rivo × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

October 31, 2024

ABSTRACT

In this report, we consider the security of smart contracts of Rivo project. Our task is to find and describe security issues in the smart contracts of the platform.

DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

SUMMARY

In this report, we considered the security of Rivo smart contracts. We described the audit process in the section below.

The audit showed one critical issue: Incorrect shares rate. The audit also revealed several issues of medium severity: Disperency with ERC4626 standard, Fee and deposit limit avoiding, Incorrect fee receiver, Project roles, Unable to run tests. Moreover, several low-severity issues were found.

The tests and the code coverage failed to run.

After the initial audit, the codebase was updated.

The developers fixed the critical issue Incorrect shares rate and medium severity issues: Disperency with ERC4626 standard, Fee and deposit limit avoiding, Incorrect fee receiver, Unable to run tests. After receiving comment from the developers, we decreased severity of the Project roles issue, and moved it to notes. One new issue Insufficient code coverage of medium severity was discovered. All low severity issues and notes were fixed or commented on.

All the tests passed. The code coverage of the audit scope is insufficient.

The issues have been fixed or commented on in the commits on different pull requests, but they have not yet been merged into one branch.

GENERAL RECOMMENDATIONS

We recommend fixing the rest of the mentioned issues. We also recommend implementing CI to run tests, calculate code coverage, and analyze code with linters and security tools.

PROJECT OVERVIEW

Project description

For the audit, we were provided with Rivo project on a private repository, commit ea7c8a9fd31b7bfbb7d578ac4396c82568c53ba3.

The scope of the audit included:

- contracts/RivoIndex.sol;
- contracts/RivoBaseStrategy.sol;
- contracts/GlobalStructs.sol;
- contracts/rivoIndex/;
- contracts/gateway/;
- contracts/rivoBaseStrategy/;
- contracts/FeeProcessor/.

The documentation for the project included the following link and comments from the developers.

The tests and the code coverage failed to run.

The total LOC of audited sources is 1803.

Codebase update #1

After the initial audit, we were provided with commits 28b855b41c905b9814ffa661d9c770d722a58025 and 5276e02464a4aaf58c34ec0305789785bc2a3efd.

In this update, the developers fixed most of the issues mentioned in the initial report and commented on the remaining issues. One new medium issue was introduced.

All 49 tests pass successfully. The code coverage of the current scope is 55.9%.

AUDIT PROCESS

We started the audit on October 4 and finished on October 22, 2024.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project. During the work, we stayed in touch with the developers and discussed confusing or suspicious parts of the code.

We manually analyzed all the contracts within the scope of the audit and checked their logic. Among other, we verified the following properties of the contracts:

- Whether the integrations are correct;
- Whether an inflation attack is not possible in the vault;
- Whether the vault correpsonds to the EIP-4626;
- Whether the code logic is the same as in the provided scheme;
- Standard Solidity issues;
- Influence of the admin's roles in the project.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts. We also sent the results to the developers in the text file.

We could not run the tests and calculate the code coverage.

In a private report, we combined all the verified issues we found during the manual audit or discovered by automated tools.

We made the recheck on 28-29 October, 2024. We re-ran the tests and recalculated the code coverage. We also checked the fixes of previous issues.

Finally, we updated the report.

MANUAL ANALYSIS

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

C01. Incorrect shares rate (fixed)

The user may receive more assets during the withdrawal process in the **RivoIndex** contract than they should have and steal other users' money.

Assume in the vault: `totalSupply = 10,000 shares`, `totalAssets = 10,000 ETH`. Let `bufferAmount = 0` (it can be reduced to 0 by withdrawing the exact `RivoIndex.bufferAmount` value).

Then 1 share = 1 ETH.

