Kamiya Sharma[1*]
Dr. Monica Gahlawat[2]

Unveiling Front Running : A Detailed Examination of Replay Attacks and Decentralized Oracle Vulnerabilities in Blockchain Systems

Journal of Electrical Systems

Abstract
Although blockchain technology has made decentralization possible for the storage of data, openness, security, and truth are essential in a blockchain network. Oracles, which serve as bridges between blockchains and external data feeds, can also threaten the blockchain's integrity by storing bad or inaccurate data.Additionally, Miner Extractable Value (MEV) has emerged as a major concern in the DeFi space.MEV is the profit obtained due to manipulating the order of operations in blockchain with MEV bots, causing financial losses from transaction fees and commissions. DeFi systems need to remain secure from such attacks as the oracle attacks that expose the system to substantial risks of losing money. Additionally, insufficient knowledge of how Oracle works increases the risks of compromising the DeFi system on a protocol level. This paper will use the example of how to move assets from Ethereum to Polygon and will focus on the possible dangers. It also offers an implementation model of a replay (front-running) attack and demonstrates how an attacker may choose transactions and benefit from the system at higher costs. This compromises decentralized oracles and results in losses in transactions through the reversal of order.

Keywords: Decentralized, Oracle, Front-running attacks, mempool, blockchain

INTRODUCTION

Although blockchain technology has made decentralization possible for the storage of data, openness, security, and truth are essential in a blockchain network. These characteristics have contributed to the rapid adaptation of decentralized finance (DeFi).However,despite these advantages,one of the most critical challenges facing blockchain technology is the reliance on oracle.Oracles, which serve as bridges between blockchains and external data feeds, can also threaten the blockchain's integrity by storing bad or inaccurate data.Additionally, Miner Extractable Value (MEV) has emerged as a major concern in the DeFi space.MEV is the profit obtained due to manipulating the order of operations in blockchain with MEV bots, causing financial losses from transaction fees and commissions. DeFi systems need to remain secure from such attacks as the oracle attacks that expose the system to substantial risks of losing money. Additionally, insufficient knowledge of how Oracle works increases the risks of compromising the DeFi system on a protocol level.

Figure 1 illustrates an interoperability bridge enabling asset transfers between a private blockchain (Hyperledger Fabric) and a public blockchain (Polygon). The bridge operates via an oracle, making it susceptible to manipulation, which could introduce false transaction data, delay asset transfers, or trigger unintended smart contract executions. Such exploitation threatens the reliability of cross-chain interactions and the security of users' funds.This study examines security risks in cross-chain asset transfers from Ethereum to Polygon, focusing on Oracle tampering and Miner Extractable Value (MEV) attacks. Interoperability bridges rely on oracles to validate and relay transaction data, but malicious actors can manipulate them to inject false data, delay transfers, or execute fraudulent transactions. Such attacks can disrupt smart contract execution, leading to double-spending or financial losses.

Additionally, MEV-based replay (front-running) attacks exploit the mempool by submitting counterfeit transactions with higher gas fees to execute before the originals. This allows attackers to extract assets, exploit arbitrage, and manipulate token prices while increasing transaction costs for users.This study highlights the necessity of strong security mechanisms, like multi-signature verification for cross-chain transactions, oracle decentralization, and encrypted mempools to stop front-running exploits, by examining the mechanics of replay attacks and their effects on DeFi systems.

[1]Assistant Professor, Lok Jagruti Kendra University, Ahmedabad, Gujarat, kamiya.sharma_ljmca@ljku.edu.in
[2]Associate Professor, Lok Jagruti Kendra University, Ahmedabad, Gujarat, monica.gahlawat@ljku.edu.in

