DeHacker

Code Security Assessment

OpenFi

Mar19th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	OPENFI
Platform	ARBITRUM
website	https://www.openfi.pro
Type	Dex
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	0	0	1
Medium	2	0	0	0	0	2
Minor	3	0	0	0	0	3
Informational	1	0	0	0	0	1
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
PWQ	Staking.sol	4bd4730f39ca6b36bc74ba518ee2481e30f6b6ab7d14610ec5e01ae2cbd50d2b
WSE	StakingConfig.sol	c75c0d5de627e42ea17e2113779134e33190b5b153bf1d36f817c700a01d89cf
OES	Staking.sol	4e7566893e6e28c0d15a023920ace36a2fc1899537146b1965663a4e120f721e
IUE	UsdtBondDepository.sol	1dd23a4974c60cb28bb541ed4bac03c94e768cb70b296ee6725d02af3e222ec3
LMN	UsdtOpenTokenBondDepository.sol	d62916e8062cde3cf0302fc32521eb91a83154e40edadf1ca03e4490dc55b4d4

Findings

ID	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	Major	Solved
FWJ-01	Unchecked Transfer	Medium	Solved
EJD-01	Logical Issue	Medium	Solved
WED-01	Missing Emit Events Minor	Minor	Solved
KEO-01	Missing Input	Minor	Solved
DFE-01	Missing Input Validation	Minor	Solved
ENQ-01	Redundant Code	Informational	Solved

Major

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization Related Risks	Major		Solved

Description

The owner of the StakingContract contract has the following permissions:

function setAddress()
function setBoolValue()
function rePayRebaseRewardDebt()

The owner of the StakingConfig contract has the following permissions:

function setRebaseRateInfo()
function setStakeBlockCountInfo()
function withdrawToken()
function setUintValue()

The owner of the UsdtOpenTokenBondDepository contract has the following permissions:

function initData()
function initializeBondTerms()
function setBondTerms()
function setAdjustment()
function setStaking()
function setMinBondPrice()
function setVestingTerm()
function setDao()
function setPriceBase()
function setAddress()

The owner of the OpenFiProTreasury contract has the following permissions:

function auditReserves()

function queue()

function toggle()

The owner will have the ability to influence the operation results of the protocol, and even take over
part of the assets managed in the protocol.
Among them, the OpenFiProTreasury contract has implemented a time lock for the operation of the
owner.

Recommendation

This finding describes the level of decentralization of the project, and it is recommended to strengthen security and improve the degree of decentralization from the following aspects:
It is recommended that privileged addresses use multi-signature wallet addresses.
For modification operations that affect protocol operation stability and key business parameters, it is recommended to implement time locks.

Medium

FWJ-01 | Unchecked Transfer Medium

Category	Severity	Location	Status
Unchecked Transfer	Medium	Treasury.sol: 171~190	Solved

Description

A transfer is in progress, but there are no checks to verify its successful completion. If the transfer
somehow fails, the user does not get the reserve tokens, but the debt data has changed.

Recommendation

Use the safeTransfer method of SafeERC20 instead of the transfer method.

EJD-01 | Logical Issue

Category	Severity	Location	Status
Logical Issue	Medium	UsdtOpenTokenBondDepository.sol: 313~341,496~510, 551~555	Solved

Description

If terms.vestingTerm is set to 0, then the vesting of bondInfo is also set to 0 when depositing, which will cause percentVestedFor() to always return 0. When the business logic runs to line 330, because the payout is always 0, the user cannot redeem bond.

Recommendation

Modify the business logic, or prohibit vestingTerm from being set to 0.

Minor

WED-01 | Missing Emit Events

Category	Severity	Location	Status
Missing Emit Events	Minor	UsdtOpenTokenBondDepository.sol: 188~208,561~563	Solved

Description

In line 188, the terms parameter is a key business calculation parameter in the running of the contract. Modifying it will have a great impact on the business running results of the contract. Events should be thrown when such parameters are changed. And when the sub-parameters in the terms parameter structure are modified in other places, an event is thrown, and the event behavior is different.

In line 561, the priceBase parameter is a key business calculation parameter in the running of the contract.
Modifying it will have a great impact on the business running results of the contract. Events should be
thrown when such parameters are changed.

Recommendation

An event is thrown when the priceBase parameter is modified.
An event is thrown when the terms parameter is modified, and the behavior of throwing an event when the terms parameter and its sub-parameters are modified should be consistent.

KEO-01 | Missing Input Validation

Category	Severity	Location	Status
Missing Input Validation	Minor	UsdtOpenTokenBondDepository.sol: 557~559,577~581	Solved

Description

There is no non-zero address check when setting the address.

Recommendation

A non-zero address check on incoming addresses is recommended to prevent accidental occurrences.

DFE-01 | Missing Input Validation

Category	Severity	Location	Status
Missing Input Validation	Minor	Staking.sol:453 ~475	Solved

Description

There is no non-zero address check when setting the address.

Recommendation

A non-zero address check on incoming addresses is recommended to prevent accidental occurrences.

Informational

ENQ-01 | Informational Informational

Category	Severity	Location	Status
Redundant Code	Informational	UsdtOpenTokenBondDepository.sol: 432~447	Solved

Description

The functions bondPrice() and _bondPrice() have the same implementation, and the code is redundant. The _bondPrice() function can be decorated as a view but not decorated.

Recommendation

It is recommended to choose a function to keep, and use the correct modifiers.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of 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 sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

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

Volatile Code

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

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.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

Marl 2023