BEOSIN
Web3 Security & Compliance

TheOpenLayer

Smart Contract Security Audit

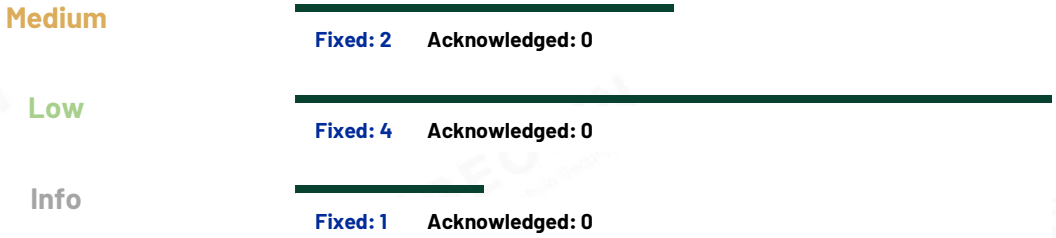
No. 202411271116

Nov 27th, 2024

Contents

Summary of Audit Results

After auditing, 2 Medium, 4 Low-risk, 1 Info item was identified in the TheOpenLayer project. Specific audit details will be presented in the Findings section. Users should pay attention to the following aspects when interacting with this project:



- **Project Description**

TheOpenLayer is a Staking project deployed on TON.

Users can stake specified tokens in the Master contract. The Master contract generates a corresponding wallet contract for each user to record their staking information. Each staking action is associated with a unique `stakeIndex`.

To withdraw staked tokens, users must first call the `unstake` function to initiate a withdrawal request. Based on the withdrawal amount, the corresponding staking order is converted into a pending order. Users can use the `restake` function to convert a pending order back into a staking order.

Pending orders require a designated waiting period (set by the owner of the Master contract) before the `withdraw` function can be called to complete the withdrawal process.

1 Overview

1.1 Project Overview

Project Name	TheOpenLayer
Project Language	Tact
Platform	The Open Network
Code Base	https://github.com/the-open-layer/TheOpenLayerTheOpenLayer/tree/dev
Commit Hash	5c14cafaaa18f100388261777b5c9be7f2f410e6 5e14a092f4d7f38fce4e0a22b401222516855378 bf0b86091ed66f2373b2359e6c6ca63cf6df868f 7e508c1ed48f7caf09c1a6d68ec4eb3b6ee1c8bb 6c5a587a12420abd3c811c4d00b15aaadfbb0892

1.2 Audit Overview

Audit work duration: Nov 19, 2024 – Nov 27, 2024

Audit team: Beosin Security Team

1.3 Audit Method

The audit methods are as follows:

1. Formal Verification

Formal verification is a technique that uses property-based approaches for testing and verification. Property specifications define a set of rules using Beosin's library of security expert rules. These rules call into the contracts under analysis and make various assertions about their behavior. The rules of the specification play a crucial role in the analysis. If the rule is violated, a concrete test case is provided to demonstrate the violation.

2. Manual Review

Using manual auditing methods, the code is read line by line to identify potential security issues. This ensures that the contract's execution logic aligns with the client's specifications and intentions, thereby safeguarding the accuracy of the contract's business logic.

The manual audit is divided into three groups to cover the entire auditing process:

The Basic Testing Group is primarily responsible for interpreting the project's code and conducting comprehensive functional testing.

The Simulated Attack Group is responsible for analyzing the audited project based on the collected historical audit vulnerability database and security incident attack models. They identify potential attack vectors and collaborate with the Basic Testing Group to conduct simulated attack tests.

The Expert Analysis Group is responsible for analyzing the overall project design, interactions with third parties, and security risks in the on-chain operational environment. They also conduct a review of the entire audit findings.

3. Static Analysis

Static analysis is a method of examining code during compilation or static analysis to detect issues. Beosin-VaaS can detect more than 100 common smart contract vulnerabilities through static analysis, such as reentrancy and block parameter dependency. It allows early and efficient discovery of problems to improve code quality and security.

2 Findings

Index	Risk description	Severity level	Status
TheOpenLayer-01	TON additional payment for the contract	Medium	Fixed
TheOpenLayer-02	Wrong withdraw address	Medium	Fixed
TheOpenLayer-03	Deprecated message mode	Low	Fixed
TheOpenLayer-04	stakeIndex record error	Low	Fixed
TheOpenLayer-05	Problems with setting unakeThreshold	Low	Fixed
TheOpenLayer-06	Struct assignment error	Low	Fixed
TheOpenLayer-07	Redundant codes	Info	Fixed

Finding Details:

[TheOpenLayer-01] TON additional payment for the contract

Severity Level	**Medium**
Lines	TheOpenLayerMaster.tact#L118-151
Type	Business Security
Description	When the user calls `withdraw`, `tonAmount` is passed in, which will be sent to the jettonWallet address. However, when the user passes in TON, `tonAmount` is not included. Only `forwardAmount` is included. As a result, this TON may be paid by the Master contract. And `forwardAmount` is used and consumed when sending a message to jettonWallet, but it is involved again when sending a message to the destination address.
Recommendation	It is recommended to conduct more comprehensive checks on the TON passed in by users, including subsequent handling fees, or delete unnecessary calls.
Status	**Fixed.** Removed the function of sending messages to destination. Changed `tonAmount` to `forwardAmount` in sending messages to jettonWallet.