User A has `3,000 shares`. They begin to withdraw from strategies (as `bufferAmount = 0`) and make 3 withdrawals in 1 transaction (it is not necessary to make in 1 transaction until strategies ETA is not updated):

1. They withdraw `1,000 shares`. The request goes to strategies, and user A will receive `1,000 ETH` after completing the withdrawal process. The `totalSupply = 9,000`, `totalAssets = 10,000 ETH` and 1 share = 1.1111..(1) ETH.
2. They withdraw another `1,000 shares`. The request goes to strategies, and user A will receive `1,111 ETH` after some time. The `totalSupply = 8,000`, `totalAssets = 10,000 ETH` and 1 share = 1.25 ETH.
3. They withdraw another `1,000 shares`. The request goes to strategies, and user A will receive `1,250 ETH` after completing the withdrawal process.

As a result, the user receives `3,361 ETH` instead of `3,000 ETH`.

> The issue has been fixed and is not present in the latest version of the code.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Disperency with ERC4626 standard (fixed)

According to the ERC4626 standard:

> "... deposit should return the same or more shares as previewDeposit if called in the same transaction".

However, in the **RivoIndex** contract, the `deposit` function will always return less because fees are taken. Consider implementing fee collection calculation inside the `previewDeposit` function in the **RivoIndex** contract.

The same case can be found in the `previewWithdraw`, `previewRedeem`, and `previewMint` functions of the **RivoIndex** contract.

> **The issues have been fixed and are not present in the latest version of the code.**

M02. Fee and deposit limit avoiding (fixed)

The **RivoIndex** contract is inherited by the ERC4626 contract, and the mint function from the ERC4626 is not overridden.

It will lead to the possibility of avoiding fee (the `_calculateAndProcessDepositFee` function) and deposit limit (the `maxVaultCapacity` variable).

Consider overriding the `mint` function and the `maxMint` function to set the corresponding limit.

> **The issue has been fixed and is not present in the latest version of the code.**

M03. Incorrect fee receiver (fixed)

All fees from the **RivoIndex** contract, during the deposit, withdraw, redeem processes, and fees received from a strategy are sent to the **TreasuryFeeProcessor** contract instead of the **TreasuryFeeCollector**.

Because of this, the process of allocating fees to fee processors (in the `TreasuryFeeProcessor.processFee` function) may not work properly, as the balance will not show the rest of the amount; it will include other fees on it. And the `balance >= rest` check at line 105 will be incorrect.

> **The issues have been fixed and are not present in the latest version of the code.**

M04. Unable to run tests (fixed)

The project's tests could not be run. We always note the availability of tests and the high code coverage percentage.

> The issue has been fixed and is not present in the latest version of the code.

M05. Insufficient code coverage (new)

The code coverage is insufficient. As an example, the **rivoBaseStrategy** folder is almost not covered. We highly recommend covering the code with tests and ensuring that all tests pass and the code coverage is high.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Duplicate function (fixed)

The `withdraw` and `redeem` functions of the **RivoIndex** contract override the functions in the ERC4626 (withdraw and redeem) but they actually duplicate each other. Consider leaving only the `_withdraw` function as `overridden`.

> The issue is not actual in the latest version of the code.

L02. Gas consumption (commented)

Reading a storage variable multiple times consumes a lot of gas. Consider declaring a local variable with this value in the following cases in the project:

- **(fixed)** The `currentLastId` variable in the `TreasuryFeeCollector.addFeeProcessor` function;
- The `activeStrategies.at(i)` and `strategies[activeStrategies.at(i)]` in the `RivoIndex._withdraw` function;
- **(fixed)** The `nextFeeProcessor` and `treasury` variables in the `TreasuryFeeProcessor.processFee` function;
- **(fixed)** The `withdrawalRequests[withdrawalId]` in the `RivoIndex._processWithdrawalRequest` function;
- The `quoteRequestIdsCounter` and `swapRequestIdsCounter` in the `requestQuote` and `requestSwap` functions of the **ManualOneInchDexAggregatorCommunicator** contract;
- The `swapBuffer` in the `_fulfillSwap` function of the **OneInchDexAggregatorCommunicator** contract.

