy attacks and improving overall transaction security.
These unique technologies are integrated into eVault architecture to not only improve legal document management but also to boost user trust in the blockchain system. It is important to use such comprehensive and adaptive security solutions to secure the reliability and privacy of sensitive data. This technology will make path for more secure and efficient blockchain applications in the future. This will also ensure that wide range of industries and large companies can provide blockchain applications without worrying about the attacks, ensuring seamless user experience.

We are ready to make a change in current Blockchain based systems to bring more secure system, by combining machine learning and temporal locks.

ACKNOWLEDGMENT

We want to express our heartfelt gratitude to everyone who has helped us to make this attempt a success. We thank our mentors for their valuable advice and guidance, as it helped us to define the direction of our study. We also respect the participation and inputs of our colleagues and peers, whose feedbacks have helped us to broaden our work. Special thanks is due to the researchers whose study helped us, including Siegel, El-Mawardy, Wood, Atzei, Hu, Khalid, Gomes, and Christidis, whose contributions provided us insights and a solid theoretical underpinning for our technique. Finally, we want to thank our family and friends for their unwavering support and encouragement throughout our journey, which motivated us to push the limits of our studies and strive for perfection.

REFERENCES

[1] D. Siegel, "Smart Contract Security: Reentrancy Attacks and How to Prevent Them," 2019.
[2] M. A. G. El-Mawardy, "Smart Contracts Security Issues and Solutions," Journal of Computer Virology and Hacking Techniques, 2020.
[3] G. A. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," 2014.
[4] N. Atzei, M. Bartoletti, & T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts," 2017.
[5] H. Hu, Z. Huang, & Y. Zhang, "Anomaly Detection in Smart Contracts Using Machine Learning," 2020.
[6] I. A. Khalid & B. S. A. M. Abdul-Sada, "A Hybrid Machine Learning Approach to Smart Contract Security," 2021.
[7] P. R. L. D. S. M. Gomes et al., "Temporal Access Control in Smart Contracts," 2021.
[8] A. Christidis & M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," 2016.
[9] K. H. Lim, "Machine Learning for Cybersecurity: A Review," IEEE Access, 2019.
[10] R. Choudhury et al., "Real-Time Anomaly Detection in Blockchain Networks," Future Generation Computer Systems, 2021.
[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
[12] L. M. V. Nascimento et al., "Evaluating Security and Performance of Smart Contracts: A Systematic Literature Review," Future Generation Computer Systems, 2020.
[13] J. Zhang, "The Role of User Feedback in Enhancing Security Protocols," Journal of Information Security, 2020.
[14] European Union General Data Protection Regulation (GDPR), 2018.
[15] Y. A. Ahmed et al., "Security and Privacy in Blockchain-Based Systems: A Survey," IEEE Transactions on Information Forensics and Security, 2021.
[16] D. M. S. S. Ferreira et al., "Blockchain-Based Solutions for Secure Data Management," Future Generation Computer Systems, 2021.