oceanos Smart Contract

Audit Report

Tue Jan 30 2024

contact@scalebit.xyz

https://twitter.com/scalebit_

ScaleBit

oceanos Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	A staking and lending project
Type	DeFi
Auditors	ScaleBit
Timeline	Mon Jan 22 2024 - Tue Jan 23 2024
Languages	Solidity
Platform	Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/oceanos-labs/oceanos-contracts
Commits	ae3b598dbe6d2781e49adb777cf6ee8e03171498 030ef1f9d4a5292efec5e90a987f6462fbfbcc68

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MIB	contracts/incentives/MultiIncentiveBase.sol	44273af56977d327a03ea92da28b62e9da1a5e14
MMI	contracts/incentives/MinterMultiIncentive.sol	2753b35131038a52a779841f44f92d98eb90c510
CMI	contracts/incentives/CollateralMultiIncentive.sol	8a2cea882475dd8be51929edba837426abd18c89
SMI	contracts/incentives/StakeMultiIncentive.sol	ef0f73d943fbdcf326fdca9796e56718f2acd45a
OOR	contracts/oracle/OceanosOracle.sol	64d42fb17b1e833acfa7aa449fdbe0556b2c9843
OUSD	contracts/token/OcUSD.sol	d7efdc1c37ed12094d5fdbb03b6a2326a2338a13
TPO	contracts/token/TethysPoint.sol	555a380c81342bb5af52b179b46e08ab1b1833d5
IERC2D	contracts/interfaces/IERC20Detailed.sol	7b88f4d2b8f349d00a3d2684ef3706311276b686
IPC	contracts/interfaces/IPriceCalculator.sol	3fb4a81dd03c0cd18212795f3a766d817fd3aa5f
IPB	contracts/interfaces/IPoolBase.sol	8fe4a0c4b64b8bc8b7cf8914e2cdc6f1f888b1da
IOUSD	contracts/interfaces/IOcUSD.sol	496c31e7b8e1fe8efcb998482cb83b9ec144d493

ISERC2	contracts/interfaces/IShoebillERC20.sol	87511f744dc21a3379d15d59dfb789567728c0a2
IMI	contracts/interfaces/IMultiIncentive.sol	c94335633f441c9c1f961d397f4a668784bd4ffe
SIP	contracts/pool/SimpleIssuedPool.sol	31d043bbe1abfd56331f82f8ffbbe3a8ee63e73f
SYP	contracts/pool/ShoebillYieldPool.sol	cbf6c6863e692a9abacc3bf3f2012e71ac86f757
PBA	contracts/pool/base/PoolBase.sol	d47068774ad598e0eef0e898dcf9b4fea0aac860
YPB	contracts/pool/base/YieldPoolBase.sol	0347ab4ef7af36d288fa4e7d06ed456dcb9ee4a9
ATI	contracts/governance/AdminTimelock.sol	a17a6d115a259bac2863f8c5044caf14f5fe9440

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	5	2
Informational	0	0	0
Minor	5	5	0
Medium	0	0	0
Major	2	0	2
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Oceanos Labs to identify any potential issues and vulnerabilities in the source code of the oceanos contracts smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 7 issues of varying severity, listed below.

ID	Title	Severity	Status
MBE-1	Centralization Risk	Major	Acknowledged
MIB-1	Incompatible With Deflationary Token	Major	Acknowledged
MIB-2	Missing A Zero Address Check	Minor	Fixed
MIB-3	Missing Param Check	Minor	Fixed
OOR-1	Lack of Events Emit	Minor	Fixed
PBA-1	Missing Borrowed Amount Check	Minor	Fixed
SYP-1	Unimplemented Function	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the oceanos contracts Smart Contract:

Admin

- The Admin can initialize a new `ERC20` token and mint\burn tokens and set the minter and burner through `initialize()\mint()\burn()\setMintAllowed()\setBurnAllowed\flashMint()`.
- The Admin can set the `Gov` \ `RewardsDistributor` address, add rewards into the contract and set the duration of rewards through `setGov()\setRewardsDistributor\addReward()\notifyRewardAmount()\setRewardsDuration()`.
- The Admin can initialize the `OceanosOracle`, set the update threshold time, set the token price, set the reporter, and set the primary price calculator through `initialize() \setThreshold() \setPrice() \setPrices() \setReporter() \setPrimaryPriceCalculator()`.
- The Admin can set the `Gov` \ `FeeReceiver` \ `PriceCalculator` address and set the `PoolConfiguration` \ `MintIncentivePool` \ `CollateralIncentivePool` through `setGov() \setFeeReceiver() \setPriceCalculator() \setPoolConfiguration() \setMintIncentivePool() \setCollateralIncentivePool()`.

