IBC ERC20 Module

Audit Report

VER 1.0

16th June 2022

No. 2022061619481

Project Summary

1. Project Introduction

The contract of IBC-ERC20 mainly serves the IBC function of OKC. When other chains transfer new assets to OKC in the form of cross-chain through IBC agreement, OKC will automatically deploy a set of corresponding ERC20 contracts for the assets, including:

- contracts/ModuleERC20. sol
- contracts/ModuleERC20Proxy. sol

Meanwhile, if there are native ERC20 assets deployed on OKC and want to cross other chains through IBC agreement, the ERC20 needs to inherit the following contracts:

- contracts/nativeERC20/NativeERC20Base. sol

2. Audit Summary

Project Name	IBC ERC20 Module	**Platform**	N/A
Token	N/A	**Token symbol**	N/A
Start date	10th June 2022	**Language**	Solidity
End date	13th June 2022	**Website**	N/A
Github	https://github.com/okex/IBC-ERC20/tree/5989f7305276ce25274b04dfc2c7499afc38a571/contracts	**Whitepaper**	N/A

16th June 2022

3. Audit Scope

ID	File	SHA-256 checksum
contracts	REC20.sol	d5830e888ac60f02a2adbc46ac1de125809ff15d1f3add9cd65530f838c692f0
contracts	ModuleERC20.sol	bfcb78767353ff237c2940853a9e9d901513c60d0c59fb11e6d6bbc07ab2af48
contracts	ModuleERC20Proxy.sol	f408a817804e4501c340db9a1d89de77bde2cc12b6dc64ba51a2222bcc69eb74
contracts/ nativeERC20	NativeERC20Base.sol	98eafb512ec1330397757bff81ad76930db177c640b5cb2f9c9b68144c68ec6f

4. Code Structure

```
├── ERC20.sol                  #Basic ERC20 Agreement
├── ModuleERC20.sol            #ERC20 token template transferred by OKC cross-chain
through IBC agreement
├── ModuleERC20Proxy.sol       #Renewable proxy contract for ERC20 tokens transferred to
OKC cross-chain through IBC agreement
└── nativeERC20
    └── NativeERC20Base.sol    #Basic category of ERC20 tokens transferred from OKC to
other chains through IBC agreement
```

Audit Report Summary

1. Audit Methods

By clearly understanding the design purpose, operation principle and implementation mode of the project, the audit team conducted in-depth research and analysis of the contract code. Based on clarifying the calling relationship between each contract and its functions, the possible loopholes in the contract are located and analyzed. Finally, a document containing the problem descriptions and corresponding modification suggestions is formed.

Audit methods	Static analysis, Manual Review

2. Audit Process

Steps	Operation	Description
1	Background	Reading the descriptions, white papers, contract source code, and other relevant information the project team provides to ensure a proper understanding of project functions.
2	Automated testing	Automated detection tools will be mainly used to scan the source code to find common potential vulnerabilities
3	Manual reveiw	The code will be thoroughly reviewed line by line by engineers to find potential vulnerabilities
4	Logical proofread	The engineer will compare the understanding of the code with the information provided by the project and check whether the code implementation is in line with the white paper information.
5	Test case	Including test case design, test scope analysis, symbolic execution, etc.
6	Optimization items	Review the project from the aspects of maintainability, security and operability according to the application scenarios, call methods and the latest research results

3. Risk Levels

Risk level	Issue description
Critical	Fatal risks and hazards that need to fixed immediately.
Major	Some high risks and hazards that will lead to related problems that must be solved
Medium	Some moderate risks and pitfalls may lead to potential risks that will eventually need to be addressed
Minor	There are low risks and hazards, mainly details of various types of mishandling or warning messages, which can be set aside for the time being
Information	Some parts can be optimized, such problems can be shelved, but it is recommended that the final solution

4. Audit Results

ID	Audit project	Risk level	Status
1	Reentrancy	None	
2	Injection	None	
3	Authentication bypass	None	
4	MEV Possibility	None	
5	Revert	None	
6	Race condition	None	
7	Insufficient Gas Griefing	None	
8	The major impact of flash loans	None	
9	Unreasonable economic model	None	
10	Predictable random numbers	None	
11	Voting rights management confusion	None	