> `Comment from the developers:`
> Unfixed items are acknowledged.

L03. Immutable variables (fixed)

The following variables are set during contract deployment and never change later. Consider declaring them as `immutable` to reduce gas consumption in the project:

- The `want` and `id` in the **RivoBaseStrategy** contract;
- The `localLzChainId`, `vault`, and `asset` in the **Gateway** contract;
- The `collector` in the **TreasuryFeeProcessor** contract.

> The issues have been fixed and are not present in the latest version of the code.

L04. Misleading comment (commented)

According to the NatSpec at line 40 of the **CrosschainMessageTransfers** contract, the `requestEstimateTotalAssets` is:

> "service function for update estimated total assets of a strategy. Can be called only by WATCHER".

However, it can also be called by the `DEFAULT_ADMIN_ROLE`.

> `Comment from the developers:`
> Acknowledged. Comments will be added just before the release.

L05. TODO comment (fixed)

The **CrosschainFundsTransfers** contract has several TODO comments.

> The issues have been fixed and are not present in the latest version of the code.

L06. Undeleted fee processors' info (fixed)

Consider removing a processor from the `feeProcessors` mapping in the whole `removeFeeProcessor` function of the **TreasuryFeeCollector** contract and set the `lastActiveFeeProcessorId` and `firstActiveFeeProcessorId` variables to 0 when `getFeeProcessorsLength() == 1` for getters to work correctly.

> The issue has been fixed and is not present in the latest version of the code.

L07. Unprotected functions (commented)

Anyone can call the `addPointCut` and `removePointCut` functions of the **EnumerableSetBasedFeesManagement** contract and add/remove the `hashOfPointCutName` from the `pointCuts` mapping. The `view` functions (`pointCutsLength` and `pointCutAt`) will not return correct values.

> `Comment from the developers:`
>
> **Acknowledged. This part will not be used, because it is deprecated logic, but for future plans we will leave it in repo.**

L08. Unused functions (fixed)

The following functions are not used in the project:

- **(fixed)** `TreasuryFeeCollector.receiveFee`;
- `SimpleCrosschainDepositary._sendFees`. However, this function is in the base strategy contracts, so it will probably be used in the next strategies.

> **The issue has been fixed and is not present in the latest version of the code.**

L09. Public -> external (fixed)

Consider declaring the following functions as `external` instead of `public` when possible to improve code readability and optimize gas consumption in the project:

- `TreasuryFeeProcessor.id`;
- `TreasuryFeeCollector.getFeeProcessorByIndex`.

> **The issues have been fixed and are not present in the latest version of the code.**

Notes

N01. Complex logic of the fee processors chain (commented)

The fee allocation process in the **TreasuryFeeProcessor** contract is complicated. It seems that it could be done within one contract to reduce gas consumption, as now 2-3 external calls are made at each iteration of the chain.

> `Comment from the developers:`
> Acknowledged. This complex logic will be used later for DAO, VE purposes.

N02. Config setting (commented)

It is essential to set the configuration of LayerZero correctly to ensure the protocol works appropriately.

For instance, it is unclear now which **SendLibrary** contract will be used. If there is some malicious logic there, then a request from the vault (`MessageVaultOperations._requestFundsFromStrategy`) may always revert as the vault calls the endpoint, and it calls the **SendLibrary** contract during the sending process in LayerZero.

The admin should also set the fees in the vault correctly, as they need to cover the gateway fees exactly in the project.

> `Comment from the developers:`
> Acknowledged. This is a part of LayerZero V2, so, we can't avoid this complexity.

N03. Lack of money in the strategy (commented)

During withdrawal, the vault requests tokens from the strategies when the buffer is insufficient. If the strategy does not have enough money, the backend should monitor such cases and send money through the `contractconvertAndSendFundsToVaultsBuffer` function of the **SimpleCrosschainDepositary** to complete the withdrawal process.

> `Comment from the developers:`
> Acknowledged. This is a risk of all DeFi applications, so, we are accepting it.

