MixBytes()

P2P.org Staking Lending Proxy Security Audit Report

MAY 19, 2025

Table of Contents

1. Introduction

1.1 Disclaimer

The audit makes no statements or warranties regarding the utility, safety, or security of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other claims about the fitness of the contracts for a particular purpose or their bug-free status.

1.2 Executive Summary

Staking Lending Proxy allows users to earn income by providing liquidity in yield protocols. The current implementation integrates with the Superform protocol. Additionally, users can claim rewards for providing liquidity through P2P.

The audit was conducted over 2 days by 3 auditors, involving a thorough manual code review of the scope and analysis via automated tools.

We analyzed the following potential attack vectors:

Superform Integration Validation
We verified that the proxy correctly integrates with Superform by asserting the supported function selector, decoding the **SingleDirectSingleVaultStateReq** calldata, matching permitted tokens to the liquidity request, and routing both native and ERC-20 assets through Superform in deposit and withdrawal flows with proper error handling – including emergency situations.

Residual Token Balance & Fee Circumvention
We identified and analyzed the path where deposit leftovers remain inside the proxy outside of recorded balances, opening room for accounting manipulation.

Reentrancy
We tested external functions (withdraw, batchClaim) for reentrancy scenarios that could drain or double-spend assets.

Malicious Calldata Bypass
We reviewed deposit, withdraw and batchClaim functions to confirm that only validated calldata can be forwarded to external yield protocols.

Rewards Claim Logic
We checked batch claim flows to ensure protocol fees are calculated once per unique token and user rewards are forwarded accurately.

Profit Calculation Accuracy
We reviewed deposit/withdraw bookkeeping to validate profit thresholds, ceiling division, and edge cases.

ERC1155 Token Handling
We verified that the system correctly implements all required ERC1155 handling mechanisms, ensuring that tokens cannot become stuck and are properly processed throughout all protocol operations.

Fee-on-Transfer and Rebasing Tokens
We confirmed that the protocol does not support fee-on-transfer or rebasing tokens by validating balance consistency before and after each transfer.

Proxy Initialization
We examined the initialization process of contracts deployed using the clones proxy pattern.

Secure Operator Transfer
We verified that the factory contract enforces a secure two-step operator transfer mechanism, restricting initiation to the current owner and requiring explicit acceptance by the pending operator.

Signature Replayability
We confirmed that signatures cannot be reused and are valid only within a specified time window, mitigating replay risks.

The results of our analysis and the corresponding recommendations are detailed in the **Findings** section below.

To improve the overall quality of the code, we also suggest enhancing the following aspects:
- Remove unused imports and unused custom errors to improve code readability.
- Increase test coverage.
- Correct the constructor comment of the P2pSuperformProxy contract to refer to Superform instead of Ethena.

1.3 Project Overview

Summary

Title	Description
Client Name	P2P.org
Project Name	Staking Lending Proxy
Type	Solidity
Platform	EVM
Timeline	06.05.2025 - 13.05.2025

Scope of Audit

File	Link
src/p2pYieldProxyFactory/P2pYieldProxyFactory.sol	P2pYieldProxyFactory.sol
src/access/P2pOperator2Step.sol	P2pOperator2Step.sol
src/access/P2pOperator.sol	P2pOperator.sol
src/p2pYieldProxy/P2pYieldProxy.sol	P2pYieldProxy.sol
src/adapters/superform/DataTypes.sol	DataTypes.sol
src/adapters/superform/p2pSuperformProxyFactory/P2pSuperformProxyFactory.sol	P2pSuperformProxyFactory.sol
src/adapters/superform/p2pSuperformProxy/P2pSuperformProxy.sol	P2pSuperformProxy.sol
src/common/AllowedCalldataChecker.sol	AllowedCalldataChecker.sol

Versions Log

Date	Commit Hash	Note
06.05.2025	677cac6d572df8600d0538df40b1e34cc41025f9	Initial Commit
13.05.2025	b7b2a4ff5b321afa7d9edaddf62953411eab8ff0	Re-audit commit