ID	Audit project	Risk level	Status
12	Privacy leak	None	
13	Improper use of time on chain	None	
14	Improper codes in fallback function	None	
15	Improper identification	None	
16	Inappropriate opcode	None	
17	Inappropriate assembly	None	
18	Constructor irregularities	None	
19	Return value irregularity	None	
20	Event irregularity	None	
21	Keywords irregularity	None	
22	Not following ERC standards	None	
23	Irregularity of condition judgment	Medium	Resolved
24	Risk of liquidity drain	None	
25	Centralization Risk	None	
26	Logic change risk	None	
27	Integer overflow	None	
28	Improper function visiblity	None	
29	Improper initialization of variables	None	
30	Improper contract calls	None	
31	Variable irregularities	None	
32	Replay	None	
33	Write to Arbitrary Storage Location	None	
34	Honeypot logic	None	
35	Hash collision	None	
36	Improper reward logic	None	
37	Deprecated methods used	None	
38	Coding principles break	None	
39	Risk of multiple initialization	Major	Resolved

* In the above table, if the status column is **Acknowledged**, the audit team has informed the project owner of the vulnerability. Still, the project owner has not made any changes to the vulnerability or has not announced to the audit team the progress of the changes to the vulnerability. If the status column is **Resolved**, the project owner has changed the exposure, and the audit team has confirmed the changes.

5. Risk and Modification Program

The following section provides detailed information about the risk items learned after the audit, including the type of risk, risk level, location of the issue, description of the problem, recommendations for changes, and feedback from the project owner.

1. Irregularity of condition judgment

Location	Contract file	Risk Status	Risk level
Line 25	ERC20.sol	Resolved	Medium

① Description

Nonstandard conditional judgment is used to describe the nonstandard coding of conditional judgment statements. When "condition judgment" is required, <require> is not called, or such conditions have nonstandard codes.
Since this type of judgment statement often determines the specific execution logic of the contract, it is recommended to standardize the code of conditional judgment in order to avoid unexpected execution logic of the contract.
In this project, ERC20 sol Line25， require（_decimals == 0， “ERC20: already initialized；”）；
Because the contract does not restrict the initialization parameter "decimals_" The value of <decimals_== 0> is passed in when deploying the contract, resulting in the invalidation of the one-time initialization restriction originally played by the code.

② Recommendation

It is recommended to allow the decimals parameter of ERC20 token to be 0 and use the bool value to mark whether to initialize.

③ Update

A. Modify the condition "whether ERC20. sol contract is initialized" to use the Independent Boolean value for judgment.
B. Modify all contracts using the 0.8.0 version of solidity to the confirmed 0.8.7

③ Code

```JavaScript
    function __ERC20_init(
        string memory name_,
        string memory symbol_,
        uint8 decimals_
    ) internal {
        //@OKLink Audit Description: Possibility of conditional failure
        //@OKLink Audit Solution: Use the bool value as the flag for initialization judgment
        require(_decimals == 0, "ERC20: already initialized;");

        _name = name_;
        _symbol = symbol_;
        _decimals = decimals_;
    }
```

16th June 2022

2. Risk of multiple initialization

Location	Contract file	Risk Status	Risk level
Line 13	MouduleERC20.sol	Resolved	Major

① Description

The initialization function sets an explicit bool value as the judgment condition for whether it has been initialized. Otherwise, after initialization is performed on the contract chain, unexpected calls for reinitialization will occur, resulting in unsafe modification of the state variables.
In this project, because the condition judgment in the superclass initialization function may fail, there is a risk that the initialization function will be executed again.

② Recommendation

It is recommended to use the bool value as the flag for initialization judgment

③ Update

A. Modify the condition "whether ERC20. sol contract is initialized" to use the Independent Boolean value for judgment.
B. Modify all contracts using the 0.8.0 version of solidity to the confirmed 0.8.7

③ Code

```JavaScript
    function initialize(string memory denom_, uint8 decimals_) public {
        //@OKLink Audit Description: Whether it has been initialized is not judged__ ERC20_  Init function initialization judgment may fail
        //@OKLink Audit Solution: Use the bool value as the flag for initialization judgment
        __ERC20_init(denom_, denom_, decimals_);
    }
```

Disclaimer

i. This audit report focuses only on the types of audits identified in the final report issued. Other unknown security vulnerabilities are not part of this audit, and we do not accept responsibility for them.

ii. We shall only issue an audit report based on an attack or vulnerability that existed or occurred before the issuance of the audit report. We cannot determine the likely impact on the security posture of our projects for new attacks or vulnerabilities that may exist or occur in the future, and we are not responsible for them.

iii. The security audit analysis and other elements of our published audit report shall be based solely on documents and materials (including, but not limited to, contract codes) provided to us by the Project Party before the release of the audit report. Such documents and materials shall not be untrue, inaccurate, uninformative, altered, deleted, or concealed, and if the documents and materials provided by the Project Party are false, inaccurate, uninformative, changed, deleted or hidden, or if the documents and materials provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted or concealed, or if the documents and materials provided by the Project Party are uninformative, uninformative, altered, deleted or hidden. If the records and information provided by the Project Party are untrue, inaccurate, uninformative, altered, deleted, or concealed, or if changes to such documents and information are made after the issuance of the audit report, we shall not be liable for any loss or adverse effect arising from any inconsistency between the reflected and actual conditions.

