19th Aug 2023

ScriptTV
Blockchain Security Assessment

PREFACE

Objectives

This document aims to highlight any identified bugs/issues in the provided codebase. This audit has been conducted in a closed and secure environment, free from influence or bias of any sort. This document may contain confidential information about IT systems/architecture and the client's intellectual property. It also includes information on potential risks and the processes involved in mitigating/exploiting the risks mentioned below. The usage of the information provided in this report is limited, internally, to the client. However, this report can be disclosed publicly to aid our growing blockchain community; at the client's discretion.

Key Understandings

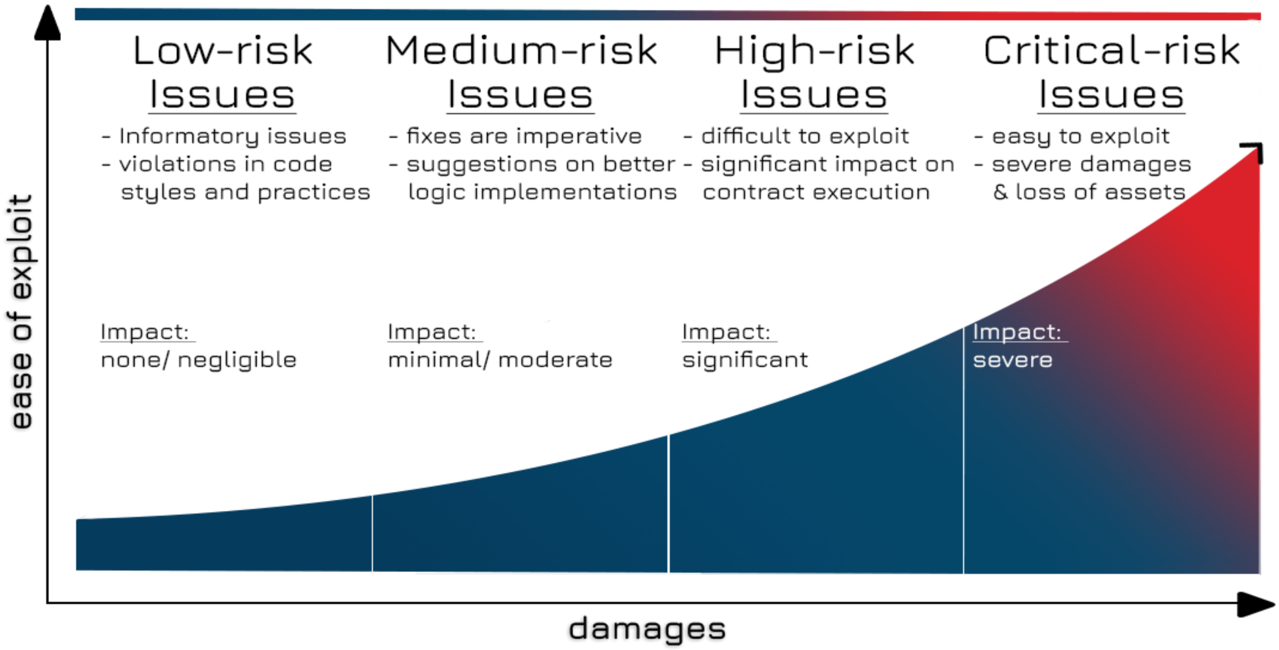


TABLE OF CONTENT

INTRODUCTION

BlockApex (Auditor) was contracted by ScriptTV (Client) to conduct a comprehensive security audit/Code Review. This document presents the findings of our analysis, which started on 19th july, 2023.

Name
ScriptTV Network
Auditors
Abdul Sami | Muhammad Jarir Uddin | Muhammad Abdullah | Gul Hameed
Platform
ScriptTV Network
Type of review
Manual Code Review | Automated Tools Analysis
Methods
Architecture Review | Functional Testing | Computer-Aided Verification | Manual Review
Git repository/ Commit Hash
Git Repo
White paper/ Documentation
White Paper
Document log
Initial Audit: 19th Aug 2023
Final Audit: 13 Oct 2023

Scope

The shared git-repository was checked for common code violations and vulnerability-specific probing to detect major issues/vulnerabilities. Some specific checks are as follows:

Code Review		Functional Review
Sybil Attack	Transaction Replay Attack	Business Logics Review
Eclipse Attack	Time-Locked Transaction Attack	Functionality Checks
Denial of Service Attack	Unchecked math	Access Control & Authorization
Http Input Attack	Unsafe type inference	Read-only Pointers
Long Range Attack	Selfing Mining	Overflows
EVM Incompatibility	Deployment Consistency	Uncaught Errors/Panics
Block Double Production	Race Attack	Out-of-Bounds error
Style guide violation	Data Consistency	Nil Pointer Exception
Cryptographic Attack	Indefinite Channel Deadlock	Precompile Invalidation
Chain Halt	Network Partitions	Goroutine Leaks

Project Overview

Script TV, a fork of the Theta Network is a decentralized video delivery network that furnishes an expansive range of blockchain-enabled solutions to the problems related to the traditional video-streaming sector. The platform offers high-quality video streaming as well as multiple incentive mechanisms for decentralized bandwidth and content-sharing at a reduced cost as compared to conventional service providers.

The Script Blockchain is a purpose-built blockchain designed for video and data relaying from the ground up. Script's unique multi-BFT consensus design combines a committee Validator Nodes with a second layer of community-run Validator Nodes. Validator Nodes propose and produce new blocks in the chain, while Guardian and Lightning Nodes seal blocks and act as a check on malicious or non-functional Validator Nodes.

The Script blockchain mainnet enables the support for Turing-complete smart contracts. Smart contracts open up a new set of user experiences and attribution models for DApps built on the Script network. In addition to the Validator and Lightning Nodes, the Script community members also host the Script Node, which forms the Script Network, a fully decentralized network for data delivery and, more generally, computing. This new fully decentralized technology stack adds the ability to capture live video, transcode it in real-time, cache and relay live stream video data to users globally - all through Script's P2P network run by thousands of community members. Not a single central server or service is used in this pipeline.

System Architecture

Introduction:
The Script Blockchain is a decentralized network designed to optimize video streaming delivery and incentivize users to share their excess bandwidth and computing resources. Its architecture is based on a hybrid Proof of Stake (PoS) and Byzantine Fault Tolerance (BFT) consensus mechanism, enabling efficient and secure video content distribution.

Components:

Validator Nodes:

These are the core nodes responsible for proposing the blocks and validation. Validator nodes are selected based on their Script tokens stake, participating in both the PoS and BFT mechanisms. They ensure the network's security and integrity.

Guardian Nodes:

Guardian nodes support validator nodes by relaying transactions, validating signatures, and participating in a lightweight PoS mechanism. They are also responsible for penalizing malicious behavior and ensuring the network remains robust.

Edge Nodes:

Edge nodes' purpose is to finalize the blocks. Edge nodes serve as caching and relay points for video content, reducing the load on the main blockchain. They contribute excess bandwidth and resources to the Script network and are rewarded with Script(S-PAY) tokens.

Smart Contracts:

Script's smart contract functionality allows developers to create decentralized applications (DApps) within the network. These contracts automate various processes, enhancing the network's versatility and usability.

Consensus Mechanism:

Script employs a hybrid consensus mechanism combining PoS and BFT. PoS enables token holders to participate in block validation, while the BFT component enhances network security. This combination allows for fast transaction finality and high throughput.

Security Measures:

Script's BFT mechanism ensures Byzantine fault tolerance, making the network resilient against malicious attacks. Validators are selected based on their stake and reputation, discouraging malicious behavior.

The Script Blockchain's system architecture combines PoS and BFT consensus, optimizing video streaming delivery and incentivizing user participation. Through validator nodes, edge nodes, and smart contracts, the network provides a decentralized ecosystem for content distribution, benefiting both content creators and consumers while ensuring security and scalability.

Methodology

The codebase was audited using a filtered audit technique. Starting with the recon phase, a basic understanding was developed, and the security researchers worked on developing presumptions for the developed codebase and the relevant documentation/whitepaper. Furthermore, the audit moved on with the manual code reviews to find logical flaws in the codebase complemented with code optimizations, software, and security design patterns, code styles, best practices, and identifying false positives detected by automated analysis tools like semgrep.

AUDIT REPORT

Executive Summary

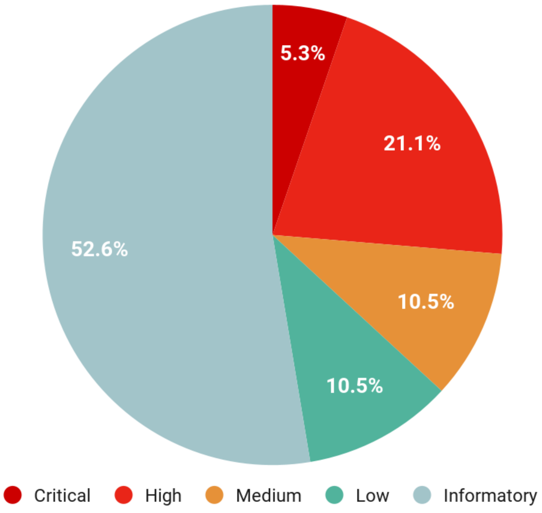
Our team performed a technique called *Filtered Audit*, where 4 individuals separately audited the Script-Tv Blockchain. After a thorough and rigorous manual testing process involving line-by-line code review for bugs, an automated tool-based review was carried out.