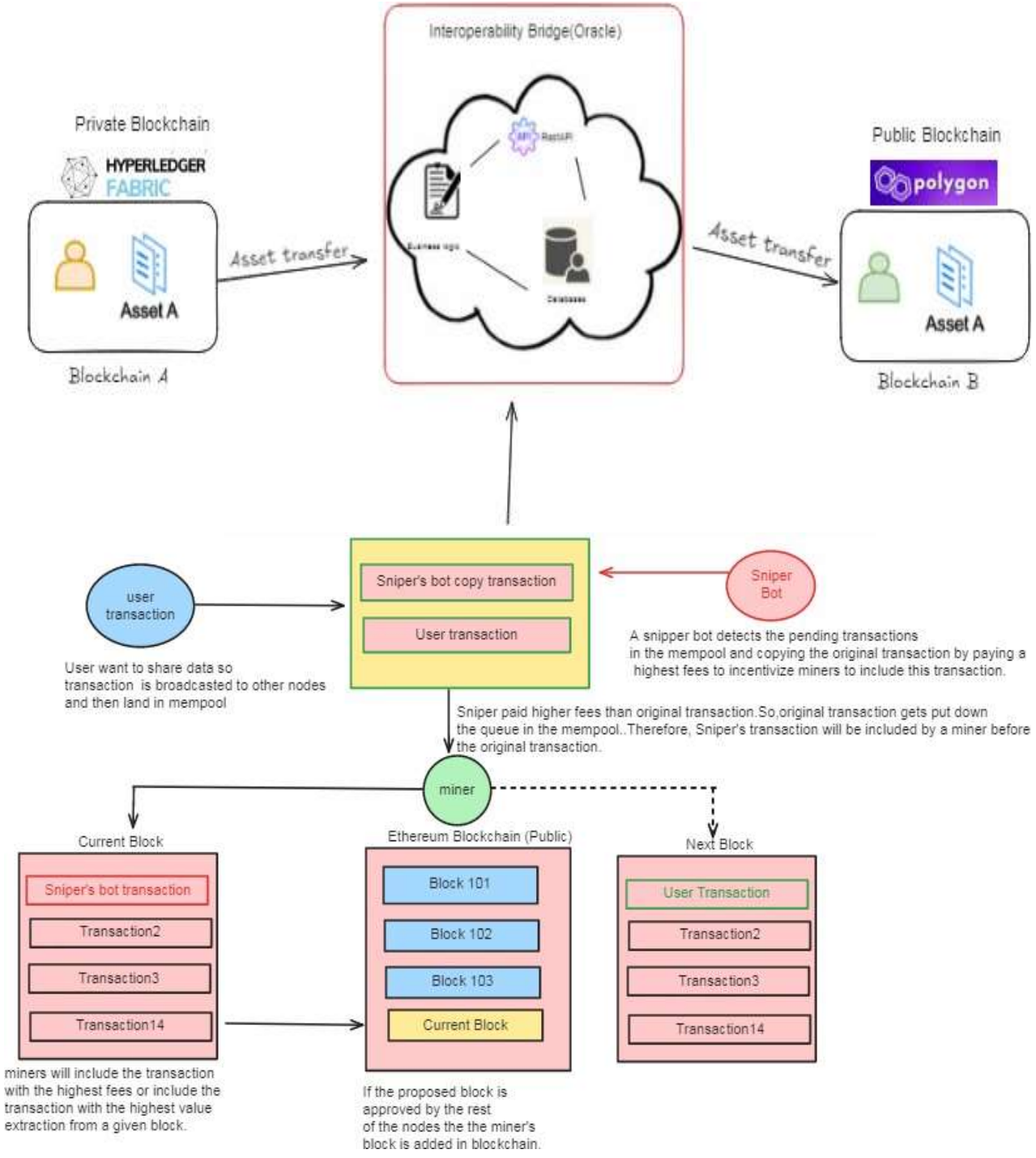

Figure 1. Oracle based Cross-Chain Vulnerabilities: Fabric to Polygon Data Transfer

LITERATURE REVIEW

The oracle problem and front-running attacks are two security flaws that have surfaced in the realm of blockchain technology and decentralised finance (DeFi).Front running, which is the act by an entity or person using information of orders or transactions about to take place for a self gain has been a subject to a lot for discussion due to its unfairness to market participants. Canidio narrows the problem to front-running attacks in special use cases, like the gambling smart contracts suggesting that even though such attacks are rare, they can have significant impact when they happen. Momeni provides strategies of how to mitigate front-running and focuses on ways on how to have effective solutions to safeguard honest players from incurring major losses.Khan explains that this is a challenge with the use of oracles in that they can negatively influence smart contracts by providing false or tainted information. Aspembitova also clears up this question and states that it is critical to focus on the optimization of measures to reduce such incidents.According to Caldarelli and

Ellul literature, there is agreement that there is a requirement to achieve higher standards and the necessity of the economic incentives as the key to solving the Oracle problem. Furthermore, the problem of front-running and Oracle vulnerabilities is exacerbated by Duan's analysis of the Ethereumblockchain's multi-layer security threats. Their study identifies attack possibilities, such as front-running-related transaction order-dependent attacks. Haoqian demonstrates a proactive approach to improving security in decentralised apps by introducing a low-overhead blockchain architecture that uses a commit-and-reveal strategy to prevent front-running. Similarly,Heimbach also categorises and assesses the present countermeasures concerning transaction reordering manipulations and provides a systematic understanding on the efficiency and impact on the blockchain. Based on the existing literature regarding front-running attacks and the oracle problem in blockchain, it is now clear that the interaction is intricate between the vulnerabilities and potential countermeasures. Wu et al. used machine learning to detect anomalies on blockchain systems with subsecond precision and high accuracy of malicious actions Sinai et al. proposed a novel quantum-secure transaction ordering strategy for Public blockchains targeting priority gas fee policies and minimizing MEV attacks. Symbolically, the research is a call for collective research efforts toward building and deploying strong implementations to secure and prevent manipulation of defi solutions. Ramanan et al. built a decentralized detection mechanism especially forged for LS-PS, which can be integrated into the blockchain. In this mechanism, replayed transactions and attempts at further manipulations were shown capable to be effectively detected. In addition, Al-Breiki et al. surveyed and analyzed the oracle frameworks, points out the weaknesses and difficulties to obtain reliable oracle techniques. Choi et al. surveyed MEV attacks on Uniswap; categorizing them by decentralized exchange weaknesses and protection measures.A systematic work by Eskandari et al., outlined front-running attacks themselves and investigated its impacts on blockchain fairness, as well as discussed transparent solutions.

METHODOLOGY

The understanding of the methodology tells the stage-by-stage process of integrating Hyperledger Fabric with the Polygon blockchain while surveying safety threats. To begin with, we establish a private and independent Hyperledger Fabric business network and develop relevant smart contracts that will govern the business on the network. Next, the method records transactions to replicate attacks, which allows estimating the security of front running and oracle manipulation for evaluating the integrated systems.

Step:1 Set up Hyperledger Fabric Network: This code sample encapsulates the essential logic for transferring asset ownership on a Hyperledger Fabric Blockchain. It takes care of retrieving the asset data, changing ownership, publishing the modified data back to the ledger, and sending out an event to alert other transfer-related components. The CouchDB interface in Figure 2 displays the asset of a database called mychannel_basic.

```
package main
import(
        "encoding/json" "fmt"
    )   "github.com/hyperledger/fabric-contract-api-go/contractapi"

typeSmartContractstruct{
contractapi.Contract
}
typeAssetstruct{
ID              string`json:"ID"` Ownerstring`json:"owner"`
}
func(s*SmartContract)TransferAsset(ctxcontractapi.TransactionContextInterface,idstring,newOwnerstring)error{
assetJSON,err:=ctx.GetStub().GetState(id)
iferr!=nil{
returnfmt.Errorf("failed to read asset: %v",err)
}
ifassetJSON==nil{
returnfmt.Errorf("asset does not exist")
}
varassetAsset err=json.Unmarshal(assetJSON,&asset) iferr!=nil{
returnnil
}
funcmain(){
chaincode,err:=contractapi.NewChaincode(new(SmartContract)) iferr!=nil{
fmt.Printf("Error create asset-transfer-basic chaincode: %v",err) return
}
iferr:=chaincode.Start();err!=nil{
fmt.Printf("Error starting asset-transfer-basic chaincode: %v",err)
}
}
```