iv. The Project Parties are aware that our audit report is based on documents and information provided by the Project Parties and relies on the technology currently available. However, due to the technical limitations of any organization, there is a possibility that our audit report may not fully detect all risks. Our audit team encourages the project development team and any interested parties to conduct subsequent testing and audits of the project.

v. The project owner warrants that the project for which we are engaged to provide audit or testing services is legal, compliant, and does not violate applicable laws. The audit report is for the project owner's reference only, and the contents, manner of obtaining, use of, and any services or resources involved in the audit report shall not be relied upon for investment, tax, legal, regulatory, or advisory purposes of any kind, and we shall not be liable therefor. The Project Party shall not refer to, quote, display, or send the Audit Report in whole or in part to any third party without our prior written consent. The Project Party shall bear any loss or liability arising from that place. We assume no responsibility for any reliance on or use of the audit report for any purpose.

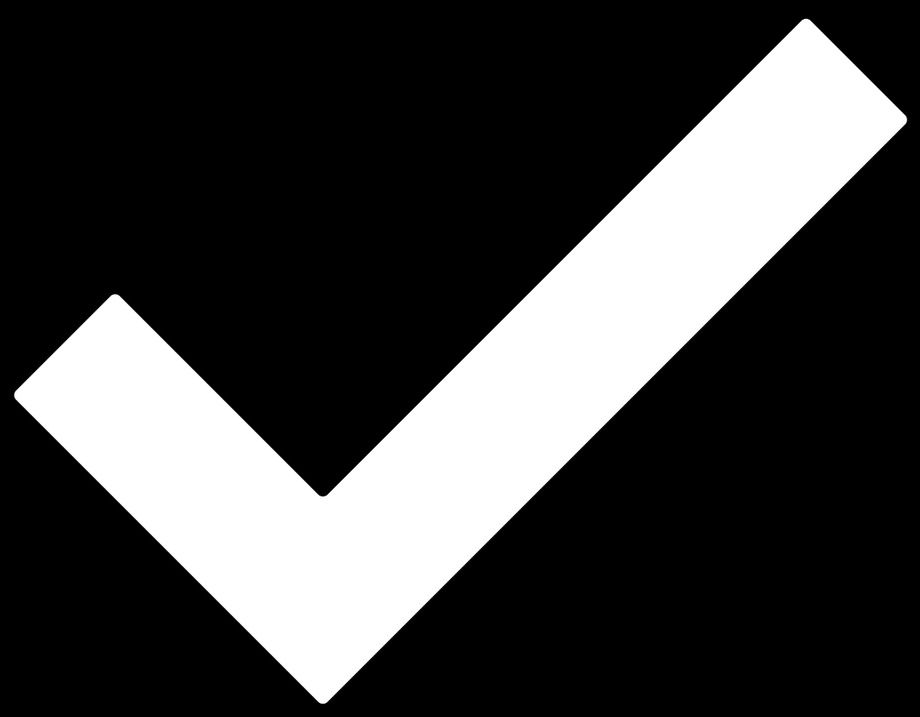
vi. This audit report does not cover the compiler of the contract or any areas beyond the programming language of the Smart Contract. The risk and liability of the audited Smart Contract arising from references to off-chain information or resources is the sole responsibility of the project party.

vii. Force Majeure. Force majeure means an unforeseen event whose occurrence and consequences cannot be avoided and cannot be overcome by the parties at the time of entering into the contract, including but not limited to natural disasters such as war, typhoon, flood, fire, earthquake, tidal wave, lightning, natural disaster, strike, nuclear explosion, epidemic and other unforeseen events such as changes in laws, regulations and policies and governmental acts, whose occurrence and consequences cannot be prevented or avoided, and which contains, affects or delays the performance by either party of all or part of its obligations under the contract.

viii. Suppose either party believes that the occurrence of force majeure affects the performance of its obligations under this Agreement. In that case, it shall promptly notify the other party and, depending on the extent of the effect of the event on the performance of the Agreement; the parties shall consult to determine whether to terminate the Agreement or partially relieve itself of its obligations to perform the Agreement, or to extend the performance of the Agreement.

ix. In force majeure, neither party shall be deemed in breach or non-performance of its obligations under this Agreement. Any financial commitments existing before the event shall not be affected, and the project party shall make payment for work performed by us.

16th June 2022



Passed.

Date	16th June 2022
Audit Team	歐科雲鏈

The purpose of this audit is to review the OKC modular ERC20 contract for IBC cross-chain based on the solid language to study its design and architecture and find possible vulnerabilities.