American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-14, Issue-4, pp-19-25
www.ajer.org
Research Paper Open Access

# Blockchain Security and the Role of Artificial Intelligence In Enhancing It

Nikoloz Katsitadze, Gia Surguladze, Aleksi Kobaidze, Tornike Japaridze, Mariam Chkhaidze, Zurab Bosikashvili
*Georgian Technical University, Tbilisi, Georgia*
*Corresponding Author: Nikoloz Katsitadze*

***ABSTRACT :*** *Blockchain technology has emerged as a transformative solution for secure, decentralized data management. However, despite its cryptographic foundations, blockchain networks remain vulnerable to various security threats, including 51% attacks, smart contract exploits, Sybil attacks, and phishing schemes. Traditional security measures often struggle to detect and mitigate these threats in real time. This paper explores the potential of Artificial Intelligence (AI) as a powerful tool for enhancing blockchain security.*
*We evaluate the security challenges faced by blockchain networks and assess existing risk mitigation strategies. Furthermore, we introduce AI-driven mechanisms for proactive threat detection, anomaly recognition, and automated security auditing of smart contracts. By integrating machine learning models and predictive analytics, AI can identify suspicious activities, detect fraudulent transactions, and enhance overall network resilience.*
*Our study highlights how AI-powered solutions can provide real-time security enhancements, reducing the risks associated with decentralized systems. By leveraging AI, blockchain platforms can move towards a more secure and adaptive security framework, ensuring improved trust and reliability in Web3 applications and decentralized finance (DeFi) ecosystems.*

***KEYWORDS*** *Blockchain Security, Artificial Intelligence, Predictive Analytics, Web3 Security, Machine Learnin.*
---------------------------------------------------------------------------------------------------------------------------------------
Date of Submission: 02-04-2025 Date of acceptance: 12-04-2025
---------------------------------------------------------------------------------------------------------------------------------------

## I. INTRODUCTION

Blockchain technology has rapidly become one of the most promising solutions for secure and decentralized data management. However, despite its cryptographic foundations, blockchain networks continue to face significant security challenges. Some of the most common threats today include 51% attacks, smart contract exploits, Sybil attacks, phishing schemes, and other malicious activities, all of which pose serious risks to decentralized finance (DeFi) systems and the broader Web3 ecosystem.

Traditional security mechanisms often struggle to detect and prevent these threats in real time. While decentralization is one of blockchain's greatest strengths, it also introduces vulnerabilities that make identifying and responding to attacks more complex. Hackers leverage sophisticated attack methods, exploit smart contract vulnerabilities, and take advantage of the network's anonymity, creating critical security issues that demand innovative solutions.

One of the most promising approaches to addressing these challenges is the integration of Artificial Intelligence (AI) and Machine Learning (ML) technologies into blockchain security strategies. AI-driven solutions can detect network anomalies, identify fraudulent transactions, and automate smart contract security auditing. These capabilities not only reduce the risks of security breaches but also provide proactive protection for blockchain infrastructure.

This paper explores the key security challenges facing blockchain networks, evaluates existing defense mechanisms, and examines the potential of AI-enhanced security models. We focus on how AI can strengthen blockchain security and improve the resilience of decentralized platforms in the face of evolving cyber threats.

## II. SIGNIFICANCE OF THE RESEARCH

Blockchain technology is increasingly being adopted across various industries, yet ensuring its security remains a major challenge. Traditional security mechanisms often struggle to keep up with evolving threats, making it essential to explore new approaches. This research is important because it examines how Artificial Intelligence (AI) can enhance blockchain security by enabling proactive monitoring, anomaly detection, and automated auditing. By integrating AI-driven solutions, blockchain networks can become more resilient, adaptive, and trustworthy. The findings of this study can contribute to the development of more robust security frameworks, ensuring the long-term stability and reliability of decentralized systems.

## III. 51% ATTACK: COST AND IMPACT

A 51% attack (Majority Attack) is one of the most critical threats to blockchain networks. In this attack, a malicious entity gains control of more than 50% of the network's hashing power, allowing them to: Perform double spending (reusing the same coins in multiple transactions); Censor or reverse transactions; Monopolize block creation, preventing other miners from adding blocks.