[TheOpenLayer-02] Wrong withdraw address

Severity Level	**Medium**
Lines	TheOpenLayerWallet.tact#L190
Type	Business Security
Description	In the withdraw function of the Wallet contract, jettonWallet should not be entered by the user and should be fixed to the jetton wallet address of the Master contract. If the user enters an error, the user's withdrawal will fail, and the pending information will be deleted and cannot be re-withdrawn.
Recommendation	It is recommended to check the jettonWallet entered by the user.
Status	**Fixed.** Now jettonwallet cannot be entered by users, but is instead set by the owner of the master contract.

[TheOpenLayer-03] Deprecated message mode

Severity Level	Low
Lines	TheOpenLayerWallet.tact#L81
Type	Coding Conventions
Description	This is a problem that exists in both Master and Wallet. We think that the `SendRemainingBalance` mode should not be used here. This mode will send all the balances in the contract and is generally only used to destroy the contract.
Recommendation	It is recommended to change it to `SendRemainingValue`.
Status	**Fixed.** The message sending mode has been changed to `SendRemainingValue`.

[TheOpenLayer-04] stakeIndex record error

Severity Level	Low
Lines	TheOpenLayerWallet.tact#L71
Type	Business Security
Description	The contract's record of `stakeIndex` is confused, which may cause the index of `stakedJettons` to be skipped and the corresponding relationship to be incorrect when using the `ReDeposit` function.
Recommendation	It is recommended to delete `self.stakeIndex` += 1 in `ReDeposit` and do not mix `self.stakeIndex` with the Index to be recorded in `stakedJettons`. Update the Index used by `stakedJettons` separately.
Status	**Fixed.** And modify the operation method of `withdraw` from extracting only the pending transformed by the corresponding `stakeIndex` to extracting after integrating multiple `stakeIndex` into one pending according to the number of `stakeIndex` to be extracted.

[TheOpenLayer-05] Problems with setting unakeThreshold

Severity Level	Low
Lines	TheOpenLayerWallet.tact#L49-53
Type	Business Security
Description	The `unstakeThreshold` is set by the owner of the TheOpenLayerWallet, that is, the user who stakes the funds. This is related to the delay from the subsequent user's initiation of a withdrawal request to the actual withdrawal. From the perspective of project operation, it is not recommended to set it as a user setting.
Recommendation	It is recommended to modify the default value of `unstakeThreshold` in StakingWalletTemplate according to actual needs, so that it remains at the initial value and cannot be modified.
Status	**Fixed.** The functionality of setting `unstakeThreshold` is moved to the Master contract.

[TheOpenLayer-06] Struct assignment error

Severity Level	Low
Lines	TheOpenLayerWallet.tact#L65-68
Type	Coding Conventions
Description	The `receiveStakeInternal` function in the TheOpenLayerWallet contract sets the parameters in the wrong order when creating the structure. `stakeIndex` and `stakeTime` are in the wrong order.
Recommendation	It is recommended to change the order of structure assignment.
Status	**Fixed.** Modified to the correct order.

[TheOpenLayer-07] Redundant codes

Severity Level	Info
Lines	TheOpenLayerMaster.tact#L18
Type	Coding Conventions

Description	After the `lockvalue` in the contract is initialized, it is not assigned a value elsewhere and remains at 0.
	`lockedValue: Int as coins = 0;`
	The `userWalletAddress` and `userWallet` functions duplicate functionality and are redundant code.
	`get fun userWalletAddress(owner: Address): Address { return contractAddress(self.getUserWallet(owner)); }`
	`get fun userWallet(owner: Address): Address { return contractAddress(self.getUserWallet(owner)); }`

Recommendation	It is recommended to add the code to assign the value according to the design requirements. If it is redundant code, it can be deleted.
Status	**Fixed.** Redundant code has been removed.

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1(Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

Likelihood \ Impact	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	Medium	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

- **Critical**

Critical impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

3.1.3 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.4 Fix Results Status

Status	Description
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Redundant Code
		Deprecated Items
		Gas Consumption*
		Event Trigger
		Throw Usage
2	General Vulnerability	Message Forgery*
		Restore on Failure*
		Integer Overflow/Underflow
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)*
		Function Call Permissions
		Message Flow Error*
		Returned Value Security
		Data Structure Error*
		Replay Attack
		Overriding Variables
		Third-party Protocol Interface Consistency
3	Business Security	Business Logics
		Business Implementations
		Missing Calibration
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Arbitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

BEOSIN
Web3 Security & Compliance

Official Website
https://www.beosin.com

Telegram
https://t.me/beosin

Twitter
https://twitter.com/Beosin_com

Email
service@beosin.com