P2P.ORG SSV INTEGRATION SECURITY AUDIT REPORT

Jul 03, 2024

MixBytes()

TABLE OF CONTENTS

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 the 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 scope encompasses several smart contracts designed to deposit ETH to validators and integrate with the SSV Network project. The P2pSsvProxyFactory is responsible for deploying the P2pSsvProxy contract, serving as the entry point for creating the initial deposit of validator stakes and their registration in the SSV Network. Meanwhile, the P2pSsvProxy contract is dedicated to interacting with the SSV Network's functions and managing operator clusters.

1.4 Project Dashboard

Project Summary

Title	Description
Client	P2P.org
Project name	SSV Integration
Timeline	November 14 2023 - July 02 2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
14.11.2023	9dd4728002d9c275e29e8ba38bcf7d90efc7531b	Commit for the audit
27.11.2023	88e3ecae57e70410ef0ea482ba10d8f541aeac59	Commit for the re-audit
19.06.2024	cfd0a114c760bc4d87121b1c38893f2110f0d474	Commit for the diff-audit
28.06.2024	dfc3e024a3f0779642b43d39ca321cbcf0eb8605	Commit for the re-audit #2

Project Scope

The audit covered the following files:

File name	Link
src/access/Ownable2Step.sol	Ownable2Step.sol
src/access/OwnableBase.sol	OwnableBase.sol

File name	Link
src/access/Ownable.sol	Ownable.sol
src/access/OwnableWithOperator.sol	OwnableWithOperator.sol
src/assetRecovering/AssetRecoverer.sol	AssetRecoverer.sol
src/assetRecovering/OwnableAssetRecoverer.sol	OwnableAssetRecoverer.sol
src/assetRecovering/OwnableTokenRecoverer.sol	OwnableTokenRecoverer.sol
src/assetRecovering/TokenRecoverer.sol	TokenRecoverer.sol
src/p2pSsvProxy/P2pSsvProxy.sol	P2pSsvProxy.sol
src/p2pSsvProxyFactory/P2pSsvProxyFactory.sol	P2pSsvProxyFactory.sol

Deployments

Commit 88e3ecae57e70410ef0ea482ba10d8f541aeac59 (previous version)

Contract	Address	Comment
P2pSsvProxyFactory	0x10f4ec...83630dfc	
P2pSsvProxy	0xbd057f...163d2365	

Commit dfc3e024a3f0779642b43d39ca321cbcf0eb8605 (latest version)

Contract	Address	Comment
P2pSsvProxyFactory	0xcb924D...373111d9	
P2pSsvProxy	0xec17A0...77c17ED5	

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	1
Medium	1
Low	9

ID	Name	Severity	Status
H-1	`depositEthAndRegisterValidators()` allows unauthorized users to drain the `SsvToken` balance	High	Fixed
M-1	Inability to revoke rights given in `setAllowedSelectorsForClient` and `setAllowedSelectorsForOperator`	Medium	Fixed
L-1	Redundant event emission in `P2pSsvProxy.fallback`	Low	Fixed
L-2	Redundant parameter in `P2pSsvProxy.registerValidators` function	Low	Fixed
L-3	Repeated assignment of the same value in the loop	Low	Fixed
L-4	Mitigating error risks in setting `SsvToken` exchange rate	Low	Acknowledged
L-5	`Unchecked` in `for-loops` is the default Solidity behavior since version 0.8.22	Low	Fixed
L-6	Custom `require` errors are supported by Solidity since version 0.8.26	Low	Acknowledged
L-7	Outdated value in the comment	Low	Fixed

L-8	Duplicate parameter validation in `makeBeaconDepositsAndRegisterValidators`	Low	Fixed
L-9	Suboptimal storage read operations from `s_allowedSsvOperatorIds`	Low	Fixed

1.6 Conclusion

During the audit, 1 high, 1 medium, and 4 low severity findings have been discovered, confirmed, and either fixed or acknowledged by the developers. An acknowledged finding does not impact the overall security of the project.