The cost of a 51% attack depends on factors such as the network's total hashrate, electricity costs, and mining hardware efficiency. The general formula for estimating the attack cost is:

$$C = T \times H \times E \times P$$

**Fig. 1. Computational Cost Formula for Blockchain Majority Attack**

Where:
- C = Total attack cost (USD)
- T = Duration of the attack (in hours)
- H = Total network hashrate (TH/s)
- E = Energy consumption per unit of hashrate (Watts per TH/s)
- P = Electricity price (USD per kWh)

In large blockchain networks like Bitcoin, a 51% attack becomes practically infeasible and economically meaningless. The immense hashrate of Bitcoin makes it nearly impossible for any single entity to control most of the mining power. Even if an attacker were able to amass enough computational resources, the cost of sustaining such an attack would be astronomical, often running into billions of dollars per hour due to the sheer energy consumption required. Moreover, the long-term consequences of a successful attack would be detrimental to the attacker as well. By undermining trust in the network, the value of Bitcoin would likely plummet, devaluing any coins the attacker holds or acquires through double spending. Additionally, the decentralized nature of Bitcoin allows the community to implement countermeasures, such as protocol updates or a hard fork, to neutralize the attack's impact. As a result, while theoretically possible, a 51% attack on a large blockchain network is neither practical nor financially viable.

## IV. THE RISING THREAT OF SMART CONTRACT EXPLOITS

A smart contract exploit refers to the act of taking advantage of security vulnerabilities or logical flaws in a smart contract to manipulate its behavior for malicious gain. These exploits often arise due to coding errors, unverified external calls, or inadequate access controls, making them a significant threat in decentralized applications, especially in the DeFi sector. One of the most infamous types of smart contract exploits is the reentrancy attack, where a contract makes an external call to another contract before updating its own state, allowing an attacker to repeatedly withdraw funds before the balance is properly adjusted. Another common exploit is integer overflow or underflow, where incorrect arithmetic operations lead to unintended values, potentially enabling unauthorized withdrawals or bypassing restrictions. Additionally, unchecked external calls can allow attackers to manipulate contract logic by passing malicious inputs, while front-running attacks involve exploiting the visibility of pending transactions to execute trades at a more favorable price before legitimate transactions are processed. Due to the immutable nature of blockchain, once a smart contract is deployed, vulnerabilities cannot be easily fixed, making security audits and formal verification crucial in preventing exploits. Without proper safeguards, smart contract vulnerabilities can lead to multimillion-dollar losses, eroding trust in blockchain-based financial systems.

A promising approach would be to use machine learning models trained on historical smart contract transactions and known exploits. By analyzing patterns in contract code, transaction flows, and interactions, AI can help identify vulnerabilities early on.

For example, employing deep learning techniques such as graph neural networks could be particularly effective. Smart contracts and their interactions can be represented as graphs, where nodes correspond to functions or external calls, and edges represent data flows or control flows. Graph-based AI models can learn to detect unusual patterns, such as repetitive calls indicative of reentrancy attacks or unexpected parameter values that may signal an integer overflow vulnerability.

Another valuable method is the integration of natural language processing (NLP) to review smart contract documentation, comments, and commit messages. By analyzing the language used in codebases and developer notes, AI could uncover hints of poor security practices, thus complementing traditional code audits.

In summary, leveraging graph-based deep learning for anomaly detection and applying NLP to project documentation are two strong choices for using AI to enhance smart contract security.

## V. SYBIL ATTACKS IN BLOCKCHAIN

Sybil Attacks occur when a malicious entity creates multiple fake identities to gain a disproportionate influence on a network. This kind of attack is named after the Sybil effect, which describes the creation of numerous identities by a single actor. It is particularly relevant in decentralized systems like blockchain networks, where consensus and decision-making often rely on participants' identities and reputations.

In a Sybil attack, an attacker generates many pseudonymous identities, gaining more influence over the network than any legitimate user. This can impact consensus mechanisms, voting systems, or reputation-based systems. For example, in proof-of-work (PoW) or proof-of-stake (PoS) systems, a Sybil attacker could manipulate the validation process by controlling a large portion of the network’s nodes or stakes.

