Loop Vaults Security Review

Pashov Audit Group

Conducted by: DadeKuma, Koolex, Shaka, 0xAristos, Pascal, seeques

April 30th 2025 - May 3rd 2025

Contents

1. About Pashov Audit Group 2
2. Disclaimer 2
3. Introduction 2
4. About Loop Vaults 2
5. Risk Classification 3
   5.1. Impact 3
   5.2. Likelihood 3
   5.3. Action required for severity levels 4
6. Security Assessment Summary 4
7. Executive Summary 5
8. Findings 7
   8.1. High Findings 7
      [H-01] Incorrect vesting interest calculation enables MEV attacks 7
   8.2. Low Findings 9
      [L-01] Timelock can be bypassed even when enabled 9
      [L-02] Protocols fail to integrate with vault, causing fund loss 9
      [L-03] Compromised allocator may drain vault via position cycling 10
      [L-04] Operations that modify the exchange rate can be frontrun 10

1. About Pashov Audit Group

Pashov Audit Group consists of multiple teams of some of the best smart contract security researchers in the space. Having a combined reported security vulnerabilities count of over 1000, the group strives to create the absolute very best audit journey possible - although 100% security can never be guaranteed, we do guarantee the best efforts of our experienced researchers for your blockchain protocol. Check our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **spacegliderrrr/loopedVault** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Loop Vaults

Looped is a vault system where user deposits are managed by an off-chain allocator to perform leveraged looping strategies using Morpho and Pendle PTs. By repeatedly borrowing and swapping to increase exposure, users aim to earn fixed interest above borrowing costs.

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash - 08253ac43834c95019ce0ad884d3eb50d8351d6a

fixes review commit hash - 5818614c023a8eb339c2cc44c707de5edef3f6fb

Scope

The following smart contracts were in scope of the audit:

- `LoopedVaults`

7. Executive Summary

Over the course of the security review, DadeKuma, Koolex, Shaka, 0xAristos, Pascal, seeques engaged with Loop Vaults to review Loop Vaults. In this period of time a total of **5** issues were uncovered.

Protocol Summary

Protocol Name	Loop Vaults
Repository	https://github.com/spacegliderrrr/loopedVault
Date	April 30th 2025 - May 3rd 2025
Protocol Type	Yield strategy

Findings Count

Severity	**Amount**
High	1
Low	4
Total Findings	**5**

Summary of Findings

ID	Title	Severity	Status
[H-01]	Incorrect vesting interest calculation enables MEV attacks	High	Resolved
[L-01]	Timelock can be bypassed even when enabled	Low	Resolved
[L-02]	Protocols fail to integrate with vault, causing fund loss	Low	Acknowledged
[L-03]	Compromised allocator may drain vault via position cycling	Low	Acknowledged
[L-04]	Operations that modify the exchange rate can be frontrun	Low	Acknowledged

8. Findings

8.1. High Findings

[H-01] Incorrect vesting interest calculation enables MEV attacks

Severity

Impact: Medium

Likelihood: High

Description

Given that the value of the positions is updated only when `updateInterval` time has passed, the interest is vested to prevent MEV attacks.

However, the implementation of `_vestingInterest()` is incorrect, as it returns 0 when `block.timestamp == lastUpdate` and increases linearly until `vestingDuration` is reached. This means that calling `totalAssets()` just after an update will include all the interest accrued, which makes the update subject to MEV attacks.

```
function totalAssets() public view override returns (uint256) {
        return lastTotalAssets - _vestingInterest();
    }

    function _vestingInterest() internal view returns (uint256) {
        if (block.timestamp - lastUpdate >= vestingDuration) return 0;

        uint256 __vestingInterest =
          (block.timestamp - lastUpdate) * vestingInterest / vestingDuration;
        return __vestingInterest;
    }
```

Recommendations

```
-       uint256 __vestingInterest =
- (block.timestamp - lastUpdate) * vestingInterest / vestingDuration;
+       uint256 __vestingInterest = (vestingDuration -
+ (block.timestamp - lastUpdate)) * vestingInterest / vestingDuration;
```

8.2. Low Findings

[L-01] Timelock can be bypassed even when enabled

Supposing that timelocks are enabled some time after the deploy, the timelock can still be skipped, as `addMarket` doesn't check if `scheduleAdd` is initialized.

As such, new markets can be added without calling `scheduleAddMarket` and waiting for the timelock.

Consider checking that `scheduledAdd[id] != 0`.

[L-02] Protocols fail to integrate with vault, causing fund loss

The `_deposit` and `_withdraw` functions are limited by `depositCap` and `idleBalance` respectively, but these limitations are not implemented in the ERC4626 functions `maxDeposit` and `maxWithdraw`.

