MixBytes()

Fluid Liquidity Layer Security Audit Report

DECEMBER 10, 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

Instadapp Fluid Liquidity Layer is a core DeFi protocol that serves as the central liquidity hub for the Fluid ecosystem. The protocol combines sophisticated liquidity management, interest rate calculations, and multi-protocol integration capabilities, enabling seamless interaction between various DeFi protocols including lending (fTokens), vaults, flashloans, and DEX operations. The system utilizes advanced mathematical libraries for precise calculations and implements a modular architecture with separate admin and user modules, all built on top of the Instadapp Infinite proxy pattern.

The audit specifically examined changes made between commits `23692b02dc20aa87aa59f2bcd8bb8ec4ad9234bf` and `f5a07116967103946791dffd1fbafa71e0a60828` within the liquidity layer. The initial commit `f5a07116967103946791dffd1fbafa71e0a60828` wasn't audited by MixBytes. Vaults ZIRCUIT and ZtakingPool were out of scope.

Key changes and attack vectors:

- **New variables read/write correctness**: new storage variables were added: `usesConfigs2` in `_exchangePricesAndConfig`, `decayAmount` in `_userSupplyData`, `decayDurationCPs` in `_userSupplyData`, `maxUtilization` in `_configs2`:
  - We verified that the maximum size of the variables does not exceed their allocated slot boundaries, and that the variables are correctly read and written back via respective bit operations.
  - Additionally, it was verified that new variable storage writes are not overwritten by subsequent function calls. For example, in `updateTokenConfigs()`, the `usesConfigs2` is updated in `_exchangePricesAndConfig` storage slot, and then the function `_updateExchangePricesAndRates()` is called, which also modifies `_exchangePricesAndConfig` and it is correctly does not nullify the `usesConfigs2` slot in the storage.
- **Reentrancy protection and DoS**: A new `safeTransferNative()` function was added with a gas limit of `20,000` for native token transfers. This amount should be sufficient for standard receive/fallback functions with event logging, but insufficient for heavy operations, providing partial protection against reentrancy attacks. This also means that integrations with the liquidity layer must avoid implementing complex logic in their receive/fallback functions, as this could lead to DoS where legitimate transfers fail due to gas exhaustion.
- **Negative borrow rate checks**: The `calcRateV1()` and `calcRateV2()` functions were modified to change the `slope_` variable from `uint256` to `int256`. However, negative `slope_` cases (negative borrow rates) are prevented by a specific check. All calculations are wrapped in

unchecked blocks, which is safe due to the limited size of variables participating in the calculations. There are some int(uint(x)-uint(y)) subtractions, which would work the same as int(int(x)-int(y)) due to unchecked{} clause.

- **No storage collisions**: The new maxUtilization field in TokenConfig poses no risk, as the struct is not stored in contract storage, so no collisions can occur when the implementation is upgraded.
- **Token decimals and code validation**: New validation functions _checkIsContract() and _checkTokenDecimalsRange() were added to the admin module. We verified that these functions are properly utilized across all admin functions where contract address validation and token decimal range validation are required. Note that these validations are not present in updateTokenConfigs(), however, this function cannot be called for a token that hasn't had updateRateDataV1s() or updateRateDataV2s() called first, which do contain these validation checks.
- **Utilization above 100%**: A new validation was added to updateTokenConfigs() in adminModule for the case maxUtilization > FOUR_DECIMALS (i.e., prohibition on maximum utilization above 100%).
- **Bit function correctness**: The leastSignificantBit() function was added to the bigMathMinified library. We verified it reverts on zero input, correctly counts trailing zeros using binary search with shifts and masks, and returns the 1-based bit position (1-256). Edge cases were validated (e.g., 1→1, 2→2, 2^255→256).
- **User withdrawal limit validation**: The new updateUserWithdrawalLimit() function was tested for correct validation of arguments and the new limit and input parameters.
- **Withdraw-decay mechanism:** verified arithmetic correctness, dust handling, linear decay across checkpoints (partial/full decay, off-by-one effect at boundaries), limit push-down on withdrawals (never exceeds remaining decay, correct handling of notPushedDown in max-expansion cases), addition of new decay on deposits (weighted duration blending, proper min/cap enforcement), edge cases (before==0, withdrawLimitAfter==0).

Key notes and recommendations:

- We recommend adding more tests for the new features – negative slopes in rate calculations, net transfers, etc.

1.3 Project Overview

Summary

Title	Description
Client Name	Instadapp
Project Name	Fluid Liquidity Layer
Type	Solidity
Platform	EVM
Timeline	09.09.2025 - 10.12.2025

Scope of Audit

File	Link
contracts/libraries/bigMathMinified.sol	bigMathMinified.sol
contracts/libraries/liquidityCalcs.sol	liquidityCalcs.sol
contracts/libraries/liquiditySlotsLink.sol	liquiditySlotsLink.sol
contracts/libraries/safeTransfer.sol	safeTransfer.sol
contracts/liquidity/adminModule/main.sol	main.sol
contracts/liquidity/adminModule/structs.sol	structs.sol
contracts/liquidity/adminModule/events.sol	events.sol
contracts/liquidity/common/variables.sol	variables.sol
contracts/liquidity/common/helpers.sol	helpers.sol
contracts/liquidity/userModule/main.sol	main.sol
contracts/liquidity/userModule/events.sol	events.sol
contracts/liquidity/errorTypes.sol	errorTypes.sol
contracts/liquidity/proxy.sol	proxy.sol
contracts/liquidity/dummyImpl.sol	dummyImpl.sol
contracts/liquidity/error.sol	error.sol
contracts/liquidity/interfaces/iLiquidity.sol	iLiquidity.sol

Versions Log

Date	Commit Hash	Note
09.09.2025	23692b02dc20aa87aa59f2bcd8bb8ec4ad9234bf	Initial Commit
09.12.2025	298c84e3daa9505457cf22265e5c4a8927d7e8c9	Commit for re-audit
10.12.2025	5bd3d775f67d8f8d731c58a3852cdf044b86795b	Commit for re-audit

Mainnet Deployments

File	Address	Blockchain	Comment
(userModule) main.sol	0xF1167F...4e5C92C8	Ethereum Mainnet	L1 and L2 were introduced
(adminModule) main.sol	0x53EFFA...478d261C	Ethereum Mainnet	L1 and L2 were introduced

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	0
Medium	0
Low	2

Findings Statuses

ID	Finding	Severity	Status
L-1	Potential ETH Loss Risk due to MAX_INPUT_AMOUNT_EXCESS Increase to 1000% in the Liquidity Layer	Low	Acknowledged
L-2	Missed preTransferOut()	Low	Fixed

2. Findings Report

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

L-1	Potential ETH Loss Risk due to MAX_INPUT_AMOUNT_EXCESS Increase to 1000% in the Liquidity Layer		
Severity	Low	**Status**	Acknowledged

Description

MAX_INPUT_AMOUNT_EXCESS in the liquidity layer was increased from 1% to 1000%, which creates a risk of ETH loss for the user:

- main.sol#L637

Recommendation

We recommend refunding excess ETH to the user.

Client's Commentary:

Indeed, but note that this only happens for DEX V2 net transfers, where DEX V2 handles the interaction so that excess shouldn't be possible in the first place. Any excess from the user is sent back to the user at the DEX V2 protocol level.

L-2	Missed preTransferOut()		
Severity	Low	**Status**	Fixed in 5bd3d775

Description

The preTransferOut() hook was missed for two safe transfers:

- main.sol#L1092
- main.sol#L1115

Recommendation

We recommend calling the hook in the aforementioned cases.

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