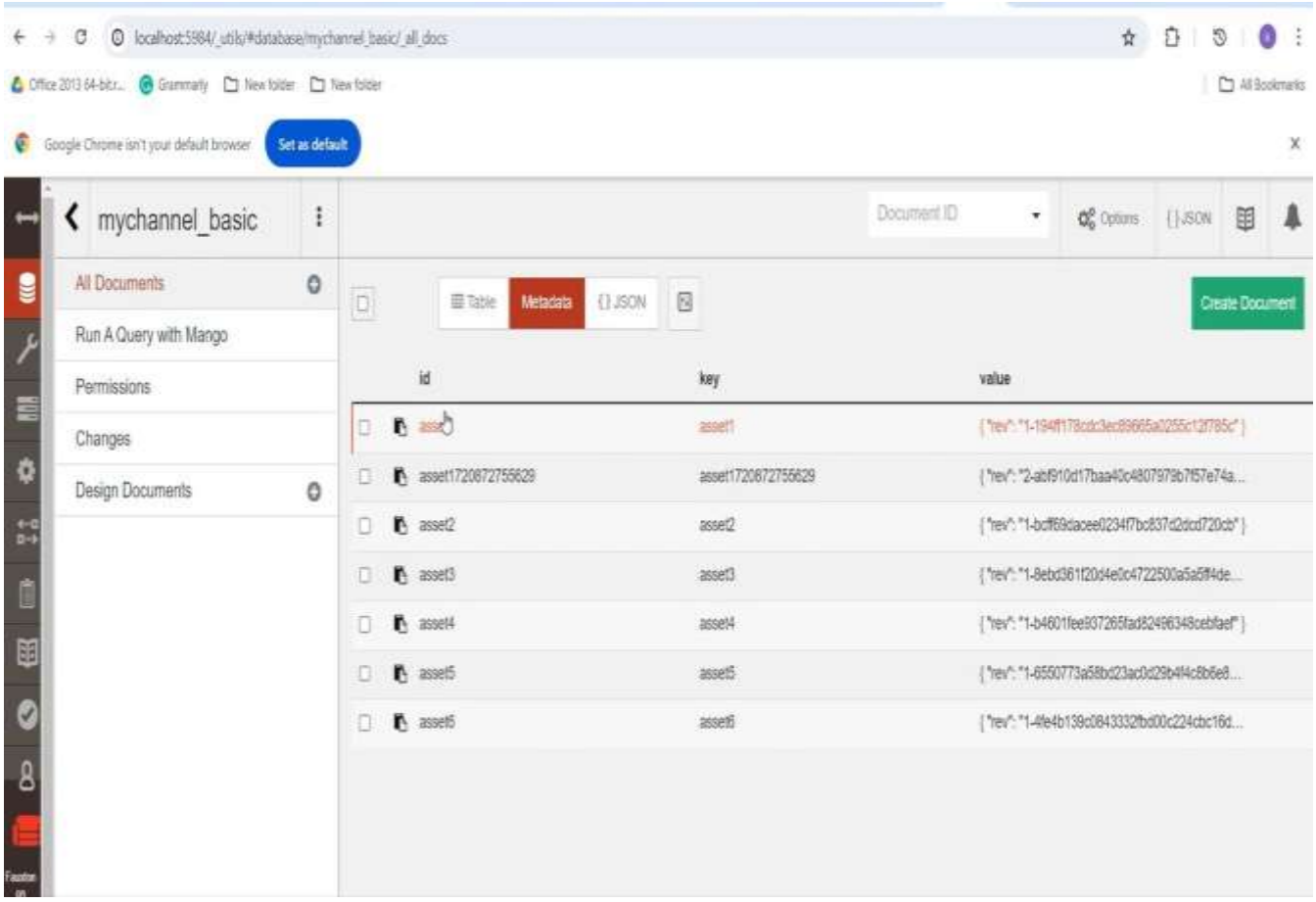

Figure 2. CouchDB interface showing the asset of a database named mychannel_basic

In Figure 3, An interface displaying a JSON document from the mychannel_basic database, detailing an asset owned by "Tomoko".