All the raised flags were manually reviewed and re-tested to identify any false positives.

The audit has resulted in the identification of (19) potential issues in the codebase.

Classification & Proportion of Vulnerabilities:

# of issues	Severity Level
1	**Critical**
4	**High**
2	**Medium**
2	Low
10	Informatory



Issue Status Descriptions

Acknowledged: The issue has been recognized and is under review. It indicates that the relevant team is aware of the problem and is actively considering the next steps or solutions.

Fixed: The issue has been addressed and resolved. Necessary actions or corrections have been implemented to eliminate the vulnerability or problem.

Closed: The issue is acknowledged as valid, but no corrective action has been taken by the development team. Despite multiple confrontations, the provided justification for inaction or the resolution offered is deemed unsatisfactory or weak.

SUMMARY OF FINDINGS

ID	Findings	Severity	Status
STB-001	Corrupted and Undefined Behavior in EVM Transactions	**Critical**	*Closed*
STB-002	Absence of Proposer/Guardian disincentivization exposes networks to multiple undefined behaviors.	**High**	*Closed*
STB-003	Potential theft of Funds Due to Misconfigured RPC	**High**	*Acknowledged*
STB-004	Unenforced Memory limits renders node config suboptimal	**High**	*Fixed*
STB-005	CORS Misconfiguration in HTTP RPC Interface leads to Data leaks	**High**	*Acknowledged*
STB-006	Inefficient Peer Filtering Leading to Network Overhead	**Medium**	*Closed*
STB-007	Missing Lock on Wallet In Send Transaction	**Medium**	*Fixed*
STB-008	Lack of Sanity Checks leads inefficient tx_stake_reward_distribution	Low	*Acknowledged*
STB-009	Incompatibility Issues with New Solidity Versions (0.8.20 and Above)	Low	*Closed*
STB-010	Inconsistent Max Round Logic in Guardian Engine	Informatory	*Fixed*
STB-011	Challenges in Running Tests Due to Parameter Value Changes	Informatory	*Closed*
STB-012	Unrestricted Validator Inclusion Beyond Limits	Informatory	*Acknowledged*

ID	Findings	Severity	Status
STB-013	Potential Mis-information relay due to Incorrect Metric Server domain	Informatory	*Fixed*
STB-014	Inconsistencies in Stake Deposit Values Between Code and Documentation	Informatory	*Fixed*
STB-015	Redundant Verification Checks in Transaction Processing	Informatory	*Closed*
STB-016	Potential performance optimization of `UpdateUnsafe` Function	Informatory	*Acknowledged*
STB-017	Unoptimized Blockchain Storage Due to Inactive Pruning	Informatory	*Acknowledged*
STB-018	Missing Vote Validation check	Informatory	*Fixed*
STB-019	Insecure Configuration of Message Signing and Verification in PubSub System	Informatory	*Fixed*

Detailed Findings

STB-001	Corrupted and Undefined Behavior in EVM Transactions		
Asset	/ledger/vm/evm.go		
Category	Implementation Error		
Status	Pending		
Rating	Severity:Critical	Likelihood:High	Impact:High

Description:

The EVM enables the functionality of executing arbitrary logic in the system in a decentralized fashion in a blockchain to increase its usability & allow general purpose interaction. We deployed a simple solidity contract built using 0.6.12, 0.7+ & 0.8+ with a simple function which increments a public variable. After sending the transaction on-chain the calldata was somehow corrupted during the execution. This corrupted data is visible both in the blockchain explorer and when queried from the CLI. The issue is consistently reproducible, but the underlying cause remains unidentified.

Impact:

1. **Data Integrity Loss**: Corrupted data undermines the blockchain's promise of a tamper-proof and transparent record, risking the validity and trustworthiness of stored information.
2. **Functional Disruption**: The intended behavior of the contract could be compromised, leading to potential malfunction or halted operations.

Proof of Concept(PoC)

```
pragma solidity 0.6.12;

contract PushZero_Test{
    uint256 public num;
    function set(uint256 _n) public{
       num = _n;
    }
}
```

After deployment we invoked set(uint256) function and passing 2 as a parameter:

- Contract Address:

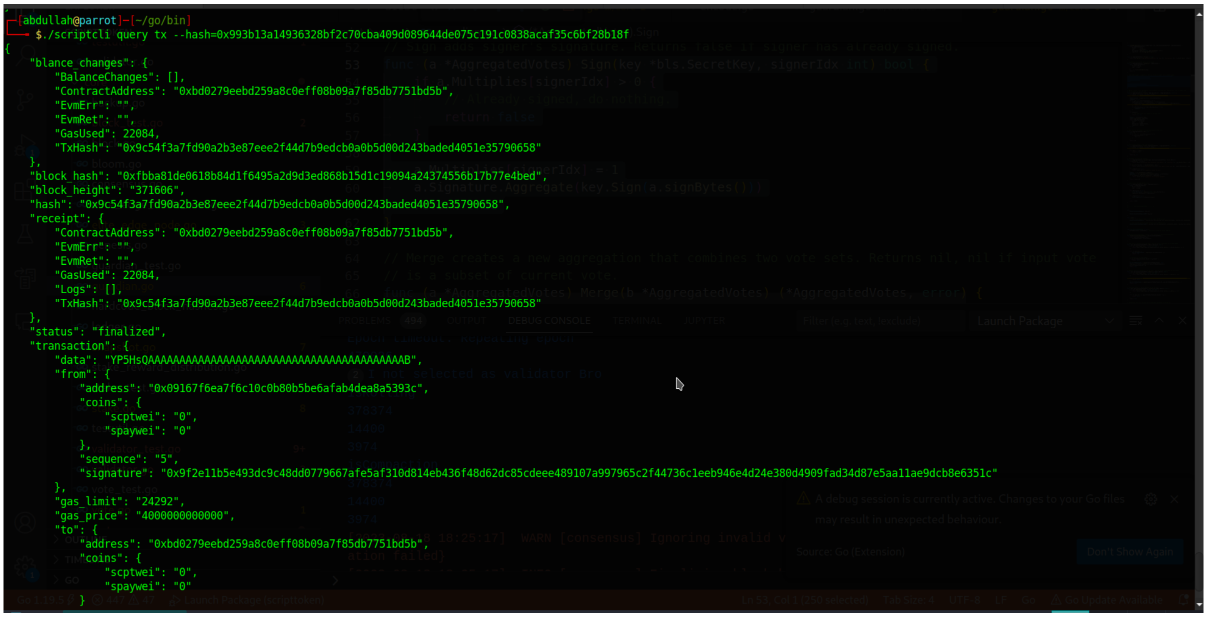
```
0xbd0279eebd259a8c0eff08b09a7f85db7751bd5b
```

- Expected Data

```
0x60fe47b100000000000000000000000000000000000000000000000000000000000
00002
```

- Received Data (on blockchain):

```
0x5950354873514141414141414141414141414141414141414141414141414141414
14141414141414141414141414142
// {YP5HsQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB} as shown below
```



```
┌─[abdullah@parrot]─[~/go/bin]
└──╼ $./scriptcli query tx --hash=0x993b13a14936328bf2c70cba409d089644de075c191c0838acaf35c6bf28b18f
{
    "blance_changes": {
        "BalanceChanges": [],
        "ContractAddress": "0xbd0279eebd259a8c0eff08b09a7f85db7751bd5b",
        "EvmErr": "",
        "EvmRet": "",
        "GasUsed": 22084,
        "TxHash": "0x9c54f3a7fd90a2b3e87eee2f44d7b9edcb0a0b5d00d243baded4051e35790658"
    },
    "block_hash": "0xfbba81de0618b84d1f6495a2d9d3ed868b15d1c19094a24374556b17b77e4bed",
    "block_height": "371606",
    "hash": "0x9c54f3a7fd90a2b3e87eee2f44d7b9edcb0a0b5d00d243baded4051e35790658",
    "receipt": {
        "ContractAddress": "0xbd0279eebd259a8c0eff08b09a7f85db7751bd5b",
        "EvmErr": "",
        "EvmRet": "",
        "GasUsed": 22084,
        "Logs": [],
        "TxHash": "0x9c54f3a7fd90a2b3e87eee2f44d7b9edcb0a0b5d00d243baded4051e35790658"
    },
    "status": "finalized",
    "transaction": {
        "data": "YP5HsQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB",
        "from": {
            "address": "0x09167f6ea7f6c10c0b80b5be6afab4dea8a5393c",
            "coins": {
                "scptwei": "0",
                "spaywei": "0"
            },
            "sequence": "5",
            "signature": "0x9f2e11b5e493dc9c48dd0779667afe5af310d814eb436f48d62dc85cdeee489107a997965c2f44736c1eeb946e4d24e380d4909fad34d87e5aa11ae9dcb8e6351c"
        },
        "gas_limit": "24292",
        "gas_price": "4000000000000",
        "to": {
            "address": "0xbd0279eebd259a8c0eff08b09a7f85db7751bd5b",
            "coins": {
                "scptwei": "0",
                "spaywei": "0"
            }
```

STB-002	Absence of Proposer/Guardian disincentivization mechanism exposes networks leads to block stalling.		
Asset	/consensus/engine.go:259-304		
Category	Logical Error, Consensus		
Status	Pending		
Rating	Severity:High	Likelihood:Medium	Impact:High

Description:

The consensus mainloop is a crucial mechanism of the blockchain. It runs indefinitely, and checks for multiple actions that node must take i.e Voting, Proposing new blocks, etc. to continue being a part of the blockchain. In the consensus mainloop of the network, there is an absence of disincentivization mechanisms for proposers (validators/guardians) who act maliciously. Though the slashing mechanism is commented out, it exposes the network to a range of vulnerabilities and undefined behaviors. In this specific scenario, there is a switch statement that runs if no block has been received for the epoch & all validators proceed to the next epoch. This will lead to block stalling as the proposer who was supposed to produce blocks is offline or maliciously avoiding building blocks, this in turn results in significant delays in transaction inclusion. Such delays can disrupt the normal functioning of the network and degrade user experience. This has been verified individually by observation while running the devnets locally & script testnet.

```
func (e *ConsensusEngine) mainLoop() {
        defer e.wg.Done()

        for {
                e.enterEpoch()
                e.propose()
        Epoch:
                for {
                        select {
                        case <-e.ctx.Done():
                                e.stopped = true
                                return
```

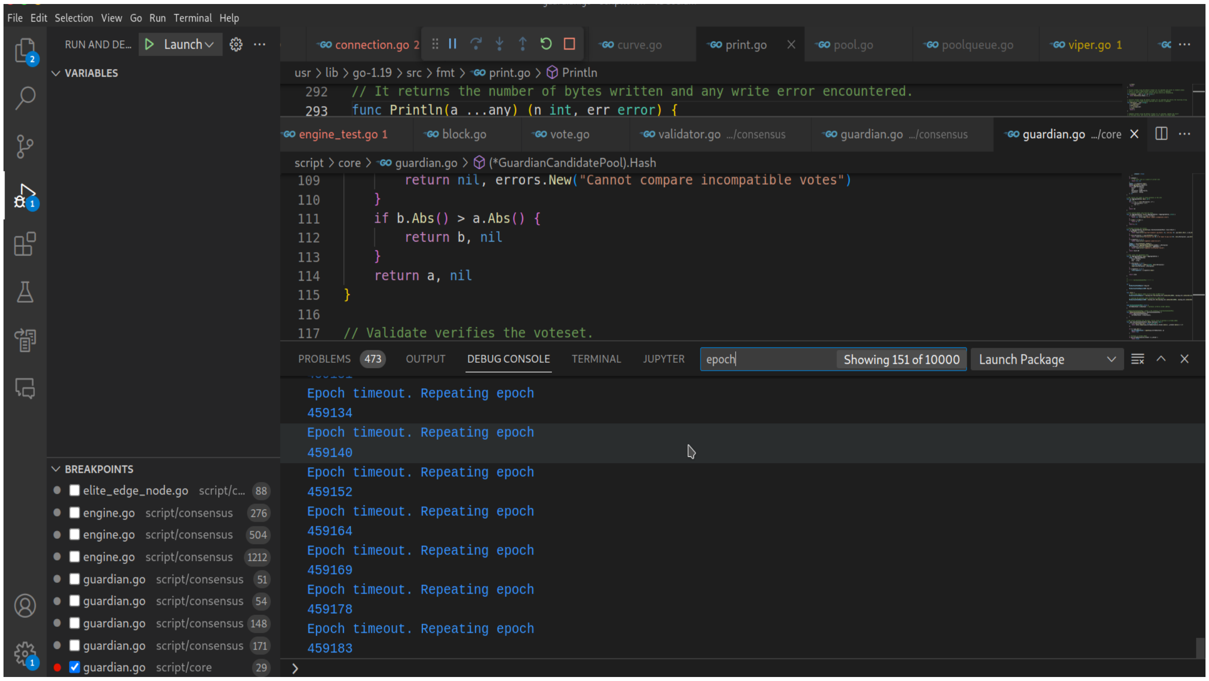
```
                    case msg := <-e.incoming:
                          endEpoch := e.processMessage(msg)
                          if endEpoch {
                                break Epoch
                          }

                  // - - -  Code Snipped - - -

                    case <-e.epochTimer.C:
                          e.logger.WithFields(log.Fields{"e.epoch":
e.GetEpoch()}).Debug("Epoch timeout. Repeating epoch")
                          e.vote()
                          break Epoch

                  // - - -  Code Snipped - - -
             }
       }
}
```

Screen Shot:



```
// It returns the number of bytes written and any write error encountered.
func Println(a ...any) (n int, err error) {
```

```
        return nil, errors.New("Cannot compare incompatible votes")
    }
    if b.Abs() > a.Abs() {
        return b, nil
    }
    return a, nil
}

// Validate verifies the voteset.
```

epoch

```
Epoch timeout. Repeating epoch
459134
Epoch timeout. Repeating epoch
459140
Epoch timeout. Repeating epoch
459152
Epoch timeout. Repeating epoch
459164
Epoch timeout. Repeating epoch
459169
Epoch timeout. Repeating epoch
459178
Epoch timeout. Repeating epoch
459183
```

BREAKPOINTS

- □ elite_edge_node.go script/c... 88
- □ engine.go script/consensus 276
- □ engine.go script/consensus 504
- □ engine.go script/consensus 1212
- □ guardian.go script/consensus 51
- □ guardian.go script/consensus 54
- □ guardian.go script/consensus 148
- □ guardian.go script/consensus 171
- ☑ guardian.go script/core 29

Recommendation:

Implement strict disincentivization mechanisms, such as slashing of stakes or banning of malicious nodes, to deter and penalize such behaviors.

STB-003	Potential theft of Funds Due to Misconfigured RPC		
Asset	Script/rpc/ Documentation		
Category	Misconfiguration		
Status	Pending		
Rating	Severity:High	Likelihood:Medium	Impact:High

Description:

The JSON-RPC service (disabled by default when running *./script start*) is an unauthenticated interface. If the JSON-RPC service is activated at port 16889, it exposes the RPC with critical functionalities , which can lead to stealing of Funds. Following are the calls which can be made on the /RPC.

- Send
- Newkey
- Listkeys
- Unlock_key
- Lockkey
- Iskey_Unlocked
- BroadCast_Raw_Transaction
- Broadcast_raw_transaction_async

Exploiting Scenario:

1. Attacker keeps sending the send transactions to the exposed RPC port.
2. Whenever the user will unlock the wallet the transaction will be executed and funds will be transferred.

Recommendation:

Explicitly asking users to expose RPC is not a good choice, RPC should only be exposed by experienced users. Moreover proper security guidelines should be published on script documentation to how to secure the RPC port. Use of Nginx HTTP basic Auth should be recommended to those who wish to enable RPC. There have been active exploitation regarding exposed RPC and there are bots which scan nodes for this type of attack.

Reference:

- https://geth.ethereum.org/docs/interacting-with-geth/rpc
- https://medium.com/coinmonks/securing-your-ethereum-nodes-from-hackers-8b7d5bac8986

STB-004	Unenforced Memory limits renders node config suboptimal		
Asset	script/mempool/mempool.go L-185		
Category	Misconfiguration		
Status	Pending		
Rating	Severity:High	Likelihood:Medium	Impact:High

Description:

The InsertTransaction function within the mempool.go file is tasked with adding new transactions to the mempool. A constant named MaxMempoolTxCount is defined, which seemingly serves as the upper boundary for the number of transactions the mempool can hold. However, upon inspection of the function, the code responsible for enforcing this limit is commented out. Consequently, there's nothing in place to stop the mempool from accepting an unbounded number of transactions.

An unchecked accumulation of transactions in the mempool can lead to several system challenges. Most critically, a continuous stream of transactions could cause the mempool to consume all available system memory. This unchecked consumption can potentially lead to memory leaks, buffer overflow, and other system-level anomalies that could disrupt the normal functioning of the node and affect the performance of the overall network.

Code Snippet

```
// if mp.size >= MaxMempoolTxCount {
//    logger.Debugf("Mempool is full")
//    return   errors.New("mempool   is   full,   please   submit   your
transaction again later")
// }
```

Recommendation:

- Uncomment the code that enforces the MaxMempoolTxCount limit.
- Ensure that the logic checks against the MaxMempoolTxCount are working effectively to prevent any addition of transactions past the defined threshold.

STB-005	CORS Misconfiguration in HTTP RPC Interface leads to Data leaks		
Asset	script/rpc/server.go		
Category	Implementation Error		
Status	Pending		
Rating	Severity:High	Likelihood:Medium	Impact:High

Description:
The HTTP RPC interface, in its current state, permits unrestricted cross-origin requests by having the Access-Control-Allow-Origin header configured to "*". Consequently, any domain can send requests to this interface. This behavior exposes the system to potential threats such as unauthorized access and data leaks. While the current exposure might not leak sensitive information, if in the future any sensitive endpoint becomes exposed, it could lead to unauthorized data access.

