Yieldoor Audit Report

Prepared by 0xSimao

Version 2.0

November 30, 2025

Contents

1 About 0xSimao

0xSimao is an indepedent security researcher, #2 on Sherlock, top-ranked on Cantina and Code4rena, and a member of Blackthorn, a leading auditing firm. Previously, he served as Head of Security at Three Sigma.

0xSimao has placed in the Top 3 in 28 public audits and has led over 60 private engagements. For private audits or collaboration opportunities, feel free to reach out to him on X (@0xSimao), Telegram (@OxSimao), or Discord (@0xSimao).

2 Disclaimer

0xSimao makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Loop Vaults are designed for users who want to earn high fixed yields by leveraging interest rate arbitrage in a fully automated and capital-efficient manner. Built for stablecoin holders, these vaults harness advanced DeFi primitives like Pendle's PT tokens and Morpho's lending markets to generate leveraged returns—without requiring users to manage positions manually.

5 Audit Scope

The scope was the addition of the withdrawal queue.

6 Executive Summary

Over the course of 4 days, 0xSimao conducted an audit on the Yieldoor smart contracts provided by Yieldoor. In this period, a total of 7 issues were found.

Summary

Project Name	Yieldoor
Repository	loopedVault
Commit	940648c0c27a...
Fix Commit	2a378db6c635...
Audit Timeline	June 23rd - June 26th
Methods	Manual Review, Stateful Fuzzing

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	7
Informational	0
Gas Optimizations	0
Total Issues	7

Summary of Findings

[L-1] Missing nonReentrant functionality in some execution paths	Resolved
[L-2] It's possible for debt to be stuck in the LoopedVault when rebalancing from Aave to Morpho	Acknowledged
[L-3] CEI pattern is not followed on LoopedVault::fulfillWithdraw()	Resolved
[L-4] Slippage protection could be added to LoopedVault::requestWithdraw() in case of black swan event	Acknowledged
[L-5] Missing key events when requesting withdrawals that could be important for the allocator bot	Resolved
[L-6] LoopedVault::maxMint() and LoopedVault::maxRedeem() are not overriden	Resolved
[L-7] User may be requesting/fulfilling withdrawals at better rates than the real share/asset ratio	Acknowledged

7 Findings

7.1 Low Risk

7.1.1 Missing nonReentrant functionality in some execution paths

Description: `LoopedVault::rebalance()` is missing the nonReentrant modifier. `LoopedVault::requestWithdraw()` flow when there are not enough idle funds also never checks for reentrancy. It only does when withdrawing as it calls `LoopedVault::_withdraw()`, which does have the modifier. `LoopedVault::fulfillWithdraw()` is missing the `nonReentrant` modifier.

Recommended Mitigation: Add the `nonReentrant` modifier to `LoopedVault::requestWithdraw()` and remove it from `LoopedVault::_withdraw()`.

0xSimao:

Fixed in #2a378db.

7.1.2 It's possible for debt to be stuck in the LoopedVault when rebalancing from Aave to Morpho

Description: `RebalanceLogic::aaveToMorpho()` borrows from Morpho and repays Aave debt. If trying to repay more Aave debt than the available, the function will go through but borrow tokens will be left in the `LoopedVault`. This is not an issue in case the borrow token is $USDC, the asset of the vault, as it is accounted for as `LoopedVault::idleBalance()`. However, if the borrowed token is for example $DAI, it will not be accounted for and cause issues.

Recommended Mitigation: Revert if trying to repay more debt than the available (default Morpho behaviour).

7.1.3 CEI pattern is not followed on LoopedVault::fulfillWithdraw()

Description: `LoopedVault::fulfillWithdraw()` transfers the asset before updating the state, breaking the CEI pattern.

Recommended Mitigation: Transfer the assets at the end of the function.

0xSimao:

Fixed in #2a378db.

7.1.4 Slippage protection could be added to LoopedVault::requestWithdraw() in case of black swan event

Description: It's technically possible for the assets of the `LoopedVault` to decrease suddenly when requesting a withdrawal in between a vesting interval update and a black swan event. This could catch a user withdrawing off guard and make them take a significant loss in case they are frontrun.

Recommended Mitigation: Consider adding slippage protection.

7.1.5 Missing key events when requesting withdrawals that could be important for the allocator bot

Description: `LoopedVault::requestWithdraw()` could emit an event when requesting assets, though it's possible to fetch this information by listening for shares burned.

Recommended Mitigation: Consider adding an event.

0xSimao:

Fixed in #2a378db.

7.1.6 LoopedVault::maxMint() and LoopedVault::maxRedeem() are not overriden

Description: `LoopedVault::maxMint()` is not overriden and does not comply with `ERC4626` as there is a deposit cap. `LoopedVault::maxRedeem()` is also not overriden, though this functionality has been removed and is now 2 step, which means that it has no impact.

Recommended Mitigation: Consider overriding maxMint().

0xSimao:

Fixed in #2a378db.

7.1.7 User may be requesting/fulfilling withdrawals at better rates than the real share/asset ratio

Description: `LoopedVault::requestWithdraw()` burns shares as per the current `totalAssets()`, which uses a cached `lastTotalAssets`. In case the LoopedVault is liquidated or its value simply lowers (collateral PT price goes down), but it's in the middle of a vesting interval, this will not be reflected on `totalAssets()`, and users will withdraw at a better rate than they really should. The same happens on `LoopedVault::fulfillWithdraw()`.

Recommended Mitigation: Optimally, the ratio used on withdrawals should be the worse one for the user, so ideally it should calculate the real total assets and use them in the shares calculation if it is lower. The same should also be compared when fulfilling withdrawals.