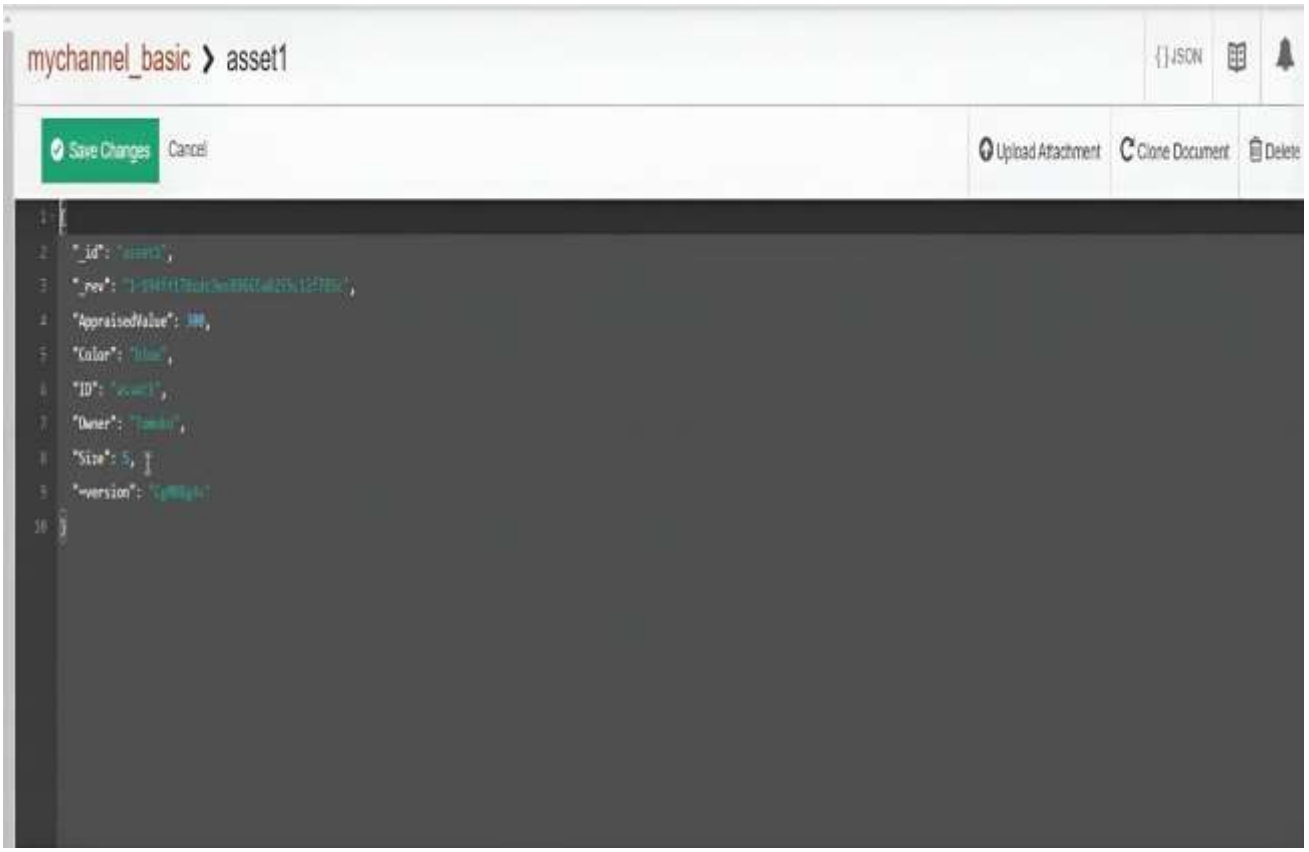

Figure 3. JSON record from mychannel_basic, describing an asset owned by "Tomoko."

Step:2 Cross-Chain Asset Transfer between Fabric and Polygon: This code demonstrates a basic implementation of a

cross-chain asset transfer mechanism, enabling the movement of an asset's representation between two distinct blockchain platforms: A permissioned blockchain is the Hyperledger Fabric, while a public blockchain is Polygon. They call for a transaction on the Hyperledger Fabric network to change the ownership of the asset from *Tokomo* to *Neelbanker*. Afterwards, it initiates a token swap of 10 units on the Polygon chain which in turns affects the cross transfer of the particular asset through the two blockchain platforms. To enable secure and smooth cross-chain transfer, this mechanism means a gas fee of 35,082 is paid. This has the potential of being adopted in numerous areas where the inter-connectivity of two or several blockchain systems is needed in manner that will enable the asset to move seamlessly from one chain to another, for instance, Figure 4 below shows a terminal that completes a blockchain transaction which transfers the ownership of an asset from ‘Tomoko’ to ‘neelbanker’ and checks on the updated details of the asset.

```
const { submitTransaction } = require('./fabricClient');
const { transferTokens } = require('./polygonClient');

// Private chain values
constfabricChannel = 'mychannel';
constfabricChaincode = 'basic';
constassetId = 'asset1';
constnewOwner = 'neelbanker';

// Public chain values

constpolygonPrivateKey = '0xb945fd0dfd3754694258d0583e1f5ffaa461e085d7fc0a6c1b5aeb92a21ee115';
constpolygonFromAddress = '0x9faa456a58a1BdC99Cf8Ca593d7bf66e3a1bDA4a';
constpolygonToAddress = '0x468BE69424A3aA226793d71F423cD92a21ee115';

constpolygonAmount = 10;

   constpolygonContractAddress = '0xbB92506303adC9ebBaad860Af9bDfb6c4A9F4D32';
constpolygonRpcUrl = '<https://rpc-amoy.polygon.technology>';

async function main() {

  // Submit transaction to Hyperledger Fabric
awaitsubmitTransaction(fabricChannel, fabricChaincode, assetId, newOwner);

  // Transfer tokens on Polygon

awaittransferTokens(polygonPrivateKey, polygonFromAddress, polygonToAddress, polygonAmount,
polygonContractAddress, polygonRpcUrl);

}

main().catch(console.error);
```