Code Snippet:

```
func corsMiddleware(handler http.Handler) http.Handler {
           return   http.HandlerFunc(func(w   http.ResponseWriter,   r
*http.Request) {
        w.Header().Set("Access-Control-Allow-Origin", "*")
        w.Header().Set("Access-Control-Allow-Headers", "*")
        if r.Method == "OPTIONS" {
            w.WriteHeader(http.StatusOK)
            return
        }
        handler.ServeHTTP(w, r)
    })
}
```

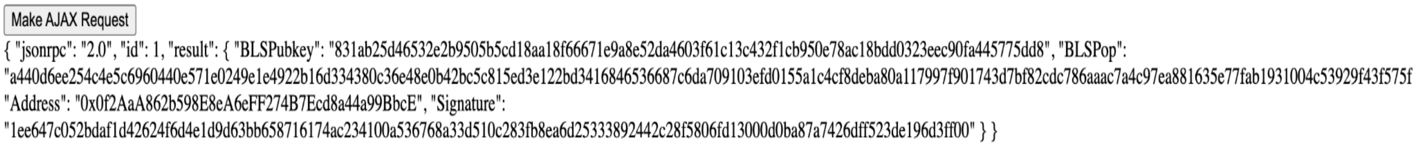
Proof of Concept (PoC):

Below is an HTML page designed to send an AJAX request to the RPC server:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>AJAX Request Example</title>
<script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
</head>
<body>
<button id="button">Make AJAX Request</button>
<div id="response"></div>
<script>
$(document).ready(function(){
  $("#button").click(function(){
    $.ajax({
      type: "POST",
      url: "http://{IP}:16888/rpc",
      data:
'{"jsonrpc":"2.0","method":"script.GetGuardianInfo","params":[]
,"id":1}',
      headers: {
        "Content-Type": "application/json; charset=utf-8",
        "Origin": "null"
      },
      success: function(data) {
        $("#response").text(JSON.stringify(data, null, 2));
      },
      error: function(xhr, status, error) {
        $("#response").text("Request failed with status: " +
status);
      },
      dataType: "json"
    });
  });
});
```

```
        </script>
        </body>
        </html>
```

Response



```
{ "jsonrpc": "2.0", "id": 1, "result": { "BLSPubkey": "831ab25d46532e2b9505b5cd18aa18f66671e9a8e52da4603f61c13c432f1cb950e78ac18bdd0323eec90fa445775dd8", "BLSPop":
"a440d6ee254c4e5c6960440e571e0249e1e4922b16d334380c36e48e0b42bc5c815ed3e122bd3416846536687c6da709103efd0155a1c4cf8deba80a117997f901743d7bf82cdc786aaac7a4c97ea881635e77fab1931004c53929f43f575f
"Address": "0x0f2AaA862b598E8eA6eFF274B7Ecd8a44a99BbcE", "Signature":
"1ee647c052bdaf1d42624f6d4e1d9d63bb658716174ac234100a536768a33d510c283fb8ea6d25333892442c28f5806fd13000d0ba87a7426dff523de196d3ff00" } }
```

Recommendation:

- Review and modify the corsMiddleware function to only allow specific trusted origins.
- Avoid using the wildcard (*) for the Access-Control-Allow-Origin header.
- While Running the node origin parameter to be passed as argument or in it should be set in config.go file

Reference:

- https://github.com/ethereum/go-ethereum/blob/master/docs/audits/2017-04-25_Geth-audit_Truesec.pdf
- https://talosintelligence.com/vulnerability_reports/TALOS-2017-0508

STB-006	Inefficient Peer Filtering Leading to Network Overhead		
Asset	script/p2pl/peer/peer_table.go script/p2pl/messenger/messenger.go		
Category	Misconfiguration		
Status	Pending		
Rating	Severity:Medium	Likelihood: High	Impact: low

Description:

The GetAllPeers function in the PeerTable class is intended to provide a list of peers with an option to exclude edge nodes via the skipEdgeNode parameter. it currently overlooks this detail. Given the primary role of edge nodes in the Script network—to finalize blocks and serve as caching and relay points—this oversight can lead to a surge in unnecessary resource consumption. As the network retrieves and processes unwanted edge node data, it inevitably impacts network efficiency and introduces unwarranted latency. Additionally, the system's message broadcasting functionality also does not respect the skipEdgeNode directive. Consequently, data intended solely for peers might inadvertently be transmitted to edge nodes.

Edge nodes are expected to be substantially more numerous than guardians and validators, primarily due to their lower stake amount requirement. This prevalence of edge nodes, combined with the aforementioned misconfiguration, can significantly increase the risk of latency and reduced efficiency in the network.

Code Snippet:

```
// GetAllPeers returns all the peers
func (pt *PeerTable) GetAllPeers(skipEdgeNode bool) *([]*Peer) {
      // TODO: support skipEdgeNode
      pt.mutex.Lock()
      defer pt.mutex.Unlock()

      ret := make([]*Peer, len(pt.peers))
      for i, p := range pt.peers {
            ret[i] = p . . .
```

Impact:

- As the network grows, the neglect of edge node filtering can contribute to potential network congestion. This is particularly concerning as unnecessary data which the network isn't intended to cater for becomes a significant portion of network traffic.
- If edge nodes receive requests they aren't supposed to handle, it might lead to unintended software behavior. This can pose a risk, especially if the software isn't equipped to handle such unexpected interactions..

Recommendation:

- Modify GetAllPeers to implement the skipEdgeNode parameter for filtering.

STB-007	Missing Lock on Wallet In Send Transaction		
Asset	cmd/scriptcli/rpc/tx.go		
Category	Logical Flaw		
Status	Pending		
Rating	Severity:Medium	Likelihood:Medium	Impact:Medium

Description:

In Script, users have the ability to expose RPC ports for specific functionalities. However, it's important to note that this practice is not recommended due to the potential risks associated with improper RPC port configuration. Such misconfiguration could lead to the complete loss of funds.

A behavior is observed when a user initiates a Send transaction via the RPC call on port 16889. The wallet associated with the transaction is not automatically locked after the RPC call is executed. This omission in locking the wallet can expose the funds to potential security vulnerabilities. It is worth mentioning that Issue "Potential theft of Funds Due to Misconfigured RPC" is different from this due to the fact that Disabling or securing RPC will keep this issue unresolved.

```
func (t *ScriptCliRPCService) Send(args *SendArgs, result *SendResult) (err error) {
if len(args.From) == 0 || len(args.To) == 0 {
return fmt.Errorf("The from and to address cannot be empty")
}
if args.From == args.To {
return fmt.Errorf("The from and to address cannot be identical")
}
.
.
.
. // Omitted for brevity
if !t.wallet.IsUnlocked(from) {
return fmt.Errorf("The from address %v has not been unlocked yet", from.Hex())
}
inputs := []types.TxInput{{
Address: from,
Coins: types.Coins{
SpayWei: new(big.Int).Add(spaywei, fee),
ScriptWei: scptwei,
},
```

Impact:
Without wallet locking, any subsequent unauthorized access to the wallet could lead to the unauthorized transfer of funds, compromising the user's assets.

Recommendation:
It is recommended to add ***“Defer wallet.lock(from)”*** after **‘t.wallet.IsUnlocked(from)’** so that the wallet is locked as the send transaction is successfully sent.

STB-08	Lack of Sanity Checks leads inefficient tx_stake_reward_distribution		
Asset	ledger/execution/tx_stake_reward_distribution.go		
Category	Logical Flaw , Commented Functions		
Status	Pending		
Rating	Severity:Low	Likelihood:High	Impact:Low

Description:

Within the "**ledger/execution/tx_stake_reward_distribution.go**" file, a security concern has been identified regarding the stake reward distribution transaction. In this transaction type, sanity checks are performed to ensure their integrity and validity. However, it has been observed that these sanity checks have been commented out in the code, rendering them inactive during transaction execution. Following checks are needed to be performed as per the commented code.

- Check if the A Validator is also a Guardian , reward split is not possible for that validator.
- Validate transactions related to guardian and prevent cases where a beneficiary is also a staker.
- Validate transactions related to elite edge nodes and prevent scenarios where a beneficiary is also a staker
- Identify and handle transactions with unrecognized or invalid purposes, providing an error message for such cases.

This omission can lead to the acceptance of invalid transactions, jeopardizing the overall security and reliability of the system.

```
func  (exec  *StakeRewardDistributionTxExecutor)  sanityCheck(chainID  string,  view
*st.StoreView, viewSel core.ViewSelector, transaction types.Tx) result.Result {
blockHeight  :=  view.Height()  +  1  //  the  view  points  to  the  parent  of  the  current
block



if tx.SplitBasisPoint > 1000 { // initially we only allow up to 10.00% reward split
return result.Error("Only allow at most 10.00%% reward split for the beneficiary for
now (i.e., SplitBasisPoint <= 1000)")
}

// stakeHolderAddress := tx.Holder.Address
//
// -- SNIP -- L:66 - L:100
// }

if minTxFee, success := sanityCheckForFee(tx.Fee, blockHeight); !success {
return  result.Error("Insufficient  fee.  Transaction  fee  needs  to  be  at  least  %v
SpayWei",
minTxFee).WithErrorCode(result.CodeInvalidFee)
```

Impact:

The absence of active sanity checks in the stake reward distribution transaction can lead to several consequences. As per the commented sanity check

- Beneficiary can be a staker address as of now, In the future if this code is enabled it will result in inconsistency or favorable & unfair advantage to other stakers.
- Future updates will have inconsistent state if enabled.

Recommendation:

If the commented Sanity Checks are not required they should be removed from code.

STB-09	Incompatibility Issues with New Solidity Versions (0.8.20 and Above)		
Asset	ledger/execution/tx_split_rule.go ledger/execution/tx_deposit_stake.go		
Category	Implementation error		
Status	Pending		
Rating	Severity:Low	Likelihood: Low	Impact: High

Description:

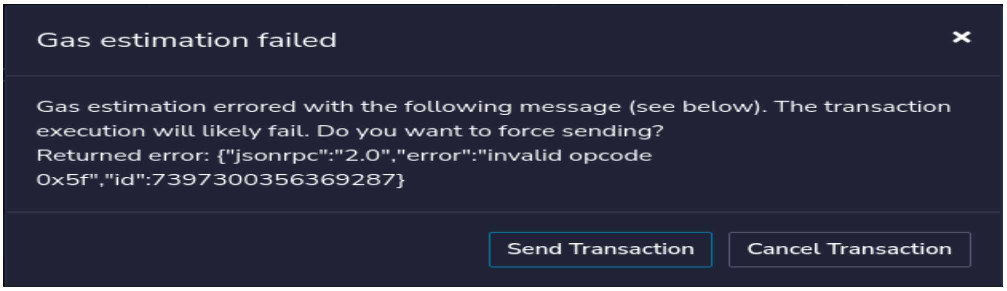
The network exhibits an incompatibility issue when interfacing with contracts compiled using Solidity versions 0.8.20 and subsequent releases. This incompatibility will always result in failure of contract deployment and loss in an amount of tokens paid for the transaction.

Code Snippet:

```
pragma solidity 0.8.20;

contract PushZero_Test{
    uint256 public num;
    function set(uint256 _n) public{
        num = _n;
    }
}
```

Screen Shot:



Gas estimation failed

Gas estimation errored with the following message (see below). The transaction execution will likely fail. Do you want to force sending?
Returned error: {"jsonrpc":"2.0","error":"invalid opcode 0x5f","id":7397300356369287}

Recommendation:

Introduce compatibility with contracts compiled with this version and above.

STB-010	Inconsistent Max Round Logic in GuardianEngine		
Asset	script/consensus/guardian.go script/consensus/elite_edge_node.go		
Category	Logical Error		
Status	Pending		
Rating	Severity:Info	Likelihood:High	Impact:Low

Description:

The GuardianEngine/EliteEdgeNodeEngine structure defines a MaxRound constant, set to 10 which is expected to Represent the maximum round number. However , in the StartNewRound function the logic Checks “if g.round < maxRound” , which means that the round number will never reach the Defined MaxRound value of 10, it will always end at 9.

Code Snippet:

```
maxRound = 10
func (g *GuardianEngine) StartNewBlock(block common.Hash) {
      g.mu.Lock()
      defer g.mu.Unlock()

      g.block = block
      g.nextVote = nil
      g.currVote = nil
      g.round = 1 . . .
}
func (g *GuardianEngine) StartNewRound() {
      g.mu.Lock()
      defer g.mu.Unlock()
      if g.round < maxRound {
            g.round++         . . .
}
```

Recommendation:

Adjust the logic to allow the round number to reach maxRound by checking if g.round <= maxRound instead.

STB-011	Challenges in Running Tests Due to Parameter Value Changes		
Asset	script/*test		
Category	Test Execution and Reliability Issues		
Status	Pending		
Rating	Severity:Info	Likelihood:Low	Impact:Medium

Description:

During our audit of the ScriptTV blockchain, we observed that there were significant challenges encountered when attempting to run tests. This was primarily attributed to changes in parameter values. The inability to effectively run the majority of the tests indicates a potential lack of rigorous testing in the project. If tests are not properly executed or if they fail due to configuration or parameter issues, it can result in undiscovered vulnerabilities or defects in the production environment.

Recommendation:

Conduct a comprehensive review of the test suite to identify which tests are failing and why.

If parameter values have changed, ensure that their corresponding test cases are updated to reflect these changes.

SPB-012	Unrestricted Validator Inclusion Beyond Limits		
Asset	/consensus/validator.go		
Category	Misconfiguration,Logical Flaw		
Status	Pending		
Rating	Severity:Info	Likelihood:Low	Impact:Low

Description:

ScriptTv Documentation suggests Maximum number of validators ranges between 10-20 which are selected by Script and it depends upon testnet participation. Participants that bought Script during initial sale are eligible after KYC by team.

As per code MaxNumber of validators defined are “31”.

```
const MaxValidatorCount int = 31
```

However, when a user deposits to stake as validator, there is no check for the MaxValidator in sanityCheck function hence after the max cap has reached as per the documentation , an address can be added into the ValidatorCandidatePool (vcp) resulting in increment of the validator set. These validators won’t participate in block proposing or voting, they will simply be available for relaying transactions in the network without any incentive.

```
if tx.Purpose == core.StakeForValidator {
sourceAccount.Balance = sourceAccount.Balance.Minus(stake)
stakeAmount := stake.ScriptWei
vcp := view.GetValidatorCandidatePool()
err := vcp.DepositStake(sourceAddress, holderAddress, stakeAmount, blockHeight)
if err != nil {
return common.Hash{}, result.Error("Failed to deposit stake, err: %v", err)
}
view.UpdateValidatorCandidatePool(vcp)
```

ledger/execution/tx_deposit_stake.go

```
func (vcp *ValidatorCandidatePool) sortCandidates() {
sort.Slice(vcp.SortedCandidates[:], func(i, j int) bool { // descending order in
(totalStake, holderAddress)
stakeCmp :=
vcp.SortedCandidates[i].TotalStake().Cmp(vcp.SortedCandidates[j].TotalStake())

if stakeCmp == 0 {
return strings.Compare(vcp.SortedCandidates[i].Holder.Hex(),

vcp.SortedCandidates[j].Holder.Hex()) >= 0
```

core/validator.go

```
func SelectTopStakeHoldersAsValidators(vcp *core.ValidatorCandidatePool)
*core.ValidatorSet {
maxNumValidators := MaxValidatorCount
topStakeHolders := vcp.GetTopStakeHolders(maxNumValidators)

valSet := core.NewValidatorSet()
for _, stakeHolder := range topStakeHolders {
valAddr := stakeHolder.Holder.Hex()
valStake := stakeHolder.TotalStake()
if valStake.Cmp(core.Zero) == 0 {
continue}
validator := core.NewValidator(valAddr, valStake)
valSet.AddValidator(validator)}

return valSet
}
```

consensus/validator.go

Recommendation:

It is recommended to enforce the limit upon deposit if the max cap for the validators is reached.

STB-013	Potential Mis-information relay due to Incorrect Metric Server domain		
Asset	mempool/tx_bookkeeper.go		
Category	Logical Flaw		
Status	Pending		
Rating	Severity:Info	Likelihood: -	Impact:Low

Description:
In the "config.go" file, a potential issue has been identified concerning the definition of the metric server. Within the code, `viper.SetDefault(CfgMetricsServer,"http://guardian-metrics.scripttoken.org/")` configures the metric server address. However, upon inspection, it has been observed that there is no deployed service on the provided server. As a consequence, this flaw opens up a potential attack vector where an attacker could claim ownership of the "scripttoken.org" domain and create a subdomain matching the server address. By doing so, the attacker can intercept the reported metrics, posing a significant security risk.

Impact:
An attacker who successfully controls the subdomain can intercept reported metrics from the vulnerable nodes & relay mis-information.

Recommendation:
To address this security concern and mitigate potential attacks, Ensure that the metric server specified in the configuration (http://guardian-metrics.scripttoken.org/) is operational and securely hosted. Deploy relevant services on the designated server to collect and manage metrics from the reporting nodes.

STB-014	Inconsistencies in Stake Deposit Values Between Code and Documentation		
Asset	/script/core/validator.go /script/core/guardian.go /script/core/elite_edge_node.go		
Category	Documentation Mismatch		
Status	Pending		
Rating	Severity: Info	Likelihood: -	Impact:low

Description:
There are discrepancies between the documented values for stake deposits and their actual values as set in the code. Specifically:

- MinValidatorStakeDeposit is commented to be "at least 2,000,000 Script", but its set value is 1,000,000 Script.
- MinValidatorStakeDeposit200K is commented to be "reduced to 200,000 Script", but its set value is also 1,000,000 Script.
- MinEliteEdgeNodeStakeDeposit is documented to be "at least 10,000 SPAY", but its set value is 5,000 SPAY.
- MaxEliteEdgeNodeStakeDeposit is documented to be "should not exceed 500,000 SPAY", but its set value is again 5,000 SPAY.

Code Snippets:
For MinValidatorStakeDeposit and MinValidatorStakeDeposit200K:

```
// Each stake deposit needs to be at least 2,000,000 Script
      MinValidatorStakeDeposit                                       =
new(big.Int).Mul(new(big.Int).SetUint64(1000000),
new(big.Int).SetUint64(1000000000000000000))

      // Minimum Validator stake deposit reduced to 200,000 Script
      MinValidatorStakeDeposit200K                                   =
new(big.Int).Mul(new(big.Int).SetUint64(1000000),
new(big.Int).SetUint64(1000000000000000000))
```

For MinEliteEdgeNodeStakeDeposit and MaxEliteEdgeNodeStakeDeposit:

```
// Each elite edge node stake deposit needs to be at least 10,000
SPAY
      MinEliteEdgeNodeStakeDeposit                               =
new(big.Int).Mul(new(big.Int).SetUint64(5000),
new(big.Int).SetUint64(1e18))

      // Each elite edge node stake deposit should not exceed 500,000
SPAY
      MaxEliteEdgeNodeStakeDeposit                               =
new(big.Int).Mul(new(big.Int).SetUint64(5000),
new(big.Int).SetUint64(1e18))
```

```
var (
   MinGuardianStakeDeposit *big.Int

   MinGuardianStakeDeposit1000 *big.Int
)

func init() {
   // Each stake deposit needs to be at least 10,000 Script
                                 MinGuardianStakeDeposit           =
new(big.Int).Mul(new(big.Int).SetUint64(10000),
new(big.Int).SetUint64(1e18))

   // Lowering the guardian stake threshold to 1,000 Script
                              MinGuardianStakeDeposit1000          =
new(big.Int).Mul(new(big.Int).SetUint64(10000),
new(big.Int).SetUint64(1e18))

}
```

Recommendation:

- Review and align the code values with the comments or vice versa to ensure consistency between the documentation and the actual code implementation.
- Regularly update code comments/documentation when making changes to maintain clarity and avoid confusion for developers.

STB-015	Redundant Verification Checks in Transaction Processing		
Asset	/script/ledger/execution/tx_split_rule.go /script/ledger/execution/tx_deposit_stake.go		
Category	Code Redundancy		
Status	Pending		
Rating	Severity:Info	Likelihood:Medium	Impact: low

Description:

There are redundant verification checks in multiple transaction processing functions:

- In tx_split_rule.go, the SplitRuleTxExecutor sanity check function contains an unnecessary verification for the percentage value. As percentage is an uint (unsigned integer), it cannot be negative, making the check if percentage < 0 redundant.
- In tx_deposit_stake.go, a check exists to verify if the stake is valid and non-negative. If the internal logic for IsValid() inherently confirms non-negativity or vice-versa, then the checks become extraneous.

