CERTIK

Security Assessment

O3 Swap II

May 12th, 2021

Summary

This report has been prepared for O3 Swap II smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	O3 Swap II
Description	O3 Swap is a cross-chain aggregation protocol, allowing users to access multi-chain liquidity sources on one platform
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/O3Labs/o3swap-contracts/tree/53c009e09ece07328a3a566262dbc4f8a1697478
Commits	1. 53c009e09ece07328a3a566262dbc4f8a1697478 2. 23e8c5b18890f81755e451e29468429d0e270883

Audit Summary

Delivery Date	May 12, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

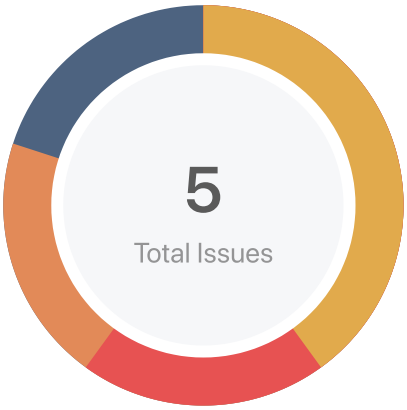
Vulnerability Summary

Total Issues	5
Critical	1
Major	1
Minor	2
Informational	1
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
OOT	core/assets/O3Token/O3.sol	4ed5fc98af137ea80057ae724903d8d4f84e134dd408c1fa00108df3af1e2b45
OSO	core/staking/O3Staking.sol	605988829f9f5d20f02e7fe0eac20be7729f485f368d343ac5164bc2309c0df1
IOO	interfaces/IO3.sol	dfa48ce2467aa83530d36fa35a739c5cd36cc42753eafadae23612c401d09970
CGS	libs/GSN/Context.sol	4d4a7387aac95c24e5a86c0252731e960cc7365a49074e4e98608ab1085669ab
POL	libs/lifecycle/Pausable.sol	02bff121fd3cec5ee6c3734413228052a6c5c5a553ea93bf472e905b4f5ce6e5
SMO	libs/math/SafeMath.sol	37b66f9303ebba057f67b1d3f1ded96c7e31679ebbc536968fd63db10be1c716
OOL	libs/ownership/Ownable.sol	c46be407eaae3647dca8bf94a009309261cd27b93bf765f7f8b81068a62b197a
ERC	libs/token/ERC20/ERC20.sol	81496a587bf6c182471a0bcd8ba47c98d631b165b2980ab07d77da39ead5a5c9
IER	libs/token/ERC20/IERC20.sol	557e00368a9f8ceb3ae18fb62461ce8bfd39d7753619c3816b1844e33aa26d68
SER	libs/token/ERC20/SafeERC20.sol	4f9b153446f8c0d52f27140de2357e5ffde19d03ebaf461a192267d241562e4e
AOL	libs/utils/Address.sol	9fa598d7828f35a742675f4be0510686e0a1863e935468e9ffbb4c02d4c80b31
RGO	libs/utils/ReentrancyGuard.sol	0531814c03d35ac068c39b958bc6b48644f3f3cf475e414c7b9c8361b31eb374
MOL	utils/Migrations.sol	f584faac284c65af00b7c7f55af32b5ddbb1e4b98fe1e5e2d5f9890b673adcbe

Findings



Critical	1 (20.00%)
Major	1 (20.00%)
Minor	2 (40.00%)
Informational	1 (20.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
OOT-01	Division Before Multiplication	Mathematical Operations	Minor	Resolved
OSO-01	Lack of Input Validation	Volatile Code	Informational	Resolved
OSO-02	Administrator Capability	Logical Issue	Major	Resolved
OSO-03	Division Before Multiplication	Mathematical Operations	Minor	Resolved
OSO-04	Calculation of `totalStaked`	Logical Issue	Critical	Resolved

OOT-01 | Division Before Multiplication

Category	Severity	Location	Status
Mathematical Operations	Minor	core/assets/O3Token/O3.sol: 205	Resolved

Description

Mathematical operations in the aforementioned lines perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation

We advise that order multiplication before division:

```
uint256 maxUnlockSpeed = toBeUnlocked.mul(FACTOR_DENOMINATOR).div(_unlockBlockGap[token]);
```

Alleviation

The development team heeded our advice and resolved this issue in commit 23e8c5b18890f81755e451e29468429d0e270883.