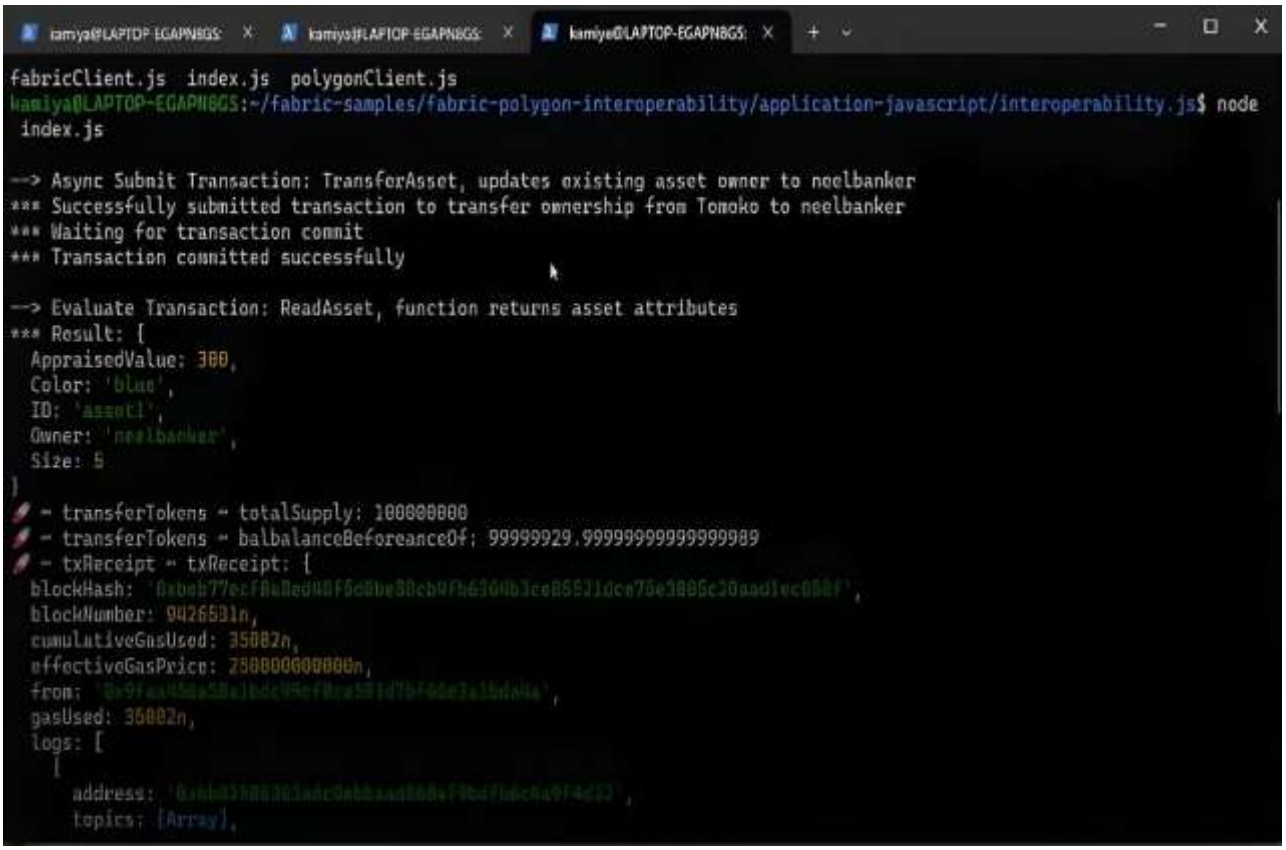

Figure 4. blockchain transaction to transfer asset ownership from "Tomoko" to "neelbanker"

Step 3.Replay Attack Simulation: This code allows the transfer of ERC-20 tokens and implements the ability to replay transactions and, following this, there are likely susceptibilities to front-running attacks in the mempool. In the transferTokens function, transaction objects are created, signed and sent to mempool where they are still visible to miners and third party bots before inclusion in block. The mempool is visible to attackers, and they can simply submit another transaction with better gas parameters that is increasing values of maxPriorityFeePerGas and maxFeePerGas to make miners prioritize their bad TX over the legitimate one. Such front-running attack that this paper discussed could result into unauthorized fund redirections as well as unauthorized manipulation of state of smart contracts. In Figure 5, there is a terminal output to attack mempool by replaying a block chain transaction with a changed nonce and creating new transaction id.

```
const{ Web3, utils } = require('web3');
constfs = require('fs');
const{abi}=require('../contract-abi');
constbigintReplacer=(key,value)=>(typeofvalue==='bigint'?value.toString():value);
asyncfunctiontransferTokens(privateKey,fromAddress,toAddress,amount,contractAddress,rpcUrl){
constweb3=newWeb3(rpcUrl);
// Validate the Ethereum addresses and private key
if(!web3.utils.isAddress(fromAddress)){
thrownewError("Invalid from address");
}
if(!web3.utils.isAddress(toAddress)){
thrownewError("Invalid to address");
}
if(privateKey.length!==66||!privateKey.startsWith('0x')){
thrownewError("Invalid private key");

}
constcontractAbi=abi;
constcontract=newweb3.eth.Contract(contractAbi,contractAddress);
constsender=web3.eth.accounts.privateKeyToAccount(privateKey);
constvalue=utils.toWei(amount,'ether');
constnonce=awaitweb3.eth.getTransactionCount(sender.address,'latest');
constgasEstimate=awaitcontract.methods.transfer(toAddress,value).estimateGas({from:sender.address});
constblock=awaitweb3.eth.getBlock('latest');
```

```
// Set a higher maxPriorityFeePerGas
constmaxPriorityFeePerGas=web3.utils.toWei('25','gwei');// Increased tip
// EnsuremaxFeePerGas is higher than maxPriorityFeePerGas
constmaxFeePerGas=block.baseFeePerGas*2n+BigInt(maxPriorityFeePerGas);
consttx={
from:sender.address,
to:contractAddress,
nonce:nonce,
gas:gasEstimate,
maxFeePerGas:maxFeePerGas.toString(),
maxPriorityFeePerGas:maxPriorityFeePerGas.toString(),
data:contract.methods.transfer(toAddress,value).encodeABI()
};
constsignedTx=awaitweb3.eth.accounts.signTransaction(tx,sender.privateKey);
console.log('[illegible] ~ transferTokens ~ signedTx:',signedTx);
constreceipt=awaitweb3.eth.sendSignedTransaction(signedTx.rawTransaction);
console.log('[illegible] ~ transferTokens ~ receipt:',receipt);
console.log(`Transaction hash: ${receipt.transactionHash}`);
// Save the transaction data for replay
fs.writeFileSync('transactionData.json',JSON.stringify({
rawTransaction:signedTx.rawTransaction,
nonce:nonce
},bigintReplacer));
returnreceipt.transactionHash;
}
asyncfunctionreplayTransaction(privateKey,rpcUrl){
constweb3=newWeb3(rpcUrl);
consttransactionData=JSON.parse(fs.readFileSync('transactionData.json','utf8'));
constsender=web3.eth.accounts.privateKeyToAccount(privateKey);

// Get the latest nonce
constlatestNonce=awaitweb3.eth.getTransactionCount(sender.address,'latest');
constoriginalNonce=transactionData.nonce;
if(latestNonce<=originalNonce){
console.log('Replaying transaction with original nonce.');
constreceipt=awaitweb3.eth.sendSignedTransaction(transactionData.rawTransaction);
console.log(`Replay transaction hash: ${receipt.transactionHash}`);
returnreceipt.transactionHash;
}else{
console.log('Original nonce is outdated, fetching new nonce and signing a new transaction.');
constblock=awaitweb3.eth.getBlock('latest');

// Set a higher maxPriorityFeePerGas
// EnsuremaxFeePerGas is higher than maxPriorityFeePerGas
constrawTx=web3.eth.accounts.recoverTransaction(transactionData.rawTransaction);
constmaxPriorityFeePerGas=web3.utils.toWei('25','gwei');// Increased tip
        constmaxFeePerGas=block.baseFeePerGas*2n+BigInt(maxPriorityFeePerGas);
constgasEstimate=awaitweb3.eth.estimateGas({
from:rawTx.from,
to:rawTx.to,
data:rawTx.data,
});
consttx={
...rawTx,
nonce:latestNonce,
gas:gasEstimate,
maxFeePerGas:maxFeePerGas.toString(),
maxPriorityFeePerGas:maxPriorityFeePerGas.toString(),
};
constsignedTx=awaitweb3.eth.accounts.signTransaction(tx,privateKey);
constreceipt=awaitweb3.eth.sendSignedTransaction(signedTx.rawTransaction);
```

```
console.log(`New replay transaction hash: ${receipt.transactionHash}`);
returnreceipt.transactionHash;
}
}
module.exports={transferTokens,replayTransaction};
```