```
if percentage < 0 {
	return result.Error("Percentage needs to be positive")
}
```

```
if !stake.IsValid() || !stake.IsNonnegative() {
	return result.Error("Invalid stake for stake deposit!").
		WithErrorCode(result.CodeInvalidStake)
}
```

Recommendation:

- Review and remove the redundant check if percentage < 0 from the SplitRuleTxExecutor function in tx_split_rule.go.
- IsValid() already checks for IsNonnegative() tx_deposit_stake.go, therefore IsNonnegative() can be safely removed.

STB-016	Potential performance optimization of `UpdateUnsafe` Function		
Asset	script/mempool/mempool.go		
Category	Logical Error		
Status	Pending		
Rating	Severity:Info	Likelihood:low	Impact:low

Description:

The UpdateUnsafe function in the mempool module can be optimized to reduce computational overhead and improve transaction processing efficiency. The function is frequently called. Optimizing this function can enhance transaction processing speed and overall system efficiency.

Code Snippet

```
// UpdateUnsafe is the non-locking version of Update. Caller must
call Mempool.Lock() before
// calling this method.
func (mp *Mempool) UpdateUnsafe(committedRawTxs []common.Bytes) {
	start := time.Now()
	mp.removeTxs(committedRawTxs)
	removeCommittedTxTime := time.Since(start)

	// Remove Txs that have become obsolete.
	start = time.Now()
	count := 0
	invalidTxs := []common.Bytes{}
	txGroups := mp.candidateTxs.ElementList()
	for _, txGroupEl := range *txGroups {
		txGroup := txGroupEl.(*mempoolTransactionGroup)
		txs := txGroup.txs.ElementList()
		for _, txEl := range *txs {
			count++
            mempoolTx := txEl.(*mempoolTransaction)



            // Check for outdated txs
            txHash :=
getTransactionHash(mempoolTx.rawTransaction)
            _, exists := mp.txBookeepper.getStatus(txHash)
            if !exists {
                // Tx has been removed from bookkeeper due to
timeout
                invalidTxs = append(invalidTxs,
mempoolTx.rawTransaction)
                continue
            }

checkTxRes := mp.ledger.ScreenTxUnsafe(mempoolTx.rawTransaction)
if !checkTxRes.IsOK() {
invalidTxs = append(invalidTxs, mempoolTx.rawTransaction)
mp.txBookeepper.markAbandoned(mempoolTx.rawTransaction)
            }
        }
    }
screenTxTime := time.Since(start)

start = time.Now()
mp.removeTxs(invalidTxs)
removeInvalidTxTime := time.Since(start)

    logger.Debugf("UpdateUnsafe: %d tx screened in %v,
removeCommittedTxTime = %v, removed %d obsolete Txs in %v: %v,",
count, screenTxTime, removeCommittedTxTime, len(invalidTxs),
removeInvalidTxTime, invalidTxs)
}
```

1. Batch Removal of Transactions:

- **Issue:** The function calls removeTxs twice, leading to redundant iterations.

- **Recommendation:** Combine committedRawTxs and invalidTxs and call removeTxs once.

2. Efficient Data Structures:

- **Issue:** Slices are used for storing transactions, leading to O(n) lookups/removals.
- **Recommendation:** Consider hashmaps for O(1) operations

Performance Impact:

Current complexities can reach O(n^2) in certain operations. Proposed optimizations can reduce most operations to O(n).

STB-017	Unoptimized Blockchain Storage Due to Inactive Pruning		
Asset	ledger/ledger.go consensus/engine.go		
Category	Unimplemented Functionality		
Status	Pending		
Rating	Severity:Info	Likelihood:Low	Impact:Medium

Description:

In the assessed codebase, blockchain pruning functionality has been deliberately disabled. Blockchain pruning is a technique used to optimize storage & efficient syncing by selectively removing older data while preserving data integrity. In this case, the absence of blockchain pruning suggests that the system retains the entire transaction history, which might have implications for storage, performance, and scalability.

```
func (e *ConsensusEngine) pruneState(currentBlockHeight uint64) {
// Permanently disabled
return
.
.
.
}
```

consensus/engine.go

```
func (ledger *Ledger) PruneState(targetEndHeight uint64) error {
// Permanently disabled
return nil

.
.
.
}
```

ledger/ledger.go

Impact:

The decision to disable blockchain pruning can have several notable impacts on the system

Increased Storage Requirements: Storing the entire transaction history without pruning can lead to significant increases in storage demands over time. This could potentially strain system resources and lead to storage-related issues, particularly as the blockchain dataset grows.

Synchronization Delays: New nodes or participants entering the network will experience longer synchronization times since they need to download and verify the entire historical blockchain. This can deter potential users from engaging with the system.

Recommendation:

It is recommended to enable pruning as it optimizes the storage and reduces the overhead which is a general practice among many different blockchains. If it is not required in the current or future implementations, then all the relevant code should be removed from the codebase.

STB-018	Missing Vote Validation check		
Asset	consensus/guardian.go		
Category	Commented Code, Unimplemented Functionality		
Status	Pending		
Rating	Severity:Info	Likelihood: -	Impact: -

Description:

```
func(g *GuardianEngine) validateVote(vote *core.AggregatedVotes) (res bool) {
...

if !g.checkMultipliesForRound(vote, g.round) {
g.logger.WithFields(log.Fields{
"local.block": g.block.Hex(),
"local.round": g.round,
"vote.block": vote.Block.Hex(),
"vote.Mutiplies": vote.Multiplies,
"vote.gcp": vote.Gcp.Hex(),
"local.gcp": g.gcpHash.Hex(),
}).Debug("Ignoring guardian vote: multiples exceed limit for round")
return
```

The check **checkMultipliesForRound(vote, g.round)** will always return true as the function is commented out hence this check will not run, as true is being returned each time.

```
func (g *GuardianEngine) checkMultipliesForRound(vote *core.AggregatedVotes, k
uint32) bool {
// for _, m := range vote.Multiplies {
// if m > g.maxMultiply(k) {
// return false
// }
// }
return true
}
```

Recommendation:

It is recommended to implement the functionality for multiplies_check to maintain the integrity of the EEN Votes.

STB-019	Insecure Configuration of Message Signing and Verification in PubSub System		
Asset	p2pl/messenger/messenger.go		
Category	Misconfiguration		
Status	Pending		
Rating	Severity:Info	Likelihood:High	Impact:Low

Description:

The CreateMessenge configures the PubSub subsystem in a manner that disables two critical security features related to message signing and its verification

```
psOpts := []ps.Option{
      ps.WithMessageSigning(false),
      ps.WithStrictSignatureVerification(false),
}
pubsub, err := ps.NewGossipSub(ctx, host, psOpts...)
```

By turning off these security measures:

- The system does not ensure that each message is signed by the claimed sender (ps.WithMessageSigning(false)).
- The system does not require a valid signature matching the sender's public key (ps.WithStrictSignatureVerification(false))

Impact:

- **Loss of Trust:** Peers in the network cannot trust the authenticity of received messages.
- **Spread of False Information:** Peers might act based on false data, assuming it's from a trusted source.
- **Potential System Disruption:** Malicious nodes can exploit this insecure configuration to disrupt system operations.

Recommendation:

Enable message signing & strict signature verification by setting the variable to true

Miscellaneous Issues

TODO's

The presence of various TODOs throughout the repository are listed below. The developers should review these to address critical issues, optimizations, or clarifications.