During the additional diff audit (comparing commit `cfd0a114c760bc4d87121b1c38893f2110f0d474` with base commit `88e3ecae57e70410ef0ea482ba10d8f541aeac59`), the following attack vectors have been checked:

1. Verification of the correctness of adding/removing validators via `bulkRegisterValidator`/`bulkRemoveValidator`:
    - The code review of the `bulkRegisterValidator` and `bulkRemoveValidator` functions confirmed that the logic for adding and removing validators is implemented correctly. The functions include checks to ensure that validators are added and removed as intended. No vulnerabilities or logical errors were identified.
2. Testing the addition of two validators with the same keys:
    - The code analysis to check the handling of validator key uniqueness showed that the code contains mechanisms to prevent the addition of duplicate validators with the same keys. Appropriate error handling is in place. No vulnerabilities were found.
3. Correctness of Access Rights Handling
    - The code review focused on the handling of access rights across various functions and modules. The analysis confirmed that access control mechanisms are correctly implemented, with appropriate checks to ensure that only authorized entities can perform specific actions.

The code review during the diff audit confirms that the examined components and integrations are secure and implemented correctly according to the specified standards. No critical, high or medium severity vulnerabilities were identified during the analysis. Only findings affecting code optimization and readability, but not security, were found.

2. FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

H-1	`depositEthAndRegisterValidators()` allows unauthorized users to drain the `SsvToken` balance
Severity	High
Status	Fixed in 67086f60

Description

This issue has been identified in the P2pSsvProxyFactory.sol#L421 function of the `P2pSsvProxyFactory` contract.
This function permits transferring any specified amount of `SsvToken` to the newly created `SsvCluster` in the `SsvNetwork`. The issue arises from the lack of restrictions on the `SsvPayload.tokenAmount` parameter, enabling users to drain the entire `SsvToken` balance by depositing `32 ETH`. Consequently, these tokens can only be recovered through a multi-step process where the owner has to P2pSsvProxy.sol#L290 the total amount from the `SsvCluster` and then P2pSsvProxy.sol#L306 it from the corresponding `P2pSsvProxy` back to `P2pSsvProxyFactory`. Moreover, the attacker retains control over their deposit and can withdraw it using their specified `withdrawCredentials`.
This vulnerability is classified as `high` severity due to its potential to block deposits from subsequent clients until the owner intervenes.

Recommendation

We recommend limiting the maximum value of `SSVToken` tokens transferred to a reasonable amount to prevent system block due to actions of unprivileged users and to enhance the system's overall security.

2.3 Medium

M-1	Inability to revoke rights given in `setAllowedSelectorsForClient` and `setAllowedSelectorsForOperator`
Severity	Medium
Status	Fixed in b96c5ab8

Description

The issue is found in P2pSsvProxyFactory.sol#L285 and P2pSsvProxyFactory.sol#L304 functions of `P2pSsvProxyFactory` contract.
These functions currently grant access rights for invoking `SsvNetwork` functions directly by `client` and `operator` through P2pSsvProxy.sol#L154-L156 but do not provide a mechanism to revoke these rights.
This shortfall presents a significant security risk, especially in scenarios where excessive permissions are incorrectly assigned by the `owner`. Additionally, the `ssvNetwork` is an upgradeable proxy contract, and the inability to revoke rights in the event of an interface change further increases the vulnerability.
This issue is classified as `medium` due to the risks associated with the irreversibility of incorrectly granted access.

Recommendation

To mitigate this risk, it is recommended to introduce `onlyOwner` functions that enable the revocation of rights for both the `client` and `operator` in invoking specific functions of the `SsvNetwork` through `P2pSsvProxy`.

2.4 Low

L-1	Redundant event emission in `P2pSsvProxy.fallback`
Severity	Low
Status	Fixed in a20c857b

Description

This issue is found in the P2pSsvProxy.sol#L164 function of the `P2pSsvProxy` contract.
The current implementation emits an event before a `revert` operation. However, it's important to note that a transaction revert fully reverses all state changes made during the transaction, including the event log. This renders the event emission redundant and ineffective, as it will be erased upon the execution of `revert`.

Recommendation

We recommend removing this event emission.

L-2	Redundant parameter in `P2pSsvProxy.registerValidators` function
Severity	Low
Status	Fixed in 901b78e1

Description

This issue is found in the P2pSsvProxy.sol#L180 function if the `P2pSsvProxy` contract.
This function, which is only accessible from the `factory`, is invoked within the P2pSsvProxyFactory.sol#L462-L465 function. Before this invocation, a validation check confirms that the `P2pSsvProxy` has been created with a specific `feeDistributorInstance`. This fee distributor is immutable and is assigned as the P2pSsvProxy.sol#L141 storage variable within `P2pSsvProxy`. Given this setup, explicitly passing the `feeDistributor` parameter to `registerValidators` is unnecessary.

Recommendation

We recommend removing the `feeDistributor` parameter from the `registerValidators` function and utilizing the existing `s_feeDistributor` storage variable instead.

L-3	Repeated assignment of the same value in the loop
Severity	Low
Status	Fixed in 20b6a086

Description

A gas efficiency issue is found in the P2pSsvProxyFactory.sol#L625-L628 function of the `P2pSsvProxyFactory` contract.
The problem arises because the `withdrawCredentials` variable is repeatedly assigned the same value within a loop. This repetitive assignment is unnecessary and increases gas costs for executing this function.

Recommendation

We recommend moving the declaration of the `withdrawCredentials` variable outside the loop.

L-4	Mitigating error risks in setting `SsvToken` exchange rate
Severity	Low
Status	Acknowledged

Description

This issue is found in the P2pSsvProxyFactory.sol#L257 function of the `P2pSsvProxyFactory` contract. Currently, the contract owner can arbitrarily set the exchange rate of `SsvToken` to `ETH` within given limits. This poses the risk of potential errors by the owner.

Recommendation

We recommend integrating an oracle for the `SsvToken` to derive its market price from real-time market data.

Client's Commentary

> This is a good idea. But we deliberately decided to do it the simple way and accept the risk to ourselves. It's important that the client is not risking here. They see the exchange rate and proceed only if they agree.

L-5	`Unchecked` in `for-loops` is the default Solidity behavior since version 0.8.22
Severity	Low
Status	Fixed in dfc3e024

Description

While the project is using Solidity version 0.8.24, it is redundant to use `unchecked` incrementing of the `for-loop` variables since Solidity implements such behavior by default starting from version 0.8.22.

Recommendation

We recommend using a standard `for-loop` increment notation for better code readability.

Client's commentary

> Fixed in dfc3e024.

L-6	Custom `require` errors are supported by Solidity since version 0.8.26
Severity	Low
Status	Acknowledged

Description

Currently, custom errors are implemented using the `if(condition) revert(CustomError)` flow, which is the only way to implement custom errors in Solidity versions before 0.8.26. However, the latest version of Solidity already supports the `require(bool, CustomError)` flow.

Recommendation

We recommend considering upgrading Solidity to the latest version and replacing the `if - revert` flow with the `require` flow when it is beneficial for code readability.

Client's commentary

> We prefer staying on 0.8.24 to match SSV Solidity version exactly. Switched to evm_version = 'cancun' for the same reason.

L-7	Outdated value in the comment
Severity	Low
Status	Fixed in dfc3e024

Description

While the constant value `MAX_ALLOWED_SSV_OPERATOR_IDS` was updated from 16 to 24, the corresponding comment `limited to 16 IDs` in the code was not updated.

Related code: P2pSsvProxyFactory.sol#L175

Recommendation

We recommend updating the comment to keep it consistent with the code.

Client's commentary

> Fixed in dfc3e024.

L-8	Duplicate parameter validation in `makeBeaconDepositsAndRegisterValidators`
Severity	Low
Status	Fixed in dfc3e024

Description

The values passed to the `makeBeaconDepositsAndRegisterValidators` function are validated twice: within the function itself and in the `P2POrgUnlimitedEthDepositor.makeBeaconDeposit` function. The validation in `makeBeaconDepositsAndRegisterValidators` can be omitted without compromising code security.

Related code: P2pSsvProxyFactory.sol#L746

Recommendation

We recommend considering to remove the redundant validation from `makeBeaconDepositsAndRegisterValidators`.

Client's commentary

> Fixed in dfc3e024.

L-9	Suboptimal storage read operations from `s_allowedSsvOperatorIds`
Severity	Low
Status	Fixed in dfc3e024

Description

The `s_allowedSsvOperatorIds` array is stored as a static-size array with the size of `MAX_ALLOWED_SSV_OPERATOR_IDS`. To obtain data from that array, the current implementation generally reads the whole array from storage, regardless of whether it uses all its capacity or not. This is a suboptimal solution in terms of gas usage.

Related code:

- P2pSsvProxyFactory.sol#L232
- P2pSsvProxyFactory.sol#L276

Recommendation

We recommend developing a more optimal solution to save gas.

Client's commentary

> Fixed in dfc3e024.

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