P2P.ORG STAKING LENDING PROXY SECURITY AUDIT REPORT

March 20, 2025

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION 2

1.1 Disclaimer 2

1.2 Security Assessment Methodology 2

1.3 Project Overview 6

1.4 Project Dashboard 7

1.5 Summary of findings 9

1.6 Conclusion 10

2.FINDINGS REPORT 11

2.1 Critical 11

2.2 High 11

2.3 Medium 11

2.4 Low 11

L-1 Limited Flexibility of Hardcoded Calldata Validation Rules 11

L-2 Missing Event on `cloneDeterministic` Deployment 13

L-3 Missing Zero-Address Checks in Constructors 14

3. ABOUT MIXBYTES 15

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 Staking Lending Proxy project enables managed interactions with yield protocols through light-weight proxy contracts. `P2pYieldProxyFactory` creates deterministic proxies during first deposits, while `P2pYieldProxy` handles core functionality including deposits, withdrawals, and configurable profit-sharing between clients and treasury. The system includes Ethena protocol integration via specialized adapters (`P2pEthenaProxyFactory` and `P2pEthenaProxy`), with all external interactions constrained by administrator-defined validation rules.

1.4 Project Dashboard

Project Summary

Title	Description
Client	P2P.org
Project name	Staking Lending Proxy
Timeline	14.03.2025 - 19.03.2025
Number of Auditors	3

Project Log

Date	Commit Hash	Note
14.03.2025	8a533a0ddbbae3b31e4f5b1a156377f28d36150a	Initial commit
19.03.2025	c20a12c18c11f90f977500494972935 95f8669d9	Re-audit commit

Project Scope

The audit covered the following files:

File name	Link
src/p2pYieldProxyFactory/P2pYieldProxyFactory.sol	P2pYieldProxyFactory.sol
src/p2pYieldProxy/P2pYieldProxy.sol	P2pYieldProxy.sol
src/adapters/ethena/p2pEthenaProxyFactory/P2pEthenaProxyFactory.sol	P2pEthenaProxyFactory.sol

File name	Link
src/adapters/ethena/p2pEthenaProxy/P2pEthenaProxy.sol	P2pEthenaProxy.sol
src/common/AllowedCalldataChecker.sol	AllowedCalldataChecker.sol
src/common/P2pStructs.sol	P2pStructs.sol

Deployments

File name	Contract deployed on mainnet	Comment
P2pEthenaProxyFactory.sol	0x933f67...659fdd88	
P2pEthenaProxy.sol	0xfcd9cf...a3596ce6	

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	0
Medium	0
Low	3

ID	Name	Severity	Status
L-1	Limited Flexibility of Hardcoded Calldata Validation Rules	Low	Acknowledged
L-2	Missing Event on `cloneDeterministic` Deployment	Low	Fixed
L-3	Missing Zero-Address Checks in Constructors	Low	Fixed

1.6 Conclusion

During our audit, we reviewed several potential vulnerabilities, including:

- **Calldata Validation**: We scrutinized the mechanism restricting calldata for callAnyFunction operations. Since users can send arbitrary calldata, the system relies on administrator-configured rules. We examined whether this framework is adaptable enough to enforce the intended restrictions.
- **PERMIT Front-Run DoS**: We looked into whether the implementation effectively guards against a front-run denial-of-service attack that might disrupt the normal functioning of the permit method.
- **Fee Calculation Accuracy**: We verified that fee computations are performed correctly and assessed the risk of any malicious interference impacting these calculations.
- **Centralization Risks**: We ensured that administrative privileges are appropriately limited, so that no single party holds excessive control beyond what is justified by the system's operational logic.
- **Token Compatibility**: We evaluated how the protocol manages non-standard ERC20 tokens, including those with fee-on-transfer or rebasing features.

The issues discovered during our review are detailed below.

2. FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

L-1	Limited Flexibility of Hardcoded Calldata Validation Rules
Severity	Low
Status	Acknowledged

Description

The protocol integrates with a yield platform that may introduce new user functions in the future. To allow clients to call these yet-unknown functions without bypassing system restrictions, a generic function `callAnyFunction` is implemented. This function validates calldata using a fixed set of configurable rules (`None`, `StartsWith`, `EndsWith`, `AnyCalldata`). However, since the set of validation rules is limited, it might not be flexible enough to enforce more complex restrictions (e.g., ensuring the second argument matches the proxy's address), potentially forcing a trade-off between excessive permissions and insufficient rights.

Recommendation

We recommend offloading the calldata validation to an external validator contract whose address is stored in the factory and can be updated by an operator. This approach would allow for more flexible and future-proof validation rules without needing to modify the core protocol logic.

Client's Commentary

> *Acknowledged. Good suggestion for future versions. Ethena's StakedUSDeV2 contract is not upgradable though. So, we are OK with the existing rules for now.*

L-2	Missing Event on `cloneDeterministic` Deployment
Severity	Low
Status	Fixed in c20a12c1

Description

This issue has been identified within the `_getOrCreateP2pYieldProxy` function of the `P2pYieldProxyFactory` contract.

The `cloneDeterministic` method from OpenZeppelin `Clones` library is used to deploy new proxy instances without emitting an event indicating the creation of a new contract. This reduces transparency and complicates monitoring new contract deployments.

Recommendation

We recommend adding an explicit event emission such as `ProxyCreated(address proxy, address client, uint96 clientBasisPoints)` upon deploying a new proxy instance.

Client's Commentary

> *Fixed in c20a12c1*

L-3	Missing Zero-Address Checks in Constructors
Severity	Low
Status	Fixed in c20a12c1

Description

This issue has been identified within constructors of `P2pEthenaProxyFactory` and `P2pEthenaProxy` contracts.

There are no explicit checks for zero addresses for critical parameters (`_p2pTreasury`, `_stakedUSDeV2`, `_USDe`, `_factory`), potentially leading to non-functional contract deployments.

Recommendation

We recommend explicitly adding checks for zero addresses.

Client's Commentary

> *Fixed in c20a12c1*

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