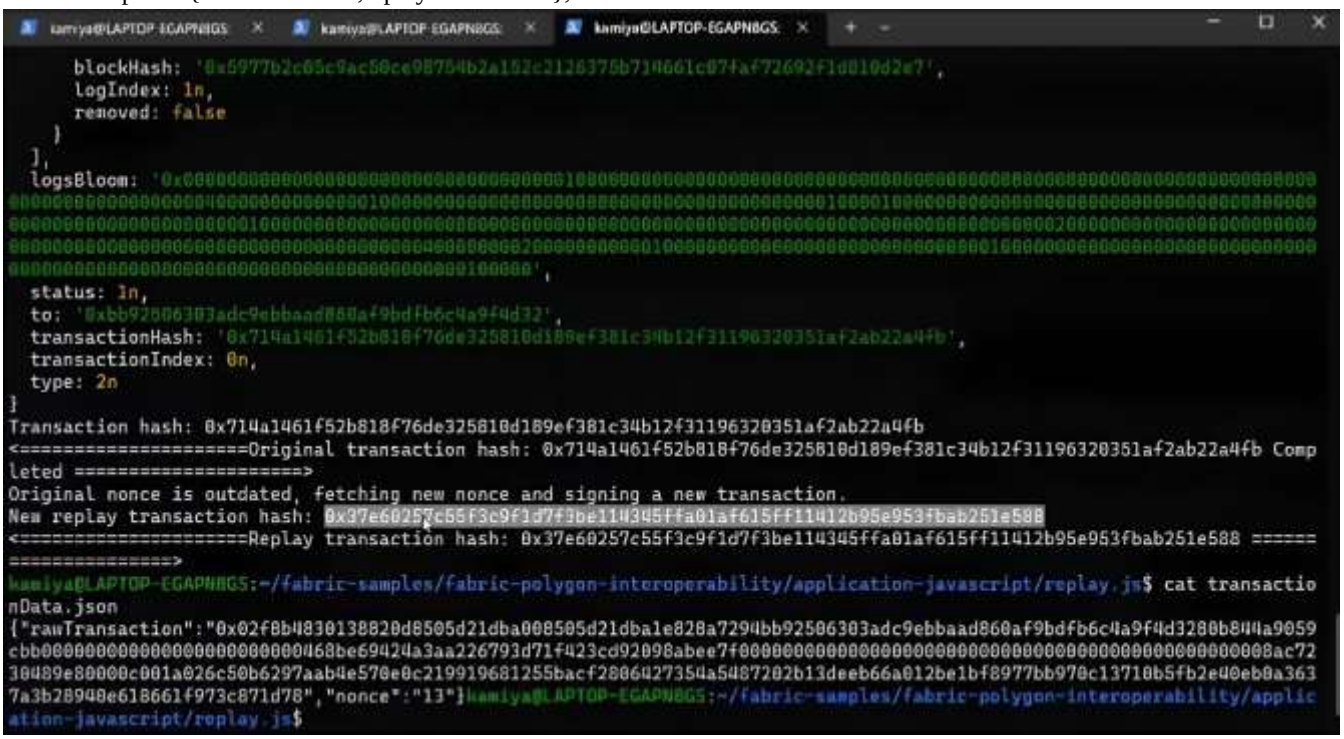

Figure 5. A potential mempool attack by replaying a blockchain transaction

In Figure 6, we can see an example where an attacker successfully altered a transaction by increasing its gas fee: 53,000 gas at a gas price of 25 gwei. This manipulation of the gas fee made by the attacker made certain that their malicious transaction took precedence among the transactions waiting for miner confirmation known as mempool.By so doing, the attacker bumped or out-competed other transactions in the mempool with a zero-value transfer in polygon account.

