Deliverable: Smart Contract Audit Report

iCommunity Labs

Smart Contract Review

Security Report

December 2021

Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Company. The content, conclusions and recommendations set out in this publication are elaborated in the specific for only project.

eNebula Solutions does not guarantee the authenticity of the project or organization or team of members that is connected/owner behind the project or nor accuracy of the data included in this study.

All representations, warranties, undertakings and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products.

Neither the Company nor any personating on the Company's behalf may be held responsible for the use that may be made of the information contained herein.

eNebula Solutions retains the right to display audit reports and other content elements as examples of their work in their portfolio and as content features in other projects with protecting all security purpose of customer. The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities fixed - upon a decision of the Customer.

Report Summary

Title		ICOM Smart Contract Audit	
Project Owner		iCommunity Labs	
Type		Public	
Reviewed by	Vatsal Raychura	**Revision date**	12/12/2021
Approved by	eNebula Solutions Private Limited	**Approval date**	12/12/2021
Nº Pages		15	

Overview

Background

iCommunity Labs requested that eNebula Solutions perform an extensive Smart Contracts audit of their token (ICOM) Smart Contracts.

Project Dates

The following is the project schedule for this review and report:

- **December 12**: Smart Contract Review Completed *(Completed)*
- **December 12**: Delivery of Smart Contract Audit Report *(Completed)*

Review Team

The following eNebula Solutions team member participated in this review:

- Sejal Barad, Security Researcher and Engineer
- Vatsal Raychura, Security Researcher and Engineer

Coverage

Target Specification and Revision

For this audit, we performed research, investigation, and review of the smart contract of iCommunity Labs ICOM token.

The following documentation was considered in-scope for the review:

- iCommunity Labs token project: https://icommunity.io/icom/en/
- Token block explorer: https://etherscan.io/address/0xb131f337c45d386ceec234e194b2663d5c3d9dcf
- Token's contract address: 0xb131f337c45d386ceec234e194b2663d5c3d9dcf

Introduction

Given the opportunity to review iCommunity Labs Contracts related smart contract source code, we in the report outline our systematic approach to evaluate potential security issues in the smart contract implementation, expose possible semantic inconsistencies between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to launch after resolving the mentioned issues, there are no critical or high or medium severity issues found related to business logic, security or performance.

About iCommunity Labs:

Item	Description
Issuer	iCommunity Labs
Website	https://www.icommunity.io/
Type	ERC20
Platform	Solidity
Audit Method	Whitebox
Latest Audit Report	December 12, 2021

The Test Method Information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

The Full List of Check Items:

Category	Check Item
Basic Coding Bugs	Constructor Mismatch
	Ownership Takeover
	Redundant Fallback Function
	Overflows & Underflows
	Reentrancy
	Money-Giving Bug
	Black hole
	Unauthorized Self-Destruct
	Revert DoS
	Unchecked External Call
	Gasless Send
	Send Instead of Transfer
	Costly Loop
	(Unsafe) Use of Untrusted Libraries
	(Unsafe) Use of Predictable Variables
	Transaction Ordering Dependence
	Deprecated Uses
Semantic Consistency Checks	Semantic Consistency Checks

Advanced DeFi Scrutiny	Business Logics Review
	Functionality Checks
	Authentication Management
	Access Control & Authorization
	Oracle Security
	Digital Asset Escrow
	Kill-Switch Mechanism
	Operation Trails & Event Generation
	ERC20 Idiosyncrasies Handling
	Frontend-Contract Integration
	Deployment Consistency
	Holistic Risk Management
Additional Recommendations	Avoiding Use of Variadic Byte Array
	Using Fixed Compiler Version
	Making Visibility Level Explicit
	Making Type Inference Explicit
	Adhering To Function Declaration Strictly
	Following Other Best Practices

Common Weakness Enumeration (CWE) Classifications Used in This Audit:

Category	Summary
Configuration	Weaknesses in this category are typically introduced during the configuration of the software.
Data Processing Issues	Weaknesses in this category are typically found in functionality that processes data.
Numeric Errors	Weaknesses in this category are related to improper calculation or conversion of numbers.
Security Features	Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)
Time and State	Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.
Error Conditions, Return Values, Status Codes	Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.

Resource Management	Weaknesses in this category are related to improper management of system resources.
Behavioral Issues	Weaknesses in this category are related to unexpected behaviors from code that an application uses.
Business Logics	Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application.
Initialization and Cleanup	Weaknesses in this category occur in behaviors that are used for initialization and breakdown.
Arguments and Parameters	Weaknesses in this category are related to improper use arguments or parameters within function calls.
Expression Issues	Weaknesses in this category are related to incorrectly written expressions within code.
Coding Practices	Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an ex pilotable vulnerability will be present in the application. They may not directly introduce a vulnerability, but indicate the product has not been carefully developed or maintained.