The cost of executing a Sybil attack is determined by various factors, including the network’s computational power (in the case of PoW) or the stake required (in PoS systems). The formula for estimating the cost can be as follows:

$$C = N \times I \times A$$

**Fig. 2. Sybil Attack Cost Estimation Formula**

Where:

- C = Total cost of the Sybil attack (USD or equivalent)
- N = Number of fake identities or nodes to be created
- I = Cost to create each fake identity (USD or cost per node)
- A = Average cost of maintaining and running each fake identity (electricity, hardware, etc.)

In a PoW system like Bitcoin, this cost would involve hardware acquisition and energy consumption, while in a PoS system, it would involve buying or controlling a significant percentage of the total stake to manipulate consensus.

Sybil attacks are a critical concern for decentralized networks, and designing systems resistant to them is an ongoing challenge in blockchain and distributed ledger technology.

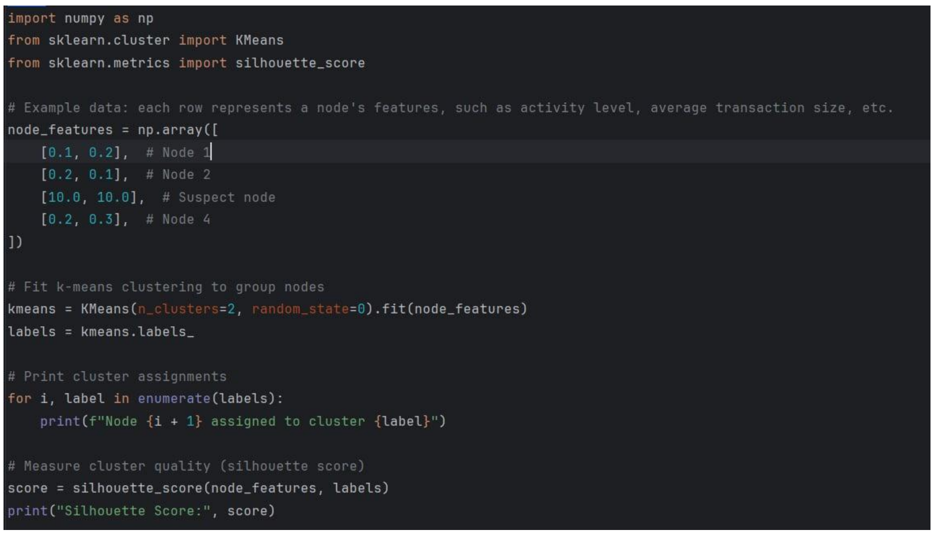
One effective design for mitigating Sybil attacks involves using reputation-based AI models coupled with robust network behavior analysis. For instance, a machine learning system could be trained to analyze the interaction patterns of nodes over time, looking for signs of coordinated activity or statistical anomalies that deviate from normal network behavior.

Specifically, a graph-based neural network might be employed to model the relationships between nodes, allowing it to detect clusters of low-reputation or short-lived nodes that might indicate a Sybil attack. Such an approach can dynamically adjust reputation scores, deprioritize or isolate suspect nodes, and reduce their influence on the network.

Additionally, incorporating temporal AI models, such as recurrent neural networks or transformers, could help track and predict patterns of activity that don’t align with legitimate node behavior. This proactive design, combining both relational and temporal AI models, can serve as a strong defense against Sybil attacks, improving the resilience of the entire blockchain ecosystem.

Below are some examples of Python code snippets that demonstrate AI-based detection mechanisms for Sybil attacks. These examples are simplified to illustrate concepts rather than provide production-ready solutions.

In the process of anomaly detection, clustering algorithms can be used**.** One way to identify potential Sybil nodes is to group nodes based on their behavior and look for outliers. Here’s a basic example using k-means clustering with the scikit-learn library:

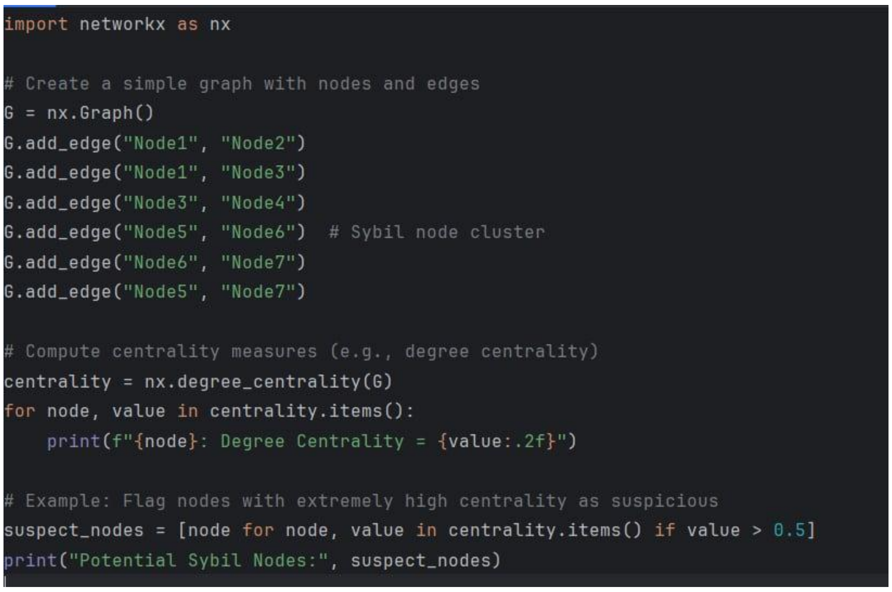


```python
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Example data: each row represents a node's features, such as activity level, average transaction size, etc.
node_features = np.array([
    [0.1, 0.2],  # Node 1
    [0.2, 0.1],  # Node 2
    [10.0, 10.0],  # Suspect node
    [0.2, 0.3],  # Node 4
])

# Fit k-means clustering to group nodes
kmeans = KMeans(n_clusters=2, random_state=0).fit(node_features)
labels = kmeans.labels_

# Print cluster assignments
for i, label in enumerate(labels):
    print(f"Node {i + 1} assigned to cluster {label}")

# Measure cluster quality (silhouette score)
score = silhouette_score(node_features, labels)
print("Silhouette Score:", score)
```

**Fig. 3. Anomaly detection example on python**

In this example (on Fig. 3), the suspect node with significantly different features might be flagged as a potential Sybil node based on its cluster assignment.

Graph-based node classification can be used in this process**.** If the blockchain network is represented as a graph, you can use libraries like networkx on the pythont platform to build the graph and then apply custom logic or integrate with AI frameworks for node



```python
import networkx as nx

# Create a simple graph with nodes and edges
G = nx.Graph()
G.add_edge("Node1", "Node2")
G.add_edge("Node1", "Node3")
G.add_edge("Node3", "Node4")
G.add_edge("Node5", "Node6")  # Sybil node cluster
G.add_edge("Node6", "Node7")
G.add_edge("Node5", "Node7")

# Compute centrality measures (e.g., degree centrality)
centrality = nx.degree_centrality(G)
for node, value in centrality.items():
    print(f"{node}: Degree Centrality = {value:.2f}")

# Example: Flag nodes with extremely high centrality as suspicious
suspect_nodes = [node for node, value in centrality.items() if value > 0.5]
print("Potential Sybil Nodes:", suspect_nodes)
```

**Fig. 4. Graph-based node classification on python**

This approach doesn’t require training a machine learning model but relies on graph metrics to spot unusual structures or clusters indicative of a Sybil attack. For more advanced AI integration, you could use a library like PyTorch Geometric to train graph neural networks that classify nodes as legitimate or Sybil based on their embeddings and relationships.

## VI. PHISHING SCHEMES IN BLOCKCHAIN: UNDERSTANDING AND PREVENTION

Phishing schemes on blockchain networks involve fraudulent attempts to steal sensitive user information, such as private keys, login credentials, or recovery phrases, through deceptive tactics. In blockchain systems, phishing usually targets users interacting with cryptocurrency wallets, decentralized applications (dApps), or exchanges. Attackers create fake websites, emails, or social media accounts that appear legitimate, tricking users into providing their personal information.

For example, attackers create fake versions of popular wallet websites, such as MetaMask or Trust Wallet, and promote them through misleading ads, emails, or social media. Users who enter their private keys or recovery phrases on these sites unknowingly give attackers full access to their wallets. Similarly, phishing emails or messages are sent claiming to be from legitimate services like exchanges, wallets, or dApps, asking users to "verify" their account or reset their password. The link in these messages leads the user to a fraudulent website, where they enter their credentials, thus compromising their account information.

