Ostium Security Scan Results

by Pessimistic

This is not a security audit

This report is public

May 27, 2024

Abstract

This report considers the security of smart contracts of the Ostium protocol. Our task is to find and describe security issues using the static-analysis tools Slither and Slitherin and help resolve them.

The work is financially covered by the Arbitrum Foundation grant.

Disclaimer

Current work does not give any warranties on the security of the code. It is not an audit or its replacement. Performing this scan, we focused on finding as many crucial issues as possible rather than making sure that the protocol was entirely secure. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Summary

In this report, we described issues found in smart contracts of the Ostium protocol.

We scanned the codebase and manually rejected or verified all automated findings, revealing five relevant issues.

The developers commented on all of them.

The entire process is described in the section below.

Scan process

Under the Arbitrum Foundation grant, we researched and developed Arbitrum-specific detectors. They became publicly available with Slitherin v0.6.0 release.

Workflow

This work consisted of five stages:

1. For the scan, we were provided with the Ostium project on a private GitHub repository, commit ad309e434e31a7a93520bd3ee7015b3e8f886ab0.
2. For the analysis of the protocol, we launched Slither v0.10.1 and Slitherin v0.6.1 on the provided codebase.
3. One auditor manually checked (rejected or accepted) all findings reported by the tools. The second auditor verified this work. We shared all relevant issues with the protocol developers and answered their questions.
4. The developers reviewed the findings and gave comments on all issues.
5. We prepared this final report summarizing all the issues and comments from the developers.

Issue categories

Within the confines of this work, we were looking for:

- Arbitrum-specific problems;
- Standard vulnerabilities like re-entrancy, overflow, arbitrary calls, etc;
- Non-compliance with popular standards like ERC20 and ERC721;
- Some access control problems;
- Integration issues with some popular DeFi protocols;
- A wide range of code quality and gas efficiency improvement opportunities.

This scan does not guarantee that these issues are not present in the codebase.

Scan results

Issue category	Number of detectors	Status
Compilation	1	Passed
Arbitrum Integration	3	Passed
AAVE Integration	1	Passed
Uniswap V2 Integration	7	Passed
OpenZeppelin	2	Passed
ERC-20	7	Passed
ERC-721	2	Passed
Known Bugs	15	Passed
Access Control	3	Passed
Arbitrary Call	5	Passed
Re-entrancy	6	Passed
Weak PRNG	2	Passed
Upgradability	2	Passed
Ether Handling	3	Passed
Low-level Calls	2	1 issue found
Assembly	2	Passed
Inheritance	3	Passed
Arithmetic	2	Passed
Old Solidity Versions Bugs	10	Passed
Code Quality	15	Passed
Best Practices	4	1 issue found
Gas	7	3 issues found

Discovered Issues

Unsafe low-level call

The `_whitelisted` function of the **OstiumVault** contract and `_isWhitelisted` function of the **OstiumTrading** and **OstiumFaucet** contracts uses low-level call to `getContractAddress` function of the **OstiumRegistry** contract to get the `OstiumWhitelist` contract address. As the `getContractAddress` function has `view` state mutability, we recommend considering the usage of the `interface` or `staticcall` instead.

> *<u>Comment from the developers:</u> Yes, we reviewed your remark and are glad that it has no severity. We will follow the recommendations.*

Missing event

The `setFaucetParams` function of the **OstiumFaucet** contract does not `emit` an `event`. Emmiting of event in setter functions allows contract owner and relevant parties to be notified about important state changes within the contract.

> *<u>Comment from the developers:</u> We reviewed the remark and will follow the recommendations.*

External vs public

The following functions can be declared as `external` instead of `public`:

- The `registerContracts`, `updateContracts`, `unregisterContracts` and `getContractAddress` functions of the **OstiumRegistry** contract;
- The `setConfig` function of the **OstiumLinkUpKeep** contract;
- The `setPairFundingFeesArray` and `setPairRolloverFeesArray` functions of the **OstiumPairInfos** contract;
- The `tvl` and `marketCap` functions of the **OstiumVault** contract;
- The `firstEmptyOpenLimitIndex` and `getOpenLimitOrder` functions of the **OstiumTradingStorage** contract.

Consider declaring functions as `external` instead of `public` when possible to improve code readability and optimize gas consumption.

> *<u>Comment from the developers:</u> We reviewed the remark and will follow the recommendations.*

Immutable variables

The following variables are set during contract deployment and never changed later:

- The `registry` in the **OstiumVerifier** contract;
- The `token` in the **OstiumFaucet** contract;
- The `registry` in the **OstiumPriceRouter** contract.

We recommend declaring them as `immutable` to reduce gas consumption and improve code quality.

> *Comment from the developers: We reviewed the remark and will follow the recommendations.*

Redundant code

We recommend verifying that the `canExecuteTimeout` variable of the **OstiumTradingStorage** contract will be removed from the production code as described in the corresponding comment.

> *Comment from the developers: We reviewed the remark and will follow the recommendations.*

This analysis was performed by Pessimistic:

Evgeny Bokarev, Junior Security Engineer
Vitalii Nikolaichuk, Junior Security Engineer
Pavel Kondratenkov, Senior Security Engineer
Nikita Kirillov, Product Owner
Evgeny Marchenko, CTO
Konstantin Zherebtsov, Business Development Lead

May 27, 2024