NUMEN

Smart Contract Audit Report

IotaSwap Smart Contract

27 Feb2023

Numen Cyber Labs - Security Services

Table of Content

1 Executive Summary 2

Methodology 2

2 Findings Overview 6

2.1 Project info and Contract address 6

2.2 Summary 6

2.3 Key Findings 7

3 Detailed Description of Findings 8

3.1 Uint overflow 8

3.2 release resources 9

3.3 redundant code 9

3.4 Sensitive Information Leakage 10

3.5 redundant code 11

3.6 logic issue 12

4 Conclusion 14

5 Appendix 15

5.1 Basic Coding Assessment 15

5.2 Advanced Code Scrutiny 16

6 Disclaimer 18

References 19

1 Executive Summary

Numen Cyber Technology was engaged by IotaSwap to review smart contract 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.

One Medium and five Low severities findings are related to overflow, owner authority, centralized risk.

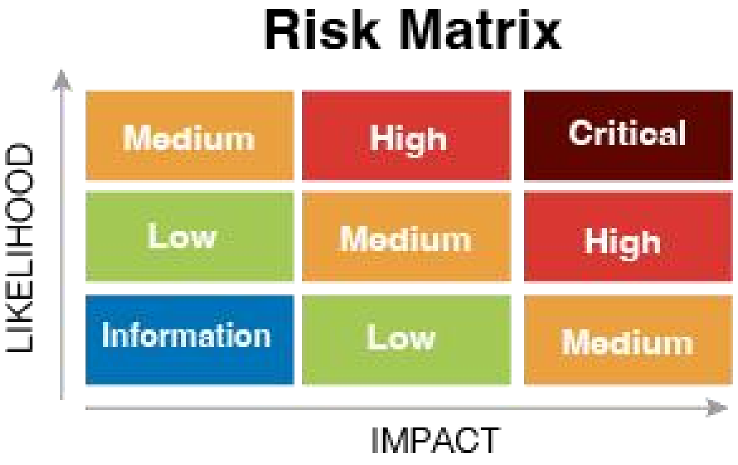
The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

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 and impact are categorized into three ratings: High, Medium and Low. Severity is determined by likelihood and impact and can be classified into four categories accordingly, Critical, High, Medium, Low shown in table 1.1.



Risk Matrix
LIKELIHOOD
IMPACT

Medium	High	Critical
Low	Medium	High
Information	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 smart contracts 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 smart contracts from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft fail Attack
	Hard fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure

	Circuit Breaker
	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
Additional Recommendations	Semantic Consistency Checks
	Following Other Best Practices

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 and Contract address

Project Name: IotaSwap

Project URL: https://github.com/TanglePay/biota-swap

Audit Time: 2023/2.27 - 2023/3.2

Language: go-lang

Commit Hash: 296f2a443e512f6b7ac56afeeff91e29e4e25ac2

Contract Name	Source Code Link
IotaSwap	https://github.com/TanglePay/biota-swap

2.2 Summary

Severity	Found	
Critical	0	
High	0	
Medium	1	🟧
Low	5	🟦 🟦 🟦 🟦 🟦
Informational	0	

2.3 KEY FINDINGS

One Medium and five Low severities findings are related to overflow, owner authority, centralized risk.

ID	Severity	Findings Title	Status	Confirm
NVE-001	Medium	Uint overflow	Ignore	Confirmed
NVE-002	Low	Release Resources	Ignore	Confirmed
NVE-003	Low	Redundant Code	Ignore	Confirmed
NVE-004	Low	Sensitive Information Leakage	Ignore	Confirmed
NVE-005	Low	Redundant Code	Ignore	Confirmed
NVE-006	Low	Logic Issue	Ignore	Confirmed