Another common scam involves fake tokens or airdrops, where attackers prompt users to connect their wallets to malicious smart contracts. Once users approve a transaction or sign a smart contract, attackers can steal assets from the wallet or use the signature to drain funds. Phishers may also impersonate well-known figures, projects, or companies on social media platforms like Twitter or Telegram, promising giveaways, rewards, or exclusive opportunities. They often create fake accounts that look legitimate, providing links to phishing websites or offering fake tokens that require users to enter their personal details.

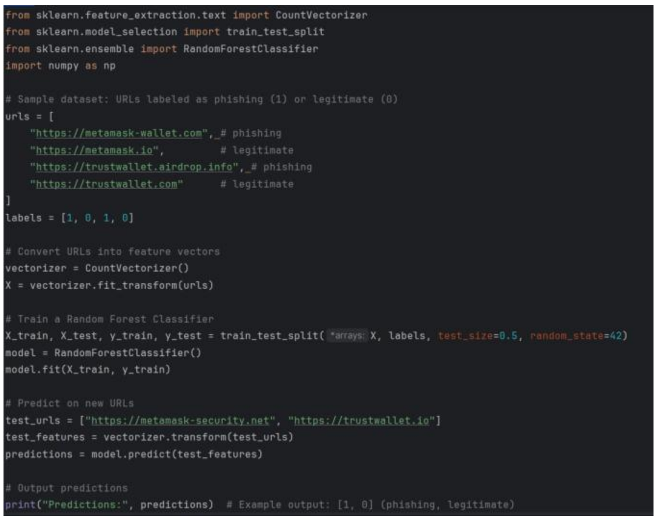
Phishing can also occur through malicious smart contracts designed to look like legitimate ones. When users interact with these contracts, they may unknowingly approve transactions that send funds to the attacker's address.

To prevent falling victim to phishing schemes, users should always verify the legitimacy of websites and dApps before interacting with them. Avoiding suspicious links in emails, messages, or social media and using two-factor authentication (2FA) wherever possible can help reduce the risk. Educating users on phishing tactics and encouraging the use of hardware wallets for storing cryptocurrency can also significantly enhance security.

We are offering AI model ideas for detecting blockchain phishing attempts:

URL and Domain Classifier: A supervised learning model that classifies websites as legitimate or phishing based on features such as URL structure, domain age, SSL certificate status, and known phishing indicators. Example approach:

- Train a logistic regression or gradient boosting classifier using a dataset of phishing and non-phishing blockchain-related URLs.
- Extract features like the length of the URL, presence of certain keywords (e.g., “metamask” or “airdrop” in unexpected ways), and DNS registration details.
- Use the model to flag suspicious links shared via email, social media, or message boards.

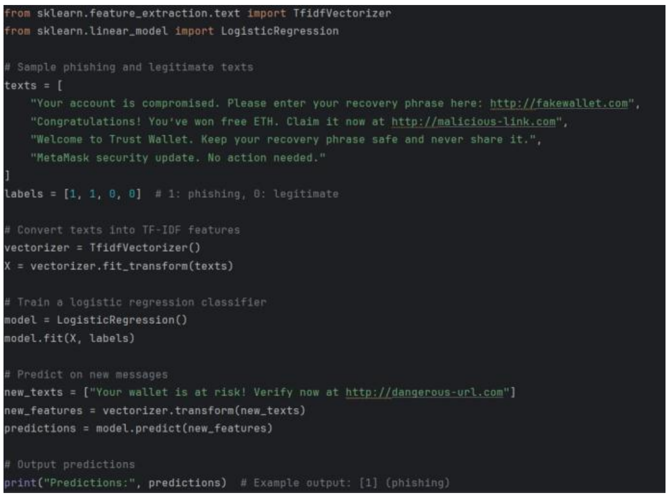


```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import numpy as np

# Sample dataset: URLs labeled as phishing (1) or legitimate (0)
urls = [
    "https://metamask-wallet.com", # phishing
    "https://metamask.io",         # legitimate
    "https://trustwallet.airdrop.info", # phishing
    "https://trustwallet.com"      # legitimate
]
labels = [1, 0, 1, 0]

# Convert URLs into feature vectors
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(urls)

# Train a Random Forest Classifier
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, labels, test_size=0.5, random_state=42)
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predict on new URLs
test_urls = ["https://metamask-security.net", "https://trustwallet.io"]
test_features = vectorizer.transform(test_urls)
predictions = model.predict(test_features)

# Output predictions
print("Predictions:", predictions)  # Example output: [1, 0] (phishing, legitimate)
```