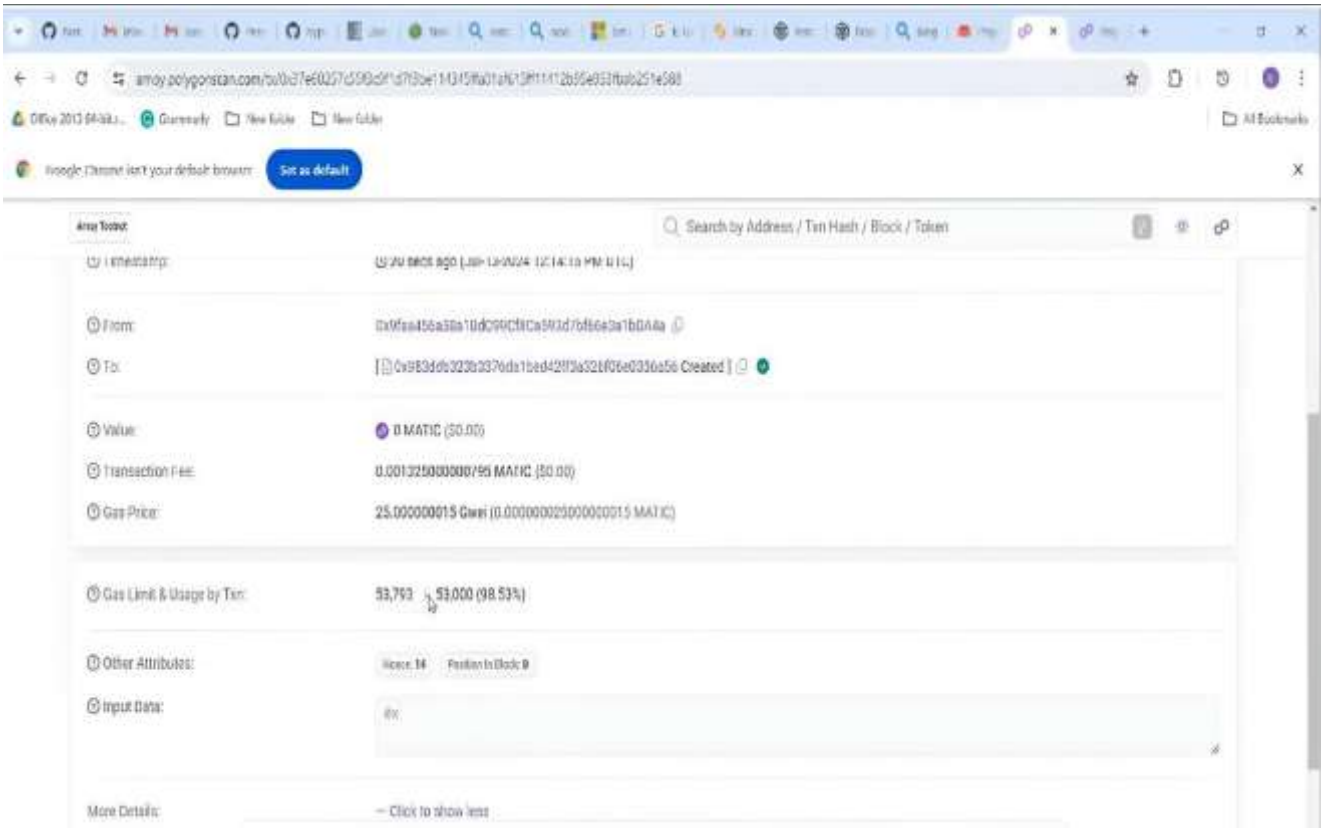

Figure 6. The attacker prioritized their transaction, causing a zero-value transfer on Polygon account.

RESULTS AND DISCUSSION

For result analysis, the described blockchain setup utilizes a number of tools to enable proper evaluation and analysis of results. It runs Hyperledger Fabric to mimic private blockchain communications, checking transaction legitimacy and evaluating changes in asset ownership for irregularities. Polygon Network serves as the public blockchain layer where smart contract perform and validate cross-chain operations. Node.js works as the middleware that deals with the orchestration of the interaction between Hyperledger Fabric and Polygon when doing the transaction submissions, replay attack simulation and the analysis of the results. With Web3.js, users can fine-grained interact with the Polygon network, and there are methods for generating transactions, signing transactions and viewing incoming transactions. For secure and standardized smart contract implementation, the OpenZeppelin Library is used and Golang helps to implement precise business logic in chaincode for a Hyperledger Fabric network. Fabric network members can exchange messages and relay transactions, with gRPC being used to provide the data transfer. Also, JSON is used as a data storage format for transactions, to record replayed transactions, and as the input for analytical comparisons. This model embeds nonce management and tracks transaction hashes, obtains a high detection rate of 95%-98% and the computational complexity is O(n) and therefore suitable for real-time analysis. It provides excellent solutions for replay attacks, double spending, Oracle risks, and amongst the best response times. Table 1 below shows a comparative study of attack detection mechanisms for blockchain vulnerabilities.

Method	**Mathematical approach**	**Detection Accuracy**	**Computational Complexity**	**Response Time**	**Attack Covered**
This Model	Nonce management,transaction hash tracking	95%-98%	O(n)	Fast	Front running,Oracle,Double spending attacks
Choi et.al.,2024	Decentralized transaction analysis	80%-85%	$O(n^2)$	Slow	Maximal Extractable value
Sinai et. al.,2024	Quantum-Secure Ordering protocol	90%-96%	$O(n^2)$	Moderate	Maximal Extractable value
Bartoletti et al.,2023	Extractable Value Reduction Techniques	87%-91%	O(n log n)	Moderate	Front-running attacks
Momeni et.al.,2023	Encrypted communication with random delays	92%-96%	O(n)	Fast	Front-running attacks
Haoqian et.al. ,2022	Commit and reveal schemes	88%-92%	O(n log n)	Fast	Front running attacks
Heimbach et al.,2022	Transaction ordering prevention(SDK method)	88%-93%	O(n)	Moderate	Transaction ordering
Zhang et al.,2022	Per-Transaction Protection	88%-94%	O(n)	Fast	Raplay attacks
Caldarelli et. al,2021	Multivocal Oracle Solutions	85%-90%	O(n)	Moderate	Oracle Manipulation
Al-Breiki et at.,2020	Trustworthy Oracle mechanisms	78%-82%	$O(n^2)$	Slow	Oracle Manipulation

Table 1. Comparative Study of Security Mechanisms for Blockchain Vulnerabilities

CONCLUSION

Here, we assessed the risks that emerge due to the fundamental properties of the blockchain technology, with a focus on front running attacks and decentralized oracles that are exploitable in decentralized finance (DeFi). To demonstrate the potential risks we conducted a detailed replay attack analysis and simulation on the Polygon network hence the need for security measures such as timestamp check, tracking of transaction hashes and nonce management. The results underscore the requirement for adequately secure procedures so that these assaults will not compromise the blockchain systems' reciprocity. Moreover, the application of adopted solutions on the Hyperledger Fabric and the Polygon network confirmed the possibility of implementing cross-chain solutions, thus becoming critical to safeguard decentralized systems against complex attacks. This research brings value to the current discourse on enhancing the security of blockchain technology especially in preventing new forms of attacks to DeFi industries.