OSO-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	core/staking/O3Staking.sol: 78~79	Resolved

Description

The assigned value to `StakingToken` in the constructor of `O3Staking.sol` should be verified as a non-zero value to prevent error. The assigned value to `O3Token` in the constructor of `O3Staking.sol` should be verified as a non-zero value to prevent error.

Recommendation

Check that the passed-in values are non-zero values.

Example:

```
require(_stakingToken != 0, "_stakingToken is a zero value");
require(_o3Token != 0, "_o3Token is a zero value");
```

Alleviation

The development team heeded our advice and resolved this issue in commit 23e8c5b18890f81755e451e29468429d0e270883.

OSO-02 | Administrator Capability

Category	Severity	Location	Status
Logical Issue	Major	core/staking/O3Staking.sol: 258~262	Resolved

Description

To bridge the trust gap between the administrator and users, the administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following capability of the administrator:

- Administrator can transfer tokens to any account under unpredicted cases via the `collect` function.

Recommendation

To improve the trustworthiness of the project, dynamic runtime changes on the protocol should be notified to clients. Any plan to call the `collect` function is better to move to the execution queue of Timelock.

Alleviation

The development team only want to rescue the tokens sent to the contract by mistake, so we advised them to avoid the withdraw of the staking token in the function `collect`, they heeded our advice and resolved this issue in commit 23e8c5b18890f81755e451e29468429d0e270883.

OSO-03 | Division Before Multiplication

Category	Severity	Location	Status
Mathematical Operations	Minor	core/staking/O3Staking.sol: 102, 231, 203	Resolved

Description

Mathematical operations in the aforementioned lines perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation

We advise that order multiplication before division:

```
uint currentProfit = (currentProfitAccumu.sub(preUnitProfit)).mul(rec.staked).div(ONE);
```

```
_unitProfit = _sharePerBlock.mul(ONE).div(totalStaked);
```

Alleviation

The development team heeded our advice and resolved this issue in commit 23e8c5b18890f81755e451e29468429d0e270883.

OSO-04 | Calculation of `totalStaked`

Category	Severity	Location	Status
Logical Issue	Critical	core/staking/O3Staking.sol: 142~159	Resolved

Description

The calculation of `totalStaked` below when user unstaking is not correct:

```
142     function unstake(uint amount) external nonReentrant _logs_ {
143         require(!_withdarawPaused, "O3Staking: UNSTAKE_PAUSED");
144
145         StakingRecord storage rec = _stakingRecords[_msgSender()];
146
147         require(amount > 0, "O3Staking: ZERO_UNSTAKE_AMOUNT");
148         require(amount <= rec.staked, "O3Staking: UNSTAKE_AMOUNT_EXCEEDED");
149
150         totalStaked = amount.sub(totalStaked);
151         _updateUnitProfitState();
152
153         uint userTotalProfit = _settleCurrentUserProfit(_msgSender());
154         _updateUserStakingRecord(_msgSender(), rec.staked.sub(amount), userTotalProfit);
155
156         emit LOG_UNSTAKE(_msgSender(), amount);
157
158         _pushToken(StakingToken, _msgSender(), amount);
159     }
```

Recommendation

Consider correcting the calculation of `totalStaked` when user unstaking as below:

```
totalStaked = totalStaked.sub(amount);
```

Alleviation

The development team heeded our advice and resolved this issue in commit 23e8c5b18890f81755e451e29468429d0e270883.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.