P2P.ORG SECURITY AUDIT REPORT

December 29, 2022

MixBytes()

TABLE OF CONTENTS

1. Introduction 2

1.1. Disclaimer 2

1.2. Security Assessment Methodology 3

1.3. Project Overview 6

1.4. Project Dashboard 6

2. Findings Report 8

2.1. Critical 8

2.2. High 8

2.3. Medium 8

M-1 Possibility Of The Ownership Transfer To An Uncontrolled Address 8

M-2 Ether Stuck In The `FeeDistributor` 9

2.4. Low 10

L-1 The DepositContract Variable Does Not Have The Optimal Type 10

L-2 Unused ReentrancyGuard Contract 11

L-3 Excess Transformation `IDepositContract` -> `Address` -> `IDepositContract` 12

L-4 No Null Checks For Recipient In S_referrerConfig 13

3. About Mixbytes 14

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

The audited project consist of two repositories.

1. `p2p-eth2-depositor` - this smart contract allows depositing up to 100 ETH2 validator nodes at one transaction. After being funded, the validator nodes will generate fees.
2. `eth-staking-fee-distributor` - these smart contracts manage the distribution of received fees among the service, the client, and (optionally) the referral.

1.4 Project Dashboard

Project Summary

Title	Description
Client	P2P.ORG (P2P Staking)
Project name	ETH2 Depositor & ETH Staking Fee Distributor
Timeline	12 Dec 2022 - 15 Dec 2022
Number of Auditors	3

Project Log

Date	Commit Hash	Note
12.12.2022	9ad1803294a968e9bc4ce9a24e37b0f312a07d0e	initial commit (eth depositor)
12.12.2022	4bac8f636d7ba14244612bcfb9e85f338feba6e3	initial commit (fee distributor)
23.12.2022	99c38b3e025cff1659755b279d3e39142ffd9713	commit with fixes (eth depositor)
23.12.2022	06f029bb27ef97d9955d83c8d92fbb0c6468536e	commit with fixes (fee distributor)

Project Scope

The audit covered the following files:

File name	Link
P2pEth2Depositor.sol	P2pEth2Depositor.sol
Ownable.sol	Ownable.sol
OwnableBase.sol	OwnableBase.sol
Ownable2Step.sol	Ownable2Step.sol
OwnableWithOperator.sol	OwnableWithOperator.sol
AssetRecoverer.sol	AssetRecoverer.sol
OwnableAssetRecoverer.sol	OwnableAssetRecoverer.sol
OwnableTokenRecoverer.sol	OwnableTokenRecoverer.sol
TokenRecoverer.sol	TokenRecoverer.sol

File name	Link
FeeDistributor.sol	FeeDistributor.sol
FeeDistributorFactory.sol	FeeDistributorFactory.sol

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	2
Low	4

ID	Name	Severity	Status
M-1	Possibility of the ownership transfer to an uncontrolled address	Medium	Fixed
M-2	Ether stuck in the `FeeDistributor`	Medium	Fixed
L-1	The depositContract variable does not have the optimal type	Low	Fixed
L-2	Unused ReentrancyGuard contract	Low	Fixed
L-3	Excess transformation `IDepositContract` -> `address` -> `IDepositContract`	Low	Fixed
L-4	No null checks for recipient in s_referrerConfig	Low	Fixed

1.6 Conclusion

We discovered 2 medium, and 4 low severity issues during the audit process. All issues were confirmed and fixed by the client.

File name	Contract deployed on mainnet
P2pEth2Depositor.sol	0x4ca21e4d3a86e7399698f88686f5596dbe74adeb
FeeDistributorFactory.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
OwnableAssetRecoverer.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
OwnableWithOperator.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
OwnableTokenRecoverer.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
AssetRecoverer.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
TokenRecoverer.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
OwnableBase.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
Ownable2Step.sol	0xba1daf11b323dd8c8f84931220918689aaad041a
FeeDistributor.sol	0xe2c725b9ed614be2864892f3b8727ce467bc5954
OwnableTokenRecoverer.sol	0xe2c725b9ed614be2864892f3b8727ce467bc5954
TokenRecoverer.sol	0xe2c725b9ed614be2864892f3b8727ce467bc5954
OwnableBase.sol	0xe2c725b9ed614be2864892f3b8727ce467bc5954

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

M-1	Possibility of the ownership transfer to an uncontrolled address
File	Ownable.sol#L47
Severity	Medium
Status	Fixed in 0013abd8

Description

The Ownable.sol#L47 can be performed to an arbitrary account whether or not it is controlled by the current authority.
After the transfer to an uncontrolled account, the ownership will be lost.

Recommendation

We recommend using a two-step ownership transfer procedure, including the confirmation (accept) of the ownership transfer.

M-2	Ether stuck in the `FeeDistributor`
File	FeeDistributor.sol#L231
Severity	Medium
Status	Fixed in 0013abd8

Description

If one of the fee receivers can't accept ether (i.e the transaction will revert or consume too much gas), no ether can be FeeDistributor.sol#L231.
Although a new instance can be redeployed, some ether may stuck in the recent instance.

Recommendation

We recommend implementing the functionality to migrate from a stuck `FeeDistributor` instance to a new one.

2.4 Low

L-1	The depositContract variable does not have the optimal type
File	P2pEth2Depositor.sol#L15
Severity	Low
Status	Fixed in 99c38b3e

Description

The `depositContract` variable is initialized only in the constructor and not changed any more.
P2pEth2Depositor.sol#L15
The current variable type (mutable) is designed for variables that are supposed to be changed during lifecycle.
Reading such variables consumes more gas than reading immutable variables.

Recommendation

We recommend adding the immutable keyword to the `depositContract` variable.

L-2	Unused ReentrancyGuard contract
File	P2pEth2Depositor.sol#L10
Severity	Low
Status	Fixed in 99c38b3e

Description

At line
P2pEth2Depositor.sol#L10
`P2pEth2Depositor` inherits `ReentrancyGuard`, but the functions of this contract are not used.
Inheritance of unnecessary code generally increases the likelihood of vulnerabilities.

Recommendation

We recommend removing the `ReentrancyGuard` inheritance.

L-3	Excess transformation `IDepositContract` -> `address` -> `IDepositContract`
File	P2pEth2Depositor.sol#L83
Severity	Low
Status	Fixed in 99c38b3e

Description

At line
P2pEth2Depositor.sol#L83
excess transformation can be removed.

Recommendation

It is recommended to replace the line with

```
    depositContract.deposit{value: collateral}(
                pubkeys[i],
                withdrawal_credentials[i],
                signatures[i],
                deposit_data_roots[i]
            );
```

L-4	No null checks for recipient in s_referrerConfig
File	FeeDistributor.sol#L256
Severity	Low
Status	Fixed in 0013abd8

Description

If `recipient` in `s_referrerConfig` is a null address, then there is no need to calculate `referrerAmount`, because `referrerAmount` will always be 0.
Also, `referrerAmount` will be redundant for `serviceAmount`. Additionaly, we can bypass calling the `sendValue` with zero value and null address.

FeeDistributor.sol#L256
FeeDistributor.sol#L244
FeeDistributor.sol#L247

Recommendation

We recommend adding a null check for `recipient` in `s_referrerConfig` in the `withdraw` function and update logic for gas saving.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts


https://github.com/mixbytes/audits_public


https://mixbytes.io/


hello@mixbytes.io


https://twitter.com/mixbytes