Table 2.1: Key Audit Findings

3 Detailed Description of Findings

3.1 Uint overflow

ID: NVE-001 Location: Accept.go

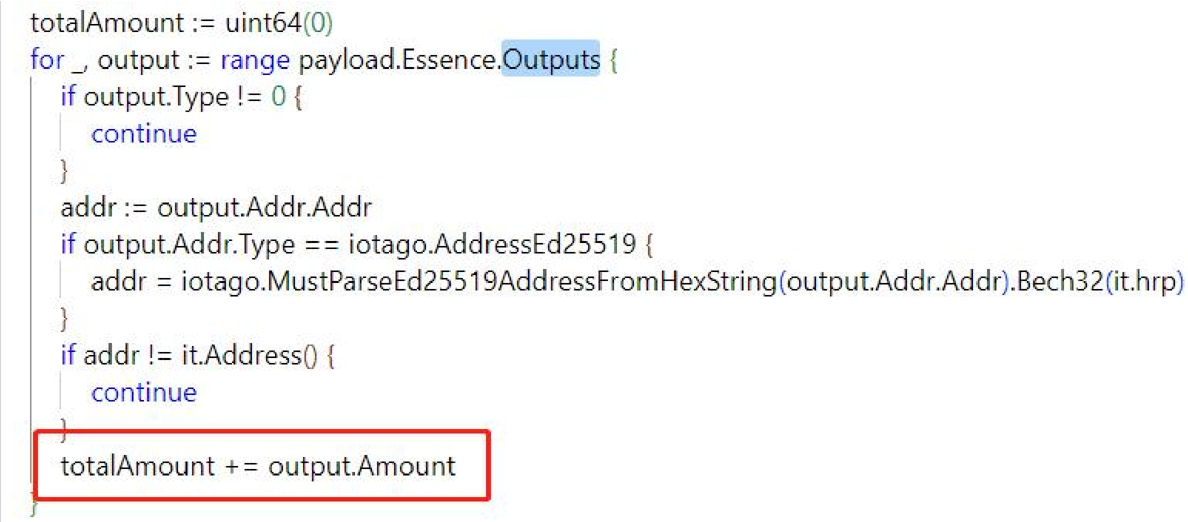
Severity: Low Category: Overflow Issues

Likelihood: Low

Impact: Low

Description:

As shown in the figure below, when "totalAmount" is performing mathematical operations, if the "output.Amount" data is too large, it will cause an overflow.



```
totalAmount := uint64(0)
for _, output := range payload.Essence.Outputs {
    if output.Type != 0 {
        continue
    }
    addr := output.Addr.Addr
    if output.Addr.Type == iotago.AddressEd25519 {
        addr = iotago.MustParseEd25519AddressFromHexString(output.Addr.Addr).Bech32(it.hrp)
    }
    if addr != it.Address() {
        continue
    }
    totalAmount += output.Amount
}
```

Recommendations:

Numen Cyber Lab recommends to ensure the security of data verification during operation to avoid overflow.

Result: Pass

Fix Result:

Ignore

3.2 RELEASE RESOURCES

ID: NVE-002

Location: Accept.go

Severity: Low

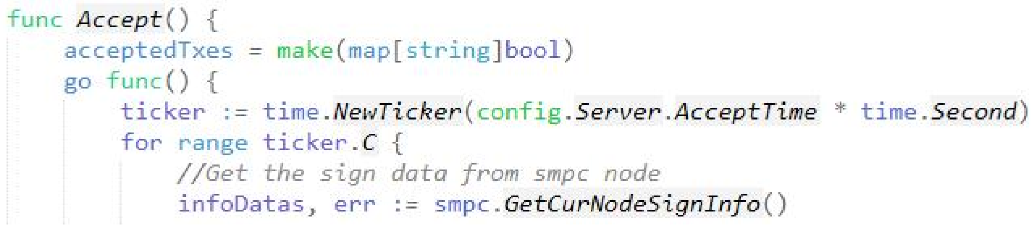
Category: Resources Issues

Likelihood: Low

Impact: Low

Description:

After using time.NewTicker without releasing resources. According to https://pkg.go.dev/time#NewTicker , ticker should be stopped.



```
func Accept() {
    acceptedTxes = make(map[string]bool)
    go func() {
        ticker := time.NewTicker(config.Server.AcceptTime * time.Second)
        for range ticker.C {
            //Get the sign data from smpc node
            infoDatas, err := smpc.GetCurNodeSignInfo()
```

Figure 1 function Accept

Recommendations:

ticker := time.NewTicker(config.Server.AcceptTime * time.Second)

defer ticker.Stop()

for range ticker.C {}

Result: Pass

Fix Result:

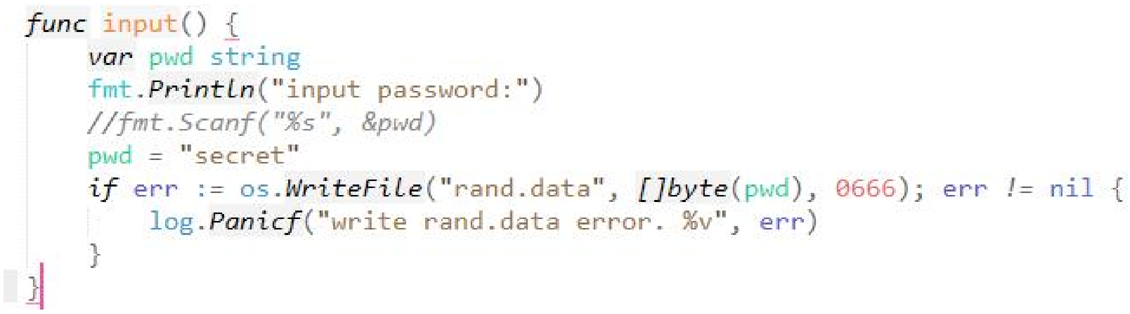
Ignore

3.3 REDUNDANT CODE

ID: NVE-003	Location: Main.go
Severity: Low	Category: Redundant Code Issues
Likelihood: Low	
Impact: Low	

Description:

It can only be used for testing, and should be removed for the production.



```
func input() {
	var pwd string
	fmt.Println("input password:")
	//fmt.Scanf("%s", &pwd)
	pwd = "secret"
	if err := os.WriteFile("rand.data", []byte(pwd), 0666); err != nil {
		log.Panicf("write rand.data error. %v", err)
	}
}
```

Figure 2 function input

Recommendations:

Numen Cyber Lab recommends before going live, delete this part of the code.

Result: Pass

Fix Result:

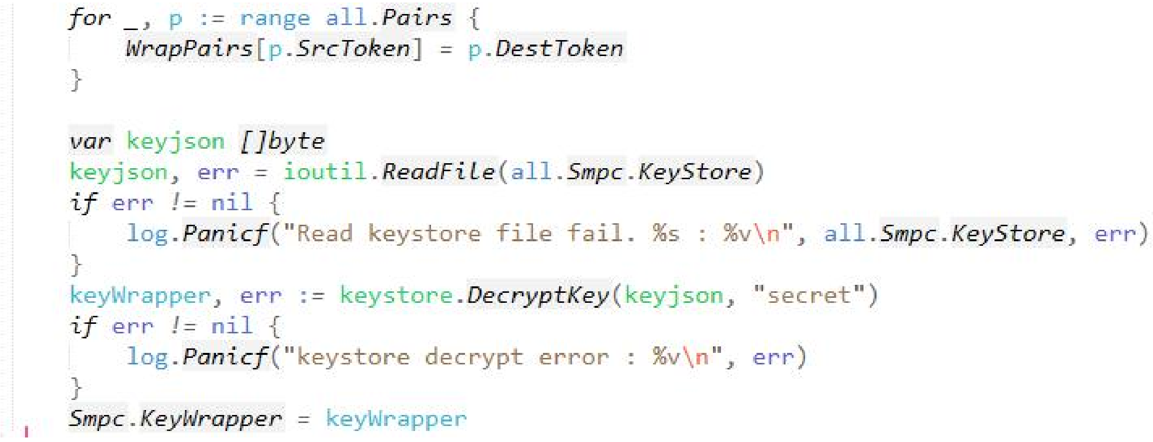
Ignore

3.4 Sensitive Information Leakage

ID: NVE-004	Location: Config.go
Severity: Low	Category: Sensitive Information Leakage Issues
Likelihood: Low	
Impact: Low	

Description:

The password plaintext is stored in the code.



```
for _, p := range all.Pairs {
    WrapPairs[p.SrcToken] = p.DestToken
}

var keyjson []byte
keyjson, err = ioutil.ReadFile(all.Smpc.KeyStore)
if err != nil {
    log.Panicf("Read keystore file fail. %s : %v\n", all.Smpc.KeyStore, err)
}
keyWrapper, err := keystore.DecryptKey(keyjson, "secret")
if err != nil {
    log.Panicf("keystore decrypt error : %v\n", err)
}
Smpc.KeyWrapper = keyWrapper
```

Figure 3 function Load

Recommendations:

Numen Cyber Lab recommends to use config file to read password.