**Fig. 5. URL monitoring system with AI**

Text-Based Phishing Detection Using NLP: A natural language processing model that analyzes the text content of phishing emails, social media posts, or dApp descriptions. Example approach:

- Fine-tune a transformer-based model (like BERT or RoBERTa) on labeled phishing and legitimate text samples.
- Identify common phishing tactics, such as urgent calls to action, grammatical errors, or offers that seem too good to be true.
- Integrate the model into messaging platforms or wallet apps to provide users with real-time alerts when suspicious messages are detected.

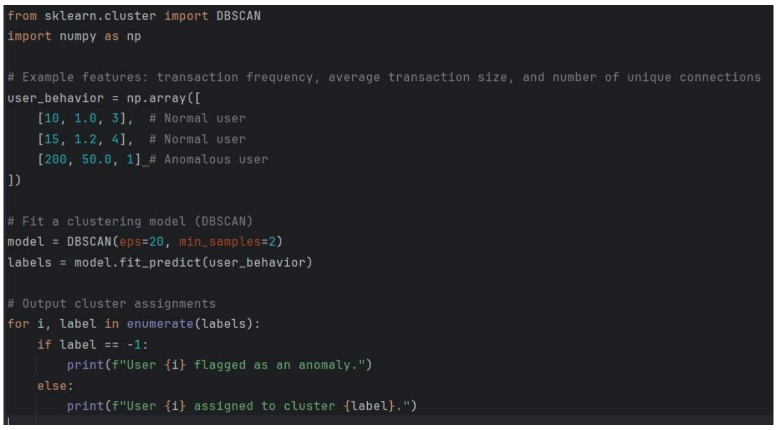


```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Sample phishing and legitimate texts
texts = [
    "Your account is compromised. Please enter your recovery phrase here: http://fakewallet.com",
    "Congratulations! You've won free ETH. Claim it now at http://malicious-link.com",
    "Welcome to Trust Wallet. Keep your recovery phrase safe and never share it.",
    "MetaMask security update. No action needed."
]
labels = [1, 1, 0, 0]  # 1: phishing, 0: legitimate

# Convert texts into TF-IDF features
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(texts)

# Train a logistic regression classifier
model = LogisticRegression()
model.fit(X, labels)

# Predict on new messages
new_texts = ["Your wallet is at risk! Verify now at http://dangerous-url.com"]
new_features = vectorizer.transform(new_texts)
predictions = model.predict(new_features)

# Output predictions
print("Predictions:", predictions)  # Example output: [1] (phishing)
```

**Fig. 6. Text-Based Phishing Detection Using NLP**

User Behavior Anomaly Detection: An unsupervised model that detects unusual patterns in how users interact with blockchain wallets or smart contracts. Example approach:

- Use clustering algorithms or autoencoders to learn normal usage patterns.
- Detect deviations such as multiple access attempts from unfamiliar IP addresses, repeated connections to unknown smart contracts, or sudden high-value transactions initiated after a user clicks a questionable link.
- When an anomaly is detected, trigger warnings or temporarily restrict high-risk actions.



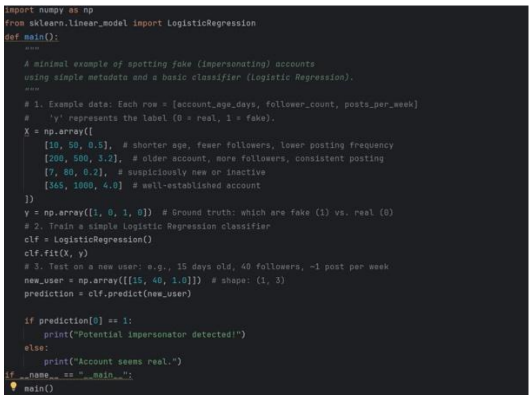
```python
from sklearn.cluster import DBSCAN
import numpy as np

# Example features: transaction frequency, average transaction size, and number of unique connections
user_behavior = np.array([
    [10, 1.0, 3],  # Normal user
    [15, 1.2, 4],  # Normal user
    [200, 50.0, 1] # Anomalous user
])

# Fit a clustering model (DBSCAN)
model = DBSCAN(eps=20, min_samples=2)
labels = model.fit_predict(user_behavior)

# Output cluster assignments
for i, label in enumerate(labels):
    if label == -1:
        print(f"User {i} flagged as an anomaly.")
    else:
        print(f"User {i} assigned to cluster {label}.")
```