Mainnet Deployments

File	Address	Blockchain
P2pSuperformProxyFactory.sol	0x105d2f...2cc500ed	Base
P2pSuperformProxy.sol	0x501761...13E2bE98	Base
P2pSuperformProxyFactory.sol	0x3dFf80...aDDB1Cd4	Ethereum
P2pSuperformProxy.sol	0x8ffC81...cD1cEdA5	Ethereum

1.4 Security Assessment Methodology

Project Flow

Stage	Scope of Work
Interim audit	**Project Architecture Review:** · Review project documentation · Conduct a general code review · Perform reverse engineering to analyze the project's architecture based solely on the source code · Develop an independent perspective on the project's architecture · Identify any logical flaws in the design OBJECTIVE: UNDERSTAND THE OVERALL STRUCTURE OF THE PROJECT AND IDENTIFY POTENTIAL SECURITY RISKS.
	Code Review with a Hacker Mindset: · Each team member independently conducts a manual code review, focusing on identifying unique vulnerabilities. · Perform collaborative audits (pair auditing) of the most complex code sections, supervised by the Team Lead. · Develop Proof-of-Concepts (PoCs) and conduct fuzzing tests using tools like Foundry, Hardhat, and BOA to uncover intricate logical flaws. · Review test cases and in-code comments to identify potential weaknesses. OBJECTIVE: IDENTIFY AND ELIMINATE THE MAJORITY OF VULNERABILITIES, INCLUDING THOSE UNIQUE TO THE INDUSTRY.
	Code Review with a Nerd Mindset: · Conduct a manual code review using an internally maintained checklist, regularly updated with insights from past hacks, research, and client audits. · Utilize static analysis tools (e.g., Slither, Mythril) and vulnerability databases (e.g., Solodit) to uncover potential undetected attack vectors. OBJECTIVE: ENSURE COMPREHENSIVE COVERAGE OF ALL KNOWN ATTACK VECTORS DURING THE REVIEW PROCESS.

Stage	Scope of Work
	Consolidation of Auditors' Reports: · Cross-check findings among auditors · Discuss identified issues · Issue an interim audit report for client review OBJECTIVE: COMBINE INTERIM REPORTS FROM ALL AUDITORS INTO A SINGLE COMPREHENSIVE DOCUMENT.
Re-audit	**Bug Fixing & Re-Audit:** · The client addresses the identified issues and provides feedback · Auditors verify the fixes and update their statuses with supporting evidence · A re-audit report is generated and shared with the client OBJECTIVE: VALIDATE THE FIXES AND REASSESS THE CODE TO ENSURE ALL VULNERABILITIES ARE RESOLVED AND NO NEW VULNERABILITIES ARE ADDED.
Final audit	**Final Code Verification & Public Audit Report:** · Verify the final code version against recommendations and their statuses · Check deployed contracts for correct initialization parameters · Confirm that the deployed code matches the audited version · Issue a public audit report, published on our official GitHub repository · Announce the successful audit on our official X account OBJECTIVE: PERFORM A FINAL REVIEW AND ISSUE A PUBLIC REPORT DOCUMENTING THE AUDIT.

1.5 Risk Classification

Severity Level Matrix

Severity	Impact: High	Impact: Medium	Impact: Low
Likehood: High	Critical	High	Medium
Likehood: Medium	High	Medium	Low
Likehood: Low	Medium	Low	Low

Impact

- **High** – Theft from 0.5% OR partial/full blocking of funds (>0.5%) on the contract without the possibility of withdrawal OR loss of user funds (>1%) who interacted with the protocol.
- **Medium** – Contract lock that can only be fixed through a contract upgrade OR one-time theft of rewards or an amount up to 0.5% of the protocol's TVL OR funds lock with the possibility of withdrawal by an admin.
- **Low** – One-time contract lock that can be fixed by the administrator without a contract upgrade.

Likelihood

- **High** – The event has a 50-60% probability of occurring within a year and can be triggered by any actor (e.g., due to a likely market condition that the actor cannot influence).
- **Medium** – An unlikely event (10-20% probability of occurring) that can be triggered by a trusted actor.
- **Low** – A highly unlikely event that can only be triggered by the owner.

Action Required

- **Critical** – Must be fixed as soon as possible.
- **High** – Strongly advised to be fixed to minimize potential risks.
- **Medium** – Recommended to be fixed to enhance security and stability.
- **Low** – Recommended to be fixed to improve overall robustness and effectiveness.

Finding Status

- **Fixed** – The recommended fixes have been implemented in the project code and no longer impact its security.
- **Partially Fixed** – The recommended fixes have been partially implemented, reducing the impact of the finding, but it has not been fully resolved.
- **Acknowledged** – The recommended fixes have not yet been implemented, and the finding remains unresolved or does not require code changes.

1.6 Summary of Findings

Findings Count

Severity	Count
Critical	0
High	1
Medium	1
Low	5

Findings Statuses

ID	Finding	Severity	Status
H-1	Locked Residual Token Amounts on Deposit	High	Fixed
M-1	Risk of Token Freeze in Emergency Situations	Medium	Fixed
L-1	Unused Errors	Low	Fixed
L-2	Deadline Timestamp Should Be Included	Low	Fixed
L-3	`ReentrancyGuard._status` Is Not Initialized in the `P2pYieldProxy.initialize` fucntion.	Low	Fixed
L-4	Redundant Conditions	Low	Fixed
L-5	Non-Generic Parameter Naming in `deposit`	Low	Fixed

2. Findings Report

2.1 Critical

Not Found

2.2 High

H-1	Locked Residual Token Amounts on Deposit		
Severity	High	**Status**	Fixed in b7b2a4ff

Description

In P2pYieldProxy._deposit (ERC-20 branch) the proxy pulls amount tokens from the client but forwards only amountToDepositAfterFee = amount * s_clientBasisPointsOfDeposit / 10 000 to the underlying protocol. The difference (amount - amountToDepositAfterFee) remains in the proxy with no subsequent path for withdrawal or accounting. As deposits and withdrawals track only the forwarded value, this leftover balance becomes inaccessible, resulting in user funds being unintentionally locked.

While the balance is not permanently lost, as governance could whitelist an auxiliary deposit call in AllowedCalldataChecker and then push the leftover through callAnyFunction to a connected yield protocol, doing so requires out-of-band intervention, breaks the current fee-accounting model, and creates a window where funds are effectively frozen and untracked. These factors justify **High** severity.

Recommendation

We recommend forwarding the residual amount to the treasury, ensuring no token balance remains orphaned inside the proxy.

Client's Commentary:

Fixed in b7b2a4ff

2.3 Medium

M-1	Risk of Token Freeze in Emergency Situations		
Severity	Medium	Status	Fixed in b7b2a4ff

Description

The P2pSuperformProxy contract enforces:

```
require(
    req.superformData.receiverAddress == address(this),
    P2pSuperformProxy__ReceiverAddressShouldBeP2pSuperformProxy(
        req.superformData.receiverAddress
            )
        );
```

1. Superform may enter an emergency state during which the normal withdrawal flow is interrupted.
2. In such an emergency, calling withdraw may only enqueue an emergency withdrawal request without actually transferring assets, so the proxy's balance may remain unchanged. If withdraw reverts when the balance delta is zero, it will break the emergency flow and block further recovery.
3. During this emergency flow, tokens can be sent by the Superform administrator to **receiverAddress** (the proxy itself) – a path that never occurs in normal operation – and without a rescue mechanism these assets will become permanently locked.

Recommendation

We recommend:

1. Ensure that withdraw never reverts when the proxy's balance change is zero. Treat a zero-delta result as a successful no-op to preserve the emergency withdrawal queue flow.
2. Implement a client-accessible rescue mechanism to recover any ERC-20 or native tokens held by the proxy as a result of the emergency flow.

Client's Commentary:

Fixed in b7b2a4ff

2.4 Low

L-1	Unused Errors		
Severity	Low	**Status**	Fixed in b7b2a4ff

Description

P2pYieldProxy declares custom errors P2pYieldProxy__ZeroSharesAmount and P2pYieldProxy__NotFactory, yet no function ever reverts with them. Dead declarations bloat bytecode, increase deployment costs, and distract maintainers who must verify their relevance.

Recommendation

We recommend either removing these errors if they are unnecessary, or using them in the appropriate places where the corresponding checks are relevant.

Client's Commentary:

Fixed in b7b2a4ff

L-2	Deadline Timestamp Should Be Included		
Severity	Low	**Status**	Fixed in b7b2a4ff

Description

The p2pSignerSignatureShouldNotExpire modifier P2pYieldProxyFactory.sol#L51 whether the deadline has passed. However, the current implementation excludes the exact deadline timestamp from the valid range, meaning that the signature becomes invalid at the deadline second itself. It is recommended to include the deadline timestamp in the validity check, so that the signature remains valid until and including the specified deadline.

Recommendation

We recommend changing the comparison operator from < to <= to include the deadline timestamp itself.

Client's Commentary:

Fixed in b7b2a4ff

L-3	ReentrancyGuard._status Is Not Initialized in the P2pYieldProxy.initialize fucntion.		
Severity	Low	**Status**	Fixed in b7b2a4ff

Description

In the P2pYieldProxy.initialize function, the _status variable is not set to the _NOT_ENTERED state.

Recommendation

We recommend explicitly initializing _status to _NOT_ENTERED inside the initialize function to improve code clarity and ensure consistency across contracts.

Client's Commentary:

Fixed in b7b2a4ff

L-4	Redundant Conditions		
Severity	Low	**Status**	Fixed in b7b2a4ff

Description

In the P2pYieldProxy.initialize, the P2pYieldProxy.sol#L146-L153 _clientBasisPointsOfDeposit >= 0 and _clientBasisPointsOfProfit >= 0 are redundant because uint48 is an unsigned integer type and cannot be negative.

Recommendation

We recommend removing these redundant conditions and retaining only the upper-bound checks to simplify the code and improve readability.

Client's Commentary:

Fixed in b7b2a4ff

L-5	Non-Generic Parameter Naming in deposit		
Severity	Low	**Status**	Fixed in b7b2a4ff

Description

The external deposit function of P2pYieldProxy names its calldata argument _superformCalldata, binding a generic proxy to a specific adapter conceptually. This diverges from the contract's goal of serving multiple yield protocols and can mislead future integrators.

Recommendation

We recommend renaming the parameter to a protocol-agnostic identifier such as _yieldProtocolDepositCalldata, updating comments and internal references for consistency.

Client's Commentary:
Fixed in b7b2a4ff

3. About MixBytes

MixBytes is a leading provider of smart contract audit and research services, helping blockchain projects enhance security and reliability. Since its inception, MixBytes has been committed to safeguarding the Web3 ecosystem by delivering rigorous security assessments and cutting-edge research tailored to DeFi projects.

Our team comprises highly skilled engineers, security experts, and blockchain researchers with deep expertise in formal verification, smart contract auditing, and protocol research. With proven experience in Web3, MixBytes combines in-depth technical knowledge with a proactive security-first approach.

Why MixBytes

- **Proven Track Record:** Trusted by top-tier blockchain projects like Lido, Aave, Curve, and others, MixBytes has successfully audited and secured billions in digital assets.
- **Technical Expertise:** Our auditors and researchers hold advanced degrees in cryptography, cybersecurity, and distributed systems.
- **Innovative Research:** Our team actively contributes to blockchain security research, sharing knowledge with the community.

Our Services

- **Smart Contract Audits:** A meticulous security assessment of DeFi protocols to prevent vulnerabilities before deployment.
- **Blockchain Research:** In-depth technical research and security modeling for Web3 projects.
- **Custom Security Solutions:** Tailored security frameworks for complex decentralized applications and blockchain ecosystems.

MixBytes is dedicated to securing the future of blockchain technology by delivering unparalleled security expertise and research-driven solutions. Whether you are launching a DeFi protocol or developing an innovative dApp, we are your trusted security partner.

Contact Information

🌐 https://mixbytes.io/

https://github.com/mixbytes/audits_public

✉ hello@mixbytes.io

https://x.com/mixbytes