Result: Pass

Fix Result:

Ignore

3.5 REDUNDANT CODE

ID: NVE-005 Location: Main.go

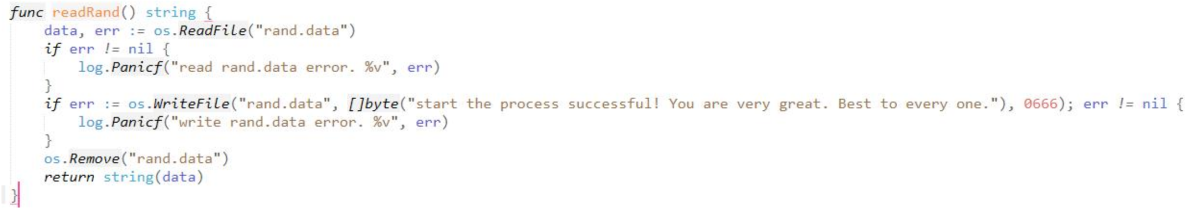
Severity: Low Category: Redundant Code Issues

Likelihood: Low

Impact: Low

Description:

Redundant code.



```go
func readRand() string {
    data, err := os.ReadFile("rand.data")
    if err != nil {
        log.Panicf("read rand.data error. %v", err)
    }
    if err := os.WriteFile("rand.data", []byte("start the process successful! You are very great. Best to every one."), 0666); err != nil {
        log.Panicf("write rand.data error. %v", err)
    }
    os.Remove("rand.data")
    return string(data)
}
```

Figure 4 function readRand

Recommendations:

Numen Cyber Lab recommends although read rand.data, but now the password is specified, it is recommended to delete.

Result: Pass

Fix Result:

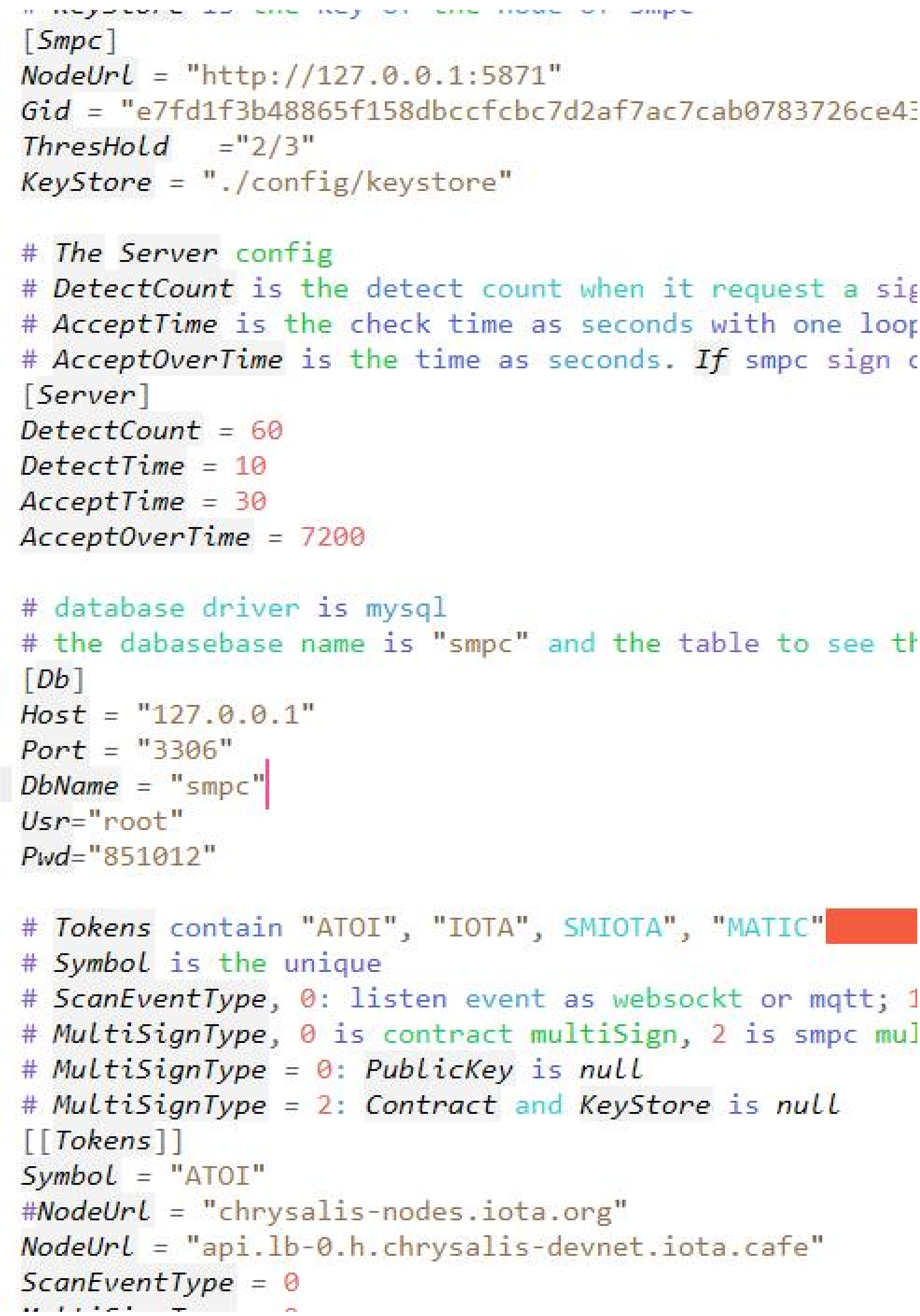
Ignore

3.6 LOGIC ISSUE

ID: NVE-006	Location: config.toml
Severity: Low	Category: Logic Issues
Likelihood: Low	
Impact: Low	

Description:

The format of nodeurl should be kept uniform, otherwise an error will occur.



```
[Smpc]
NodeUrl = "http://127.0.0.1:5871"
Gid = "e7fd1f3b48865f158dbccfcbc7d2af7ac7cab0783726ce43
ThresHold   ="2/3"
KeyStore = "./config/keystore"

# The Server config
# DetectCount is the detect count when it request a sig
# AcceptTime is the check time as seconds with one loop
# AcceptOverTime is the time as seconds. If smpc sign c
[Server]
DetectCount = 60
DetectTime = 10
AcceptTime = 30
AcceptOverTime = 7200

# database driver is mysql
# the dabasebase name is "smpc" and the table to see th
[Db]
Host = "127.0.0.1"
Port = "3306"
DbName = "smpc"
Usr="root"
Pwd="851012"

# Tokens contain "ATOI", "IOTA", SMIOTA", "MATIC"[redacted]
# Symbol is the unique
# ScanEventType, 0: listen event as websockt or mqtt; 1
# MultiSignType, 0 is contract multiSign, 2 is smpc mul
# MultiSignType = 0: PublicKey is null
# MultiSignType = 2: Contract and KeyStore is null
[[Tokens]]
Symbol = "ATOI"
#NodeUrl = "chrysalis-nodes.iota.org"
NodeUrl = "api.lb-0.h.chrysalis-devnet.iota.cafe"
ScanEventType = 0
```

Figure 5 config.toml

Recommendations:

Numen Cyber Lab recommends the nodeurl format remains unified.

Result: Pass

Fix Result:

Ignore

4 Conclusion

In this audit, we thoroughly analyzed IotaSwap smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been brought up to the project party, ignored issues are in line with the project design, and permissions are only used for the project to properly function. We therefore deem the audit result to be a **PASS.** 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 Behaviour

- Description: Examine whether sensitive behaviour 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 Numen'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 Numen 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. Numen's position is that each company and individual are responsible for their own due diligence and continuous security. Numen'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.

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

Numen Cyber Technology Pte. Ltd.

11 North Buona Vista Drive, #04-09,

The Metropolis, Singapore 138589

Tel: 65-63555555

Fax: 65-63666666

Email: sales@numencyber.com

Web: https://numencyber.com