Findings

Summary

Here is a summary of our findings after analyzing the iCommunity Labs Smart Contract Review.

During the first phase of our audit, we studied the smart contract source code and ran our in-house static code analyzer through specific tools. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tool. We further manually review business logics, examine system operations, and place token management aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	0
High	0
Medium	0
Low	3
Total	3

We have so far identified that there are potential issues with severity of **0 Critical, 0 High, 0 Medium, and 3 Low**. Overall, these smart contracts are well-designed and engineered, though the implementation can be improved and bug free by common recommendations given under POCs.

Detailed Results

Basic Code Bugs (1 “level low” issue detected)

1. **Function Default Visibility**

    - **SWC ID:** 100
    - **Severity:** Low
    - **Location:** iComToken.sol
    - **Relationships:** CWE-710: Improper Adherence to Coding Standards
    - **Description:** The function definition of "null" lacks a visibility specified. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.
    - **Remediation:** Functions can be specified as being external, public, internal or private. It is recommended to make a conscious decision on which visibility type is appropriate for a function. This can dramatically reduce the attack surface of a contract system.

Basic Coding Bugs

1. Constructor Mismatch

- Description: Whether the contract name and its constructor are not identical to each other.
- Result: Not found
- Severity: Critical

2. Ownership Takeover

- Description: Whether the set owner function is not protected.
- Result: Not found
- Severity: Critical

3. Redundant Fallback Function

- Description: Whether the contract has a redundant fallback function.
- Result: Not found
- Severity: Critical

4. Overflows & Underflows

- Description: Whether the contract has general overflow or underflow vulnerabilities
- Result: Not found
- Severity: Critical

5. Reentrancy

- Description: Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs.
- Result: Not found
- Severity: Critical

6. Money-Giving Bug

- Description: Whether the contract returns funds to an arbitrary address.
- Result: Not found
- Severity: High

7. Blackhole

- Description: Whether the contract locks ETH indefinitely: merely in without out.
- Result: Not found
- Severity: High

8. Unauthorized Self-Destruct

- Description: Whether the contract can be killed by any arbitrary address.
- Result: Not found

- Severity: Medium

9. Revert DoS

- Description: Whether the contract is vulnerable to DoS attack because of unexpected reverts.
- Result: Not found
- Severity: Medium

10. Unchecked External Call

- Description: Whether the contract has any external call without checking the return value.
- Result: Not found
- Severity: Medium

11. Gasless Send

- Description: Whether the contract is vulnerable to gasless send.
- Result: Not found
- Severity: Medium

12. Send Instead of Transfer

- Description: Whether the contract uses send instead of transfer.
- Result: Not found
- Severity: Medium

13. Costly Loop

- Description: Whether the contract has any costly loop which may lead to Out-Of-Gas exception.
- Result: Not found
- o Severity: Medium

14. (Unsafe) Use of Untrusted Libraries

- Description: Whether the contract use any suspicious libraries.
- Result: Not found
- Severity: Medium

15. (Unsafe) Use of Predictable Variables

- Description: Whether the contract contains any randomness variable, but its value can be predicated.
- Result: Not found
- Severity: Medium

16. Transaction Ordering Dependence

- Description: Whether the final state of the contract depends on the order of the transactions.
- Result: Not found

- Severity: Medium

17. Deprecated Uses

- Description: Whether the contract use the deprecated tx.origin to perform the authorization.
- Result: Not found
- Severity: Medium

Semantic Consistency Checks

- Description: Whether the semantic of the white paper is different from the implementation of the contract.
- Result: Not found
- Severity: Critical

Automated Tools Results

MythX (2 “level low” issues detected):

We have made a deep scan (very sensitive), 90 minutes of duration:

https://dashboard.mythx.io/#/console/analyses/81ea98fa-8253-47db-912f-65706a5e4174

Line	SWC Title	Severity	Short Description
1	Floating Pragma	Low	A floating pragma is set.
57	State Variable Default Visibility	Low	The state variable visibility is not set.

Conclusion

In this audit, we thoroughly analyzed iCommunity Labs Smart Contracts. The current code base is well organized but there are promptly some low-level issues found in this phase of Smart Contract Audit.

Meanwhile, we need to emphasize that smart contracts as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

About eNebula Solutions

We believe that people have a fundamental need to security and that the use of secure solutions enables every person to more freely use the Internet and every other connected technology. We aim to provide security consulting service to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through the production to launch and surely after.

The eNebula Solutions team has skills for reviewing code in C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, and JavaScript for common security vulnerabilities & specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent and open about the work we do.

For more information about our security consulting, please mail us at – contact@enebula.in