**Fig. 6. Behavior Anomaly Detection Using Clustering**

Social Media Account Impersonation Detection**.** A model designed to spot fake profiles or accounts impersonating trusted entities. Example approach:

- Train a classifier using profile metadata, posting history, and engagement patterns.
- Incorporate graph-based approaches to analyze the social network structure, identifying clusters of related fake accounts.
- Use image recognition to detect doctored profile pictures or logos that closely resemble official accounts but have subtle differences.

- By deploying these AI models, the blockchain community can enhance its ability to preempt phishing attempts, protect users' sensitive information, and build trust in decentralized ecosystems.



```python
import numpy as np
from sklearn.linear_model import LogisticRegression
def main():
    """
    A minimal example of spotting fake (impersonating) accounts
    using simple metadata and a basic classifier (Logistic Regression).
    """
    # 1. Example data: Each row = [account_age_days, follower_count, posts_per_week]
    #    'y' represents the label (0 = real, 1 = fake).
    X = np.array([
        [10, 50, 0.5],  # shorter age, fewer followers, lower posting frequency
        [200, 500, 3.2],  # older account, more followers, consistent posting
        [7, 80, 0.2],  # suspiciously new or inactive
        [365, 1000, 4.0]  # well-established account
    ])
    y = np.array([1, 0, 1, 0])  # Ground truth: which are fake (1) vs. real (0)
    # 2. Train a simple Logistic Regression classifier
    clf = LogisticRegression()
    clf.fit(X, y)
    # 3. Test on a new user: e.g., 15 days old, 40 followers, ~1 post per week
    new_user = np.array([[15, 40, 1.0]])  # shape: (1, 3)
    prediction = clf.predict(new_user)

    if prediction[0] == 1:
        print("Potential impersonator detected!")
    else:
        print("Account seems real.")
if __name__ == "__main__":
    main()
```

**Fig. 7. Social Media Account Impersonation Detection**

## VII.CONCLUSION

This paper has examined a range of security threats confronting blockchain networks—such as 51% attacks, smart contract exploits, Sybil attacks, and phishing schemes—and has highlighted how AI-driven techniques can serve as a powerful defense. By leveraging machine learning, predictive analytics, and anomaly detection, decentralized platforms can more effectively identify fraudulent activities in real time, strengthen their resilience against malicious actors, and adapt to emerging cyber threats.

In particular, AI methods offer proactive measures: from machine learning models that detect and mitigate vulnerabilities in smart contracts to anomaly-detection algorithms that identify unusual network behaviors indicative of Sybil attacks or phishing attempts. These strategies not only help prevent catastrophic breaches but also reduce the reliance on purely manual or reactive security measures. Furthermore, AI-driven security solutions can be seamlessly integrated into existing blockchain infrastructures, supporting both on-chain and off-chain analysis to safeguard user funds and preserve network integrity.

Looking ahead, continued research and development in this field will be vital. As blockchain technology evolves and new decentralized applications gain traction, the sophistication of attackers is also likely to grow. The integration of more advanced AI techniques—such as graph-based neural networks, deep reinforcement learning, and large language models for code analysis—will be essential for proactively uncovering vulnerabilities and bolstering trust in decentralized finance (DeFi) and other Web3 services. Ultimately, the synergy between AI and blockchain holds the promise of creating self-learning, adaptive security layers that protect users, enhance the reliability of decentralized ecosystems, and further the adoption of blockchain technologies.

## REFERENCES

[1]. Katsitadze, N.: DeFi and NFT Adoption in Blockchain Financial Analysis Using Transaction Data. American Journal of Engineering Research (AJER), vol. 14, no. 3, pp. 6–9. AJER Publications.
[2]. Katsitadze, N. (2025). Blockchain-based authentication – A security and performance evaluation. *American Journal of Engineering Research, 14*(3).
[3]. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.
[4]. Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. O'Reilly Media.