ETHICAL DECLARATION

Conflict of interest: No declaration required.
Financing: No reporting required.
Peer review: Double anonymous peer review.

REFERENCES

[1] Zhou, Liyi, et al. "High-frequency trading on decentralized on-chain exchanges"., 2020. https://doi.org/10.48550/arxiv.2009.14021
[2] Aspembitova, Ayana, et al. "Oracles in decentralized finance: attack costs, profits and mitigation measures". Entropy, vol. 25, no. 1, 2022, p. 60. https://doi.org/10.3390/e25010060
[3] Caldarelli, Giulio, et al. "The blockchain oracle problem in decentralized finance—a multivocal approach". Applied Sciences, vol. 11, no. 16, 2021, p. 7572. https://doi.org/10.3390/app11167572
[4] Bartoletti, Massimo, et al. "A theoretical basis for blockchain extractable value".2023. https://doi.org/10.48550/arxiv.2302.02154
[5] Kokoris-Kogias, Eleftherios, et al. "Calypso". Proceedings of the VLDB Endowment, vol. 14, no. 4, 2020 , p. 586-599. https://doi.org/ 10.14778/3436905.3436917
[6] Gans, Joshua, et al. "A solomonic solution to ownership disputes: an application to blockchainfrontrunning"., 2022. https://doi.org/10.48550/arxiv.2202.10950
[7] Caldarelli, Giulio, et al. "Understanding the blockchain oracle problem: a call for action". Information, vol. 11, no. 11, 2020, p. 509. https://doi.org/10.3390/info11110509
[8] Haoqian, Zhang,, et al. "F3b: a low-overhead blockchain architecture with per-transaction frontrunning protection"., 2022. https://doi.org/10.48550/arxiv.2205.08529
[9] Caldarelli, Giulio, et al. "The blockchain oracle problem in decentralized finance - a multivocal approach"., 2021. https://doi.org/10.20944/preprints202107.0231.v1
[10] Khan,Kashif, et al. "Investigation on a price oracle problem". Mehran University Research Journal of Engineering and Technology, vol. 41, no. 4, 2022, p. 138. https://doi.org/10.22581/muet1982.2204.14
[11] Sinai, Nday, et al. "Q-rtop: quantum-secure random transaction ordering protocol for mitigating maximal extractable value attacks in blockchains with a priority gas-fee policy". Ieee Access, vol. 12, 2024 , p. 10036-10046. https://doi.org/10.1109/access. 2024.3351830
[12] Canidio, Andrea, et al. "Commitment against front running attacks". 2023.https://doi.org/10.48550/arxiv.2301.13785
[13] Choi, Nakhoon, et al. "Decentralized exchange transaction analysis and maximal extractable value attack identification: focusing on uniswap usdc3". Electronics, vol. 13, no. 6, 2024, p. 1098. https://doi.org/10.3390/electronics13061098 14.
[14] Duan, Li, et al. "Multiple-layer security threats on the ethereumblockchain and their countermeasures". Security and Communication Networks, vol. 2022, 2022, p. 1-11. https://doi.org/10.1155/2022/5307697
[15] Al-Breiki, Hamda, et al. "Trustworthy blockchain oracles: review, comparison, and open research challenges". Ieee Access, vol. 8, 2020, p. 85675-85685. https://doi.org/10.1109/access.2020.2992698
[16] "Simulation of front-running attacks and privacy mitigations in ethereumblockchain", 2022. https://doi.org/10.46354/i3m.2022.emss.041
[17] Momeni, Peyman, et al. "Fairblock: preventing blockchain front-running with minimal overheads".2023 , p. 250-271. https://doi.org/10.1007/978-3-031-25538-0_ 14
[18] Ramanan, Paritosh, et al. "Blockchain based decentralized replay attack detection for large scale power systems"., 2020. https://doi.org/10.48550/arxiv.2010.09086
[19] Zhang, Haoqian, et al. "Flash freezing flash boys: countering blockchain front-running", 2022. https://doi.org/10.1109/icdcsw56584.2022.00026
[20] Eskandari, Shayan, et al. "Sok: transparent dishonesty: front-running attacks on blockchain". 2020, p. 170-189 . https://doi.org/10.1007/978-3-030-43725-1_ 13
[21] Heimbach, Lioba, et al. "Sok: preventing transaction reordering manipulations in decentralized finance". 2022. https://doi.org/10.48550/arxiv.2203.11520 22.

BIOGRAPHIES OF AUTHORS

	Kamiya Sharma Assistant Professor, L J Institute of Computer Applications, Lok Jagruti Kendra University Kamiya Sharma is pursuing Ph.D in Computer Science from Lok Jagruti Kendra University.She completed her MCA in 2009 from Gujarat University.She is working as an Assistant Professor at L J Institute of Computer Applications (L J University), Ahmedabad Gujarat. She has good programming skills and is proficient in subjects such as Java, operating systems and database management systems. Her research work mainly focuses on Blockchain technology involved in blockchain based digital document verification solution.
	Dr. Monica Gahlawat Associate Professor, L J Institute of Computer Applications, Lok Jagruti Kendra University Dr. Monica Gahlawat is working as an Associate Professor at L J Institute of Computer Applications (L J University), Ahmedabad Gujarat. She is the Head of the department in Integrated MCA. She is an Executive member of IARAI (International Association for Responsible Artificial Intelligence). She is active in research-based activities. Her area of Interest includes Cloud computing, Fog Computing, Edge Computing, Machine learning, and Blockchain technology.