User

- The User can stake into the contract through `stake()`.
- The User can withdraw from the contract through `withdraw()`.
- The User can withdraw and get reward from the contract through `exit()`.
- The User can get the rewards through `getReward()`.
- The User can stake their collateral into the contract and get the `usdAsset` token through `mint()`.
- The User can withdraw their collateral through `withdraw()`.
- The User can burn their `usdAsset` token and get the collateral through `repay()`.
- The User can repay the `usdAsset` on behalf of the target address and get their collateral through `redeem()`.
- The User can liquidate others' collateral through `liquidation()`.

4 Findings

MBE-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/token/MintBurnERC20.sol#45,53

Descriptions:

Centralization risk was identified in the smart contract.

- The privileged `admin` can invoke `mint()` and `burn()` to mint or burn any amount of tokens.

Any potential leaks or malicious manipulation could lead to serious issues.

Suggestion:

It is recommended to confirm if it aligns with the design.

MIB-1 Incompatible With Deflationary Token

Severity: Major

Status: Acknowledged

Code Location:

contracts/incentives/MultiIncentiveBase.sol#141;

contracts/incentives/StakeMultiIncentive.sol#50

Descriptions:

Due to the unknown address of `_token`, when the token is deflationary, the amount of tokens transferred to the contract by the user may not be accurate.

Suggestion:

It is recommended to add a check for the deflationary token as:

```
amountBefore = _token.balanceOf(address(this));
IERC20(_rewardsToken).safeTransferFrom(msg.sender, address(this), amount);
amountAfter = _token.balanceOf(address(this));
require(amountAfter - amountBefore >= amount);
```

MIB-2 Missing A Zero Address Check

Severity: Minor

Status: Fixed

Code Location:

contracts/incentives/MultiIncentiveBase.sol#26,41

Descriptions:

It should be checked whether the set address is a zero address.

Suggestion:

It is recommended to add a zero address check for these addresses.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MIB-3 Missing Param Check

Severity: Minor

Status: Fixed

Code Location:

contracts/incentives/MultiIncentiveBase.sol#45 113

Descriptions:

The function `addReward` is missing a check for the params `_rewardsDistributor` and `_rewardsDuration` . And the `setRewardsDistributor` function has the same issue.

Suggestion:

It is recommended to add a check for the params as:

```
require(_rewardsDistributor != address(0), "Reward Distributor must be non-zero address");
require(_rewardsDuration > 0, "Reward duration must be non-zero");
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

OOR-1 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

contracts/oracle/OceanosOracle.sol#51-68;

contracts/pool/base/PoolBase.sol#73-115;

contracts/incentives/MultiIncentiveBase.sol#41,45,113,168

Descriptions:

The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for those sensitive functions.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

PBA-1 Missing Borrowed Amount Check

Severity: Minor

Status: Fixed

Code Location:

contracts/pool/base/PoolBase.sol#247

Descriptions:

In the `_repay` function, there is a missing check for `borrowedAmount[_onBehalfOf]`.

Suggestion:

It is recommended to add a check as:

```
require(borrowedAmount[_onBehalfOf] >= amount, "repay amount exceeds borrowed amount");
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

SYP-1 Unimplemented Function

Severity: Minor

Status: Fixed

Code Location:

contracts/pool/ShoebillYieldPool.sol#39

Descriptions:

There is an unused function in the smart contract, and the function `claimYield` does not implement any functionality.

Suggestion:

It is recommended to confirm if it aligns with the design.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.

- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.

- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.

- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.

- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.

- **Partially Fixed:** The issue has been partially resolved.

- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.

SCALEBIT VERIFIED

SCALEBIT VERIFIED