N04. Project roles (commented)

In the current implementation, the system depends heavily on the admin role. Thus, there are scenarios that can lead to undesirable consequences for the project and its users, e.g., if the admin's private keys become compromised. The roles can:

- The `DEFAULT_ADMIN_ROLE` and `WATCHER` roles can create any strategy through the **StrategiesManagement** contract and send users money to it (`CrosschainFundsTransfers.sendFunds`). They can also remove strategies or change a strategy info.
- In the **RivoIndex** contract, the `DEFAULT_ADMIN_ROLE` can:
    - Change the fee receiver address, change the deposit limit;
    - Can call the `initializeVault` function multiple times and change shares rate;
    - Change withdrawal, deposit, and redemption fees;
    - Change the gateway address.
- In the **Gateway** contract:
- The `DEFAULT_ADMIN_ROLE` can add or delete strategies;
- The `WATCHER` can:
    - Trigger the vault when the fees from a strategy are received. However, if this is not done immediately after the vault has received the tokens, this money will be mixed with user money - as it will be included in the buffer of the vault and can be withdrawn. When calling the `triggerVaultCallbackFeesWereReceivedFromStrategy` function, there may not be enough money in the vault balance, and the fee will not be sent.
    - Trigger the vault when the vault receives the requested funds from the strategy. This directly influences the delay of the withdrawal request.
- In the **FundsVaultOperations** contract, the `DEFAULT_ADMIN_ROLE` can change the bridge parameters;
- In the **MessageVaultOperations** contract, the `DEFAULT_ADMIN_ROLE` can change gas limits for the send in the LayerZero;
- In the **TreasuryFeeProcessor** contract, the owner can change the distribution percentage, change the treasury address, and change processors in the chain.

- In the **TreasuryFeeCollector** contract, the owner can:
  - Add or remove fee processors from the chain;
  - Launch the fee distribution process;
  - Change the vault address;
  - Withdraw all fees from the collector.
- In the **CrosschainFundsTransfers** contract, the `DEFAULT_ADMIN_ROLE` and `WATCHER` roles can request and send funds to a strategy.
- In the **EnumerableSetBasedFeesManagement** contract, the `DEFAULT_ADMIN_ROLE` can set the point cut info.
- In the **SimpleCrosschainDepositary** contract, the `DEFAULT_ADMIN_ROLE` can change the gateway address and send funds from strategy to the vault.
- In the **SwitchableDexAggregatorCommunicatorSubscriber** contract, the `DEFAULT_ADMIN_ROLE` can change the address of the DexAggregatorCommunicator (which can request swaps and receive tokens rate), change the slippage and the flag whether the DexAggregator is used.

  > The `DEFAULT_ADMIN_ROLE` has been replaced with the `MANAGER` role.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig in the project.

> `Comment from the developers:`
> Acknowledged. Every our product is under multisig control, this is part of our internal security rules, so, this vault also will be fully controlled by the following multisig accounts:
>
> - 0x942f39555D430eFB3230dD9e5b86939EFf185f0A
> - 0xC1287e8e489e990b424299376f37c83CD39Bfc4c
> - 0x9391047E5e66D55390a045c6055ceD37Cd80fA50

N05. Withdrawal delay (commented)

Due to a delay in the withdrawal process of the vault (the **RivoIndex** contract), a receiver will not be able to catch the tokens receiving moment (the `RivoIndex._processWithdrawalRequest` function) as they can also be a contract. It may be necessary, for instance, if a receiver has complex logic and sends tokens to further protocols.

> `Comment from the developers:`
> Acknowledged. Yes, there is withdrawal delay, because it is crosschain Vault, so, also can not avoid this.

This analysis was performed by Pessimistic:

Daria Korepanova, Senior Security Engineer
Yhtyyar Sahatov, Security Engineer
Konstantin Zherebtsov, Business Development Lead
Irina Vikhareva, Project Manager
Alexander Seleznev, CEO

October 31, 2024