```
common/metrics/memory.md:
   76:TODO 0.808 kB resident per counter.**
  152:TODO 15.084 resident per histogram.**
consensus/validator.go:
  105: // TODO: replace with more secure randomness.
core/stake.go:
  96: // TODO: Should rename StakeHolder to StakeDelegate
crypto/bn256/google/bn256.go:
  30  // BUG(agl): this implementation is not constant time.
  31: // TODO(agl): keep GF(p²) elements in Mongomery form.
crypto/ecies/README:
  56: TODO
crypto/secp256k1/curve.go:
  115: //TODO: double check if the function is okay
crypto/secp256k1/secp256_test.go:
  160: // TODO: why do we flip around the recovery id?
crypto/secp256k1/libsecp256k1/src/tests_exhaustive.c:
   422:  /*  TODO  set  z  =  1,  then  do  num_tests  runs  with  random  z
values */
crypto/secp256k1/libsecp256k1/src/java/org/bitcoin/NativeSecp256k1.j
ava:
  152: //TODO add a 'compressed' arg
crypto/secp256k1/libsecp256k1/src/java/org/bitcoin/NativeSecp256k1Te
st.java:
  14: //TODO improve comments/add more tests
crypto/sha3/sha3.go:
  142: todo := d.rate - len(d.buf)
dispatcher/dispatcher.go:
   136: // TODO: for 1.3.0 upgrade only, delete it after the upgrade
completed
ledger/ledger.go:
    743: // TODO: potentially O(m*n) runtime complexity, but the
number of stakes
ledger/execution/executils.go:
  16: // TODO: need to implement the following two functions
ledger/execution/tx_coinbase.go:
   240: // TODO - Need to confirm: should we get the VCP from the
current view?
   306: // TODO - Need to confirm: should we get the VCP from the
current view?
ledger/execution/tx_release_fund.go:
    82: // TODO: revisit whether we should panic or just log the
error.
ledger/execution/tx_slash.go:
    118: // TODO: We should transfer the collateral to a special
address, e.g.
  144: // TODO: need proper logging and error handling here.
ledger/execution/tx_smart_contract.go:
  191: // TODO: Add tx receipt: status and events
ledger/types/account.go:
   18: // TODO: replace the slice with map
   320: // TODO: The proof can be simplied to only contain the signed
. . .
ledger/types/eth_tx_utils.go:
  141:// TODO: copy pointed-to address
ledger/types/reserved_fund.go:
  72: // TODO: this implementation is not very efficient
ledger/types/tx.go:
  620: // TODO: remove chainID from all Tx sign bytes.
  653: // TODO: verify it is signed
ledger/vm/asm/compiler.go:
  174: // TODO figure out how to return the error properly
node/node.go:
  73: // TODO: check if this is a guardian node
p2p/connection/auth.go:
  430: // TODO: fewer pointless conversions
   455: // TODO: replace this when Msg contains the protocol type
code.
  516: // TODO: check overflow
p2p/connection/connection.go:
  430: // TODO: replace with lightweight Reset()
  521: // TODO: error handling
p2p/messenger/addr_book_test.go:
  98:// TODO: do more testing :)
p2p/messenger/addr_book.go:
  594: // TODO: Move to old probabilistically.
  811: // TODO refactor to return error?
p2p/messenger/discovery.go:
   172:// TODO: may need to stop peer regardless of the remote
address . . .
p2p/netutil/net_addr_test.go:
  73: // TODO add more test cases
  97: // TODO add more test cases
p2p/netutil/net_addr.go:
   161: // TODO(oga) bitcoind doesn't include RFC3849 here, but
should we?
p2p/netutil/upnp.go:
    8: // BUG(jae): TODO: use syscalls to get actual ourIP.
  350: // TODO: check response to see if the port was forwarded
  376: // TODO: check response to see if the port was deleted
p2pl/messenger/messenger.go:
  572:// TODO: support skipEdgeNode
p2pl/peer/peer_table.go:
  139: // TODO: support skipEdgeNode
  243: // TODO: support skipEdgeNode
p2pl/transport/stream.go:
   74: // TODO: Read implements the io.Reader
  105: // TODO: figure out close vs reset
  112: // TODO: figure out close vs reset
p2pl/transport/buffer/recv_buffer.go:
   142: // TODO: protection for attacks, e.g. send a very large
message to peers
rlp/decode.go:
  129: // TODO: this could use a Stream from a pool.
  137: // TODO: this could use a Stream from a pool.
rlp/encode.go:
  194: // TODO: encode to w.str directly
  398: // TODO: encode int to w.str directly
rpc/query.go:
  554: // TODO: use ID() instead after 1.3.0 upgrade
rpc/lib/rpc-codec/jsonrpc2/context_test.go:
  74: //- TODO batch call all
rpc/lib/rpc-codec/jsonrpc2/protocol_test.go:
  633: // TODO test for rpc.ErrShutdown && io.ErrUnexpectedEOF
snapshot/snapshot_import.go:
   1001: // TODO: this would lead to mismatch between the proven and
retrieved .
store/treestore/treestore.go:
  96: // TODO: find alternative way without traversal
wallet/coldwallet/usb_hub.go:
  67: // TODO(karalabe): remove if hotplug lands on Windows
wallet/coldwallet/hid/hidapi/libusb/hid.c:
   205: /*TODO: Implement this function on hidapi/libusb.. */
  1495:/* TODO: Do we need this? */
wallet/coldwallet/hid/hidapi/mac/hid.c:
   609: /*TODO: CFRunLoopGetCurrent()*/
  1025: /* TODO: */
  1033: /* TODO: */
wallet/coldwallet/hid/hidapi/windows/hid.c:
  400: /* TODO: Determine Size */
   504: /* TODO: Merge this with the Linux version. This function is
. . .
   520: /* TODO: Merge this functions with the Linux version. This
function . ..
wallet/coldwallet/hid/libusb/libusb/os/haiku_usb_backend.cpp:
  138: // TODO Handle this error
wallet/coldwallet/hid/libusb/libusb/os/sunos_usb.c:
  1276:/* TODO */

wallet/coldwallet/hid/libusb/libusb/os/threads_posix.c:
  77: /* TODO: NetBSD thread ID support */
wallet/coldwallet/hid/libusb/libusb/os/threads_windows.c:
  69: (ToDo: check that abandoned is ok)
  93:(ToDo: check that abandoned is ok)
wallet/coldwallet/hid/libusb/libusb/os/windows_usbdk.c:
  737: //TODO: Check whether we can support this in UsbDk
wallet/coldwallet/hid/libusb/libusb/os/windows_winusb.c:
   476: // TODO: (post hotplug): try without sanitizing
  2568: // TODO: can we move this whole business into the K/0 DLL?
   2934: // TODO: (post hotplug): see if we can force eject the
device and . . .
```

```
wallet/coldwallet/hid/libusb/libusb/os/windows_winusb.h:
  135: // TODO (v2+): move hid desc to libusb.h?
wallet/coldwallet/keystore/trezor.go:
  309: //TODO
wallet/coldwallet/keystore/trezor/protobuf.go:
  421: //TODO
  423: //TODO
  460: //TODO
  462: //TODO
wallet/coldwallet/keystore/trezor/transport.go:
  80://TODO: should be Model
wallet/softwallet/keystore/ks_encrypted.go:
   348: // TODO: can we do without this when unmarshalling dynamic
JSON?
```

Unused Functions

Our code review process identified certain functions that are either unused or have been left as stubs without a proper implementation. Addressing these areas can lead to cleaner, more maintainable code, and in some cases, enhanced efficiency and robustness

Note: This is not an exhaustive list. The team should perform a thorough review to identify and remove any other unused functions.

- **Location:**
  `/script/store/database/backend/leveldb.go`
  Function:

  ```
  func (dt *table) Close() {
     // Do nothing; don't close the underlying DB.
  }
  ```

- **Location:**
  `p2pl/transport/stream.go`
  Functions:

  ```
  // SetDeadline is a stub
  func (s *BufferedStream) SetDeadline(t time.Time) error {
        return nil
  }

  // SetReadDeadline is a stub
  func (s *BufferedStream) SetReadDeadline(t time.Time) error {
        return nil
  }

  // SetWriteDeadline is a stub
  func (s *BufferedStream) SetWriteDeadline(t time.Time) error {
        return nil
  }
  ```

- **Location:**
  `/script/p2pl/messenger/notify.go`
  Functions:

  ```
  func (p *PeerNotif) OpenedStream(n network.Network, s
  ```

```
network.Stream) {
      // peerID := s.Conn().RemotePeer()
      // logger.Infof("OpenedStream %v", peerID)
}

func (p *PeerNotif) ClosedStream(n network.Network, s
network.Stream) {
      // peerID := s.Conn().RemotePeer()
      // logger.Infof("ClosedStream %v", peerID)
}
func (p *PeerNotif) Listen(n network.Network, _ ma.Multiaddr) {
}