This could cause integration compatibility issues with other protocols that expect a compliant vault, particularly if they rely on `maxDeposit` and `maxWithdraw` to determine the permissible deposit/withdrawal amounts.

Impact: Indirect loss of protocol fees if other protocols cannot integrate with the contract, or locked funds in some edge cases.

Likelihood: While using a wrapper could avoid this issue, it might be impossible with some protocols that expect a compliant vault.

Recommendations:

Consider overriding `maxDeposit` and `maxWithdraw` to enforce the same limits used in `_deposit` and `_withdraw`, ensuring the contract properly reports the depositable/withdrawable amounts.

[L-03] Compromised allocator may drain vault via position cycling

A compromised allocator can drain vault funds by repeatedly opening and closing positions to extract value through legitimate slippage allowances. With configurable slippage protection (`positionMaxSlippage`, default 1%), an allocator can execute cycles where:

1. Opening position: lose X% on borrowed token to collateral swaps.
2. Closing position: lose Y% on collateral to borrowed token swaps, plus additional slippage if borrowed asset differs from vault asset.

Combined cycle loss approximates 2X-3X the allowed slippage percentage. With 15 enabled markets and no rate limiting, multiple rapid cycles could drain significant vault value.

Recommendations:

Implement rate limiting on position operations (e.g., minimum time between open/close cycles per market).

[L-04] Operations that modify the exchange rate can be frontrun

`onMorphoSupplyCollateral()` and `onMorphoRepay()` update the `lastTotalAssets` variable with the net profit or loss resulting from the operation. This will instantly modify the exchange rate for shares to assets, which opens the door for frontrunning attacks.

In the case of an increase in the value of the assets, an attacker can make a deposit just before the allocator's call and redeem the shares just after, profiting from the difference in the exchange rate.

In the case of a decrease in the value of the assets, shareholders can redeem their shares just before the allocator's call to avoid incurring losses, increasing the losses for the rest of the shareholders.

Proof of concept

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import "forge-std/Test.sol";
import "../src/LoopedVaults.sol";
import "./utils/MockPricefeed.sol";

contract AuditTest is Test {
    using MarketParamsLib for MarketParams;

    address morpho = 0xBBBBBbbBBb9cC5e90e3b3Af64bdAF62C37EEFFCb;
    address pendlerouter = 0x888888888889758F76e7103c6CbF23ABbF58F946;
    address usdc = 0x833589fCD6eDb6E08f4c7C32D4f71b54bdA02913;
    address pricefeed;

    LoopedVault vault;
    address feeRecipient = makeAddr("FeeRecipient");
    address allocator = makeAddr("Allocator");
    address alice = makeAddr("Alice");
    address bob = makeAddr("Bob");
    address ptUsr = 0xa6F0A4D18B6f6DdD408936e81b7b3A8BEFA18e77;

    MarketParams mp;

    function setUp() public {
        vm.createSelectFork("https://mainnet.base.org");

        pricefeed = address(new MockPricefeed());
        MockPricefeed(pricefeed).setPrice(usdc, 1e18);
        MockPricefeed(pricefeed).setPrice(ptUsr, 1e18);

        vault = new LoopedVault
          (usdc, pendlerouter, morpho, feeRecipient, allocator, pricefeed);

        mp.collateralToken = ptUsr;
        mp.loanToken = usdc;
        mp.oracle = 0x6AdeD60f115bD6244ff4be46f84149bA758D9085;
        mp.irm = 0x46415998764C29aB2a25CbeA6254146D50D22687;
        mp.lltv = 915000000000000000;

        vault.scheduleAddMarket(mp.id());
        vault.addMarket(mp);

        deal(usdc, alice, 10_000e6);
        deal(usdc, bob, 10_000e6);

        vm.prank(alice);
        IERC20(usdc).approve(address(vault), type(uint256).max);
        vm.prank(bob);
        IERC20(usdc).approve(address(vault), type(uint256).max);
    }

    function test_frontRunAllocator() public {
        vm.prank(alice);
        vault.deposit(100e6, alice);

        skip(7 days);

        MorphoLoop memory ml;
        ml.marketParams = mp;
        ml.amountToSupply = 5e18;
        ml.amountToBorrow = 4e6;

        SwapData memory swapData;
        swapData.swapType = SwapType(1);
        swapData.extRouter = 0x6131B5fae19EA4f9D964eAc0408E4408b66337b5;
```

```
                swapData.extCalldata = hex"e21fd0e900000000000000000000000000000000000

        TokenInput memory input;
        input.tokenIