InitVerse

BLOCKCHAIN AUDIT REPORT

March 2025

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by InitVerse to review Blockchain implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

Likelihood \ IMPACT	Informational	Low	Medium	High
High	Informational	Medium	High	Critical
Medium	Informational	Low	Medium	High
Low	Informational	Low	Low	Medium

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment

and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given Blockchain with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of Blockchains from the perspective of proven programming practices.

Category	Assessment Item
P2P Communication Security	Connection Number Occupation Audit
	Eclipse Attack
	Packet Size Limit
	Node Communication Protocol Security
RPC Interface Security	RPC Sensitive Interface Permissions
	Traditional Web Security
	RPC Interface Security
Consensus Mechanism Security	Design Of Consensus Mechanism
	Implementation Of Consensus Verification
	Incentive Mechanism Audit
Transaction processing Security	Transaction Signature Logic
	Transaction Verification Logic
	Transaction Processing Logic
	Transaction Fee Setting
	Transaction Replay
Cryptography Security	Random Number Range And Probability Distribution
	Cryptographic Algorithm Lmplementation/Use
Wallet Module & Account Security Audit	Private Key / Mnemonic Word Storage Security
	Private Key / Mnemonic Word Usage Security
	Private key/mnemonic generation algorithm
Others Security Audit	Database Security
	Thread Security
	File Permission Security

Category	Assessment Item
	Historical Vulnerability Security

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info

Project Name: InitVerse

Audit Time: February 14, 2025 – February 27, 2025

Language: GO

File Name	HASH
InitVerse	https://github.com/Project-InitVerse/chain/commit/d544fcddcc400a4afd787fa3285bb6e662a0971e

2.2 Summary

Severity	Found	
Critical	0	
High	0	
Medium	0	
Low	3	
Informational	1	

2.3 Key Findings

ID	Severity	Findings Title	Status	Confirm
NVE-001	Low	Uncle block is canceled but still have verify logic	Acknowledged	Confirmed
NVE-002	Low	unusedCode SealParentHash	Acknowledged	Confirmed
NVE-003	Low	rawVreflect.SliceHeader is deprecated	Acknowledged	Confirmed
NVE-004	Info	Typos in the Comments.	Acknowledged	Confirmed

Table 2.3: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 Uncle block is canceled but still have verify logic

ID:	NVE-001	**Location:**	
Severity:	Low	**Category:**	Bussiness Logic
Likelihood:	Low	**Impact:**	Low

Description:

In init hash we remove the uncles logic , set maxUncles = 0.

```go
// Inihash proof-of-work protocol constants.
var (
    BaseBlockReward                = big.NewInt(0).Mul(big.NewInt(145833333), big.NewInt
    maxUncles                      = 0
    allowedFutureBlockTimeSeconds = int64(140)

)
```

But we don't modify verify uncle logic

```go
func (inihash *Inihash) VerifyUncles(chain consensus.ChainReader, block *types.Block) e
    // If we're running a full engine faking, accept any input as valid
    if inihash.config.PowMode == ModeFullFake {
        return nil
    }
@>  // Verify that there are at most 2 uncles included in this block
@>  if len(block.Uncles()) > maxUncles {
        return errTooManyUncles
    }
```

Impact:

Return unexpected errors

Recommended mitigation:

```go

// VerifyUncles verifies that the given block's uncles conform to the consensus rules.
func (inihash *Inihash) VerifyUncles(chain consensus.ChainReader, block *types.Block) e
    return nil
}
```

Result: Confirmed

Fix Result: Acknowledged

3.2 unusedCode SealParentHash

ID:	NVE-002	Location:	
Severity:	Low	Category:	Bussiness Logic
Likelihood:	Low	Impact:	Low

Description:

parentHash is not used in mine logic ,but function SealParentHash still exists

```go

// mine is the actual proof-of-work miner that searches for a nonce starting from
// seed that results in correct final block difficulty.
func (inihash *Inihash) mine(block *types.Block, id int, seed uint64, abort chan struct
    // Extract some data from the header
    var (
        header = block.Header()
        hash   = inihash.SealHash(header).Bytes()
        target = new(big.Int).Div(two256, header.Difficulty)
        //number      = header.Number.Uint64()

        //parentHash = inihash.SealParentHash(header).Bytes()
        extraNonce = header.ExtraNonce
    )
```

```go
func (inihash *Inihash) SealParentHash(header *types.Header) (hash common.Hash) {
    hasher := sha3.NewLegacyKeccak256()

    rlp.Encode(hasher, []interface{}{
        header.ParentHash,
        header.Time,
    })

    hasher.Sum(hash[:0])
    return hash
}
```

Impact:

unusedCode SealParentHash

Recommended mitigation:

Delete unused code

Result: Confirmed

Fix Result: Acknowledged

3.3 reflect.SliceHeader is deprecated

ID:	NVE-003	Location:	
Severity:	Low	Category:	Code Maintainability
Likelihood:	Low	Impact:	Low

Description:

The reflect.SliceHeader function, found in algorithm.go, has been deprecated.

```go
    var cache []byte
    cacheHdr := (*reflect.SliceHeader)(unsafe.Pointer(&cache))
    dstHdr := (*reflect.SliceHeader)(unsafe.Pointer(&dest))
    cacheHdr.Data = dstHdr.Data
    cacheHdr.Len = dstHdr.Len * 4
    cacheHdr.Cap = dstHdr.Cap * 4
```

Impact:

It may increase the likelihood of runtime crashes.

Recommended mitigation:

Replace it with unsafe.Slice or unsafe.SliceData for better compatibility and performance.

Result: Confirmed

Fix Result: Acknowledged

3.4 Typos in the Comments

ID:	NVE-004	Location:	
Severity:	Info	Category:	Code Maintainability
Likelihood:	Low	Impact:	Low

Description:

There are several inaccurate statements in the comments within api.go and consensus.go.

```go
// GetHashrate returns the current hashrate for local CPU miner and remote miner.
func (api *API) GetHashrate() uint64 {
    return uint64(api.inihash.Hashrate())
}
// typo here
// GetHashrate returns the current hashrate for local CPU miner and remote miner.
func (api *API) GetBlockReward(number hexutil.Uint64) string {
    realBlockReward := api.inihash.GetBlockReward(uint64(number))
    //realBlockReward.Mul(realBlockReward, big9)
    //realBlockReward.Div(realBlockReward, big10)
    return "0x" + realBlockReward.Text(16)
}

```

Go

Wrap | Copy

```go
// calcDifficultyHomestead is the difficulty adjustment algorithm. It returns
// the difficulty that a new block should have when created at time given the
// parent block's time and difficulty. The calculation uses the Homestead rules.
func calcDifficulty(time uint64, parent *types.Header) *big.Int {
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.md
    // algorithm:
    // diff = (parent_diff +
    //         (parent_diff / 12288 * max(6 - (block_timestamp - parent_timestamp) // 5
    //        )
    bigTime := new(big.Int).SetUint64(time)
    bigParentTime := new(big.Int).SetUint64(parent.Time)
    // holds intermediate values to make the algo easier to read & audit
    x := new(big.Int)
    y := new(big.Int)
    // 1 - (block_timestamp - parent_timestamp) // 10
    x.Sub(bigTime, bigParentTime)
    x.Div(x, big5)
    x.Sub(big6, x)
    // max(1 - (block_timestamp - parent_timestamp) // 10, -99) ///typo here///
    if x.Cmp(bigMinus599) < 0 {
        x.Set(bigMinus599)
    }
    // (parent_diff + parent_diff // 2048 * max(1 - (block_timestamp - parent_timestamp
    y.Div(parent.Difficulty, params.DifficultyBoundDivisor)
    x.Mul(y, x)
    x.Add(parent.Difficulty, x)

    // minimum difficulty can ever be (before exponential factor)
    if x.Cmp(params.MinimumDifficulty) < 0 {
        x.Set(params.MinimumDifficulty)
    }

    return x
}
```

Go

Wrap | Copy

```go

func (inihash *Inihash) VerifyUncles(chain consensus.ChainReader, block *types.Block) e
    // If we're running a full engine faking, accept any input as valid
    if inihash.config.PowMode == ModeFullFake {
        return nil
    }
@>  // Verify that there are at most 2 uncles included in this block
@>  if len(block.Uncles()) > maxUncles {
        return errTooManyUncles
    }
```

Impact:

Incorrect comments may mislead developers and result in errors.

Recommended mitigation:

Correct and update the inaccurate comments.

Result: Confirmed

Fix Result: Acknowledged

4. CONCLUSION

In this audit, we thoroughly analyzed **InitVerse** Blockchain implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be PASSED. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.com

@EXVULSEC

github.com/EXVUL-Sec

ExVul