func (p *PeerNotif) ListenClose(n network.Network, _
ma.Multiaddr) {
}
```

Unused Variables

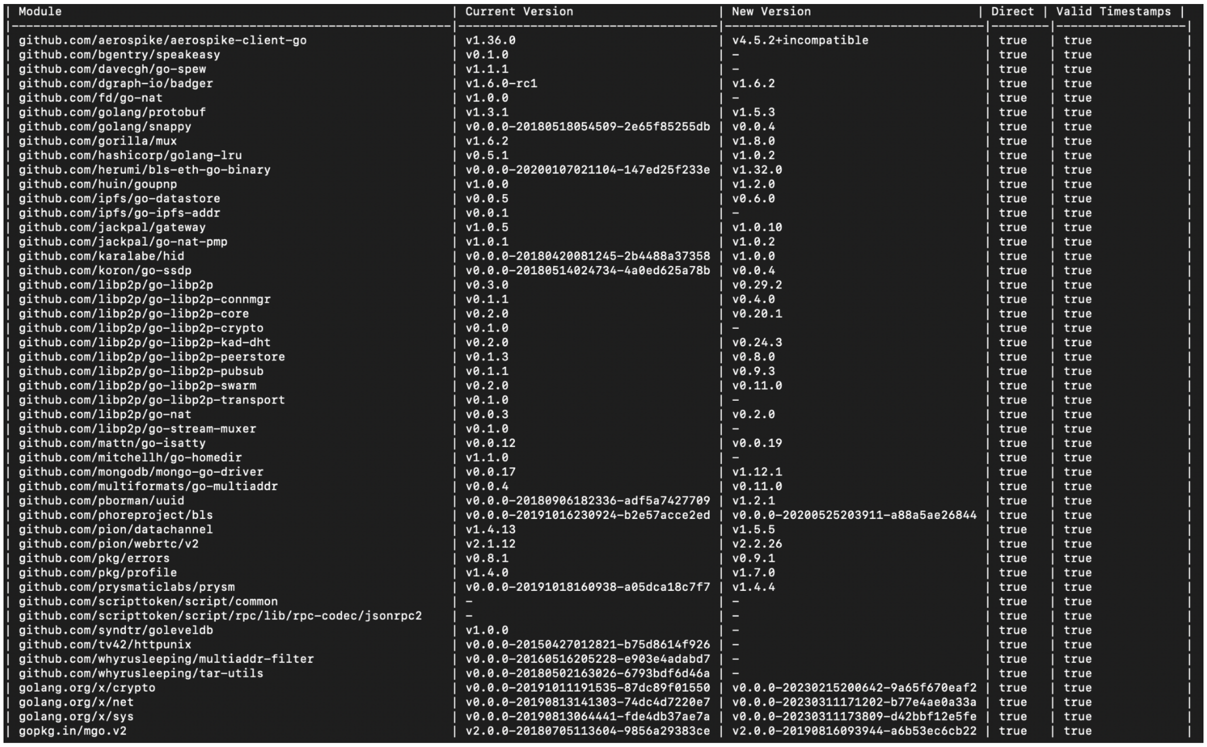
Unused variables in the codebase can lead to confusion for developers, increase code size, and mask potential issues where a variable was supposed to be used but was accidentally overlooked. There are a lot of instances in the codebase where unused variables are presented. It is recommended to remove them to make the codebase cleaner.

Commented Code

Commented-out code sections can be misleading, as they often represent outdated logic or approaches that were discarded. They clutter the codebase and can make it harder for developers to understand the current logic. It's generally a good practice to rely on version control systems (like Git) to keep track of older code versions, rather than leaving them commented out in the codebase.

Outdated Dependencies

The following table lists outdated packages identified by the `go mod outdated` tool. It is recommended to review and update these packages to their respective new versions to ensure optimal performance and security.



```
| Module                                                     | Current Version                    | New Version                        | Direct | Valid Timestamps |
|------------------------------------------------------------|------------------------------------|------------------------------------|--------|------------------|
| github.com/aerospike/aerospike-client-go                   | v1.36.0                            | v4.5.2+incompatible                | true   | true             |
| github.com/bgentry/speakeasy                               | v0.1.0                             | -                                  | true   | true             |
| github.com/davecgh/go-spew                                 | v1.1.1                             | -                                  | true   | true             |
| github.com/dgraph-io/badger                                | v1.6.0-rc1                         | v1.6.2                             | true   | true             |
| github.com/fd/go-nat                                       | v1.0.0                             | -                                  | true   | true             |
| github.com/golang/protobuf                                 | v1.3.1                             | v1.5.3                             | true   | true             |
| github.com/golang/snappy                                   | v0.0.0-20180518054509-2e65f85255db | v0.0.4                             | true   | true             |
| github.com/gorilla/mux                                     | v1.6.2                             | v1.8.0                             | true   | true             |
| github.com/hashicorp/golang-lru                            | v0.5.1                             | v1.0.2                             | true   | true             |
| github.com/herumi/bls-eth-go-binary                        | v0.0.0-20200107021104-147ed25f233e | v1.32.0                            | true   | true             |
| github.com/huin/goupnp                                     | v1.0.0                             | v1.2.0                             | true   | true             |
| github.com/ipfs/go-datastore                               | v0.0.5                             | v0.6.0                             | true   | true             |
| github.com/ipfs/go-ipfs-addr                               | v0.0.1                             | -                                  | true   | true             |
| github.com/jackpal/gateway                                 | v1.0.5                             | v1.0.10                            | true   | true             |
| github.com/jackpal/go-nat-pmp                              | v1.0.1                             | v1.0.2                             | true   | true             |
| github.com/karalabe/hid                                    | v0.0.0-20180420081245-2b4488a37358 | v1.0.0                             | true   | true             |
| github.com/koron/go-ssdp                                   | v0.0.0-20180514024734-4a0ed625a78b | v0.0.4                             | true   | true             |
| github.com/libp2p/go-libp2p                                | v0.3.0                             | v0.29.2                            | true   | true             |
| github.com/libp2p/go-libp2p-connmgr                        | v0.1.1                             | v0.4.0                             | true   | true             |
| github.com/libp2p/go-libp2p-core                           | v0.2.0                             | v0.20.1                            | true   | true             |
| github.com/libp2p/go-libp2p-crypto                         | v0.1.0                             | -                                  | true   | true             |
| github.com/libp2p/go-libp2p-kad-dht                        | v0.2.0                             | v0.24.3                            | true   | true             |
| github.com/libp2p/go-libp2p-peerstore                      | v0.1.3                             | v0.8.0                             | true   | true             |
| github.com/libp2p/go-libp2p-pubsub                         | v0.1.1                             | v0.9.3                             | true   | true             |
| github.com/libp2p/go-libp2p-swarm                          | v0.2.0                             | v0.11.0                            | true   | true             |
| github.com/libp2p/go-libp2p-transport                      | v0.1.0                             | -                                  | true   | true             |
| github.com/libp2p/go-nat                                   | v0.0.3                             | v0.2.0                             | true   | true             |
| github.com/libp2p/go-stream-muxer                          | v0.1.0                             | -                                  | true   | true             |
| github.com/mattn/go-isatty                                 | v0.0.12                            | v0.0.19                            | true   | true             |
| github.com/mitchellh/go-homedir                            | v1.1.0                             | -                                  | true   | true             |
| github.com/mongodb/mongo-go-driver                         | v0.0.17                            | v1.12.1                            | true   | true             |
| github.com/multiformats/go-multiaddr                       | v0.0.4                             | v0.11.0                            | true   | true             |
| github.com/pborman/uuid                                    | v0.0.0-20180906182336-adf5a7427709 | v1.2.1                             | true   | true             |
| github.com/phoreproject/bls                                | v0.0.0-20191016230924-b2e57acce2ed | v0.0.0-20200525203911-a88a5ae26844 | true   | true             |
| github.com/pion/datachannel                                | v1.4.13                            | v1.5.5                             | true   | true             |
| github.com/pion/webrtc/v2                                  | v2.1.12                            | v2.2.26                            | true   | true             |
| github.com/pkg/errors                                      | v0.8.1                             | v0.9.1                             | true   | true             |
| github.com/pkg/profile                                     | v1.4.0                             | v1.7.0                             | true   | true             |
| github.com/prysmaticlabs/prysm                             | v0.0.0-20191018160938-a05dca18c7f7 | v1.4.4                             | true   | true             |
| github.com/scripttoken/script/common                       | -                                  | -                                  | true   | true             |
| github.com/scripttoken/script/rpc/lib/rpc-codec/jsonrpc2   | -                                  | -                                  | true   | true             |
| github.com/syndtr/goleveldb                                | v1.0.0                             | -                                  | true   | true             |
| github.com/tv42/httpunix                                   | v0.0.0-20150427012821-b75d8614f926 | -                                  | true   | true             |
| github.com/whyrusleeping/multiaddr-filter                  | v0.0.0-20160516205228-e903e4adabd7 | -                                  | true   | true             |
| github.com/whyrusleeping/tar-utils                         | v0.0.0-20180502163026-6793bdf6d46a | -                                  | true   | true             |
| golang.org/x/crypto                                        | v0.0.0-20191011191535-87dc89f01550 | v0.0.0-20230215200642-9a65f670eaf2 | true   | true             |
| golang.org/x/net                                           | v0.0.0-20190813141303-74dc4d7220e7 | v0.0.0-20230311171202-b77e4ae0a33a | true   | true             |
| golang.org/x/sys                                           | v0.0.0-20190813064441-fde4db37ae7a | v0.0.0-20230311173809-d42bbf12e5fe | true   | true             |
| gopkg.in/mgo.v2                                            | v2.0.0-20180705113604-9856a29383ce | v2.0.0-20190816093944-a6b53ec6cb22 | true   | true             |
```

DISCLAIMER

The following blockchain audit report has been carefully carried out based on the data provided by the client. It aims to identify potential vulnerabilities in compliance with globally recognized best practices regarding cybersecurity in blockchain technology. The conclusions drawn herein relate only to the security aspects of the audited blockchain and are not a reflection on the project or its team.

This report does not vouch for the absolute security of the blockchain, nor does it touch upon the economic prospects of any related tokens or the broader project. The realm of blockchain technology, being a subset of decentralized finance, inherently carries technical risks and uncertainties. No part of this report should be viewed as a guarantee or endorsement for third parties, particularly concerning the infallibility of the audited code, its associated business model, or its legal standing.

Any third party should be cautious and refrain from using this report as the sole determinant in decision-making processes, especially when it comes to buying or selling tokens, products, or services. To be clear, this report is not an investment guide, and its content is not to be mistaken as investment advice.

The blockchain's foundation – its platform, language, and related software – might have inherent vulnerabilities that may not be captured in this audit. Our review is strictly confined to the blockchain code as presented and highlighted within this report. The blockchain language, which remains in its evolving phase, may have undiscovered risks.

The thoroughness of this audit notwithstanding, it shouldn't be the only tool you rely upon. It is imperative to understand that no single audit can conclusively validate the functionality and security of a blockchain. Thus, for a comprehensive security assessment, we advise seeking multiple independent audits and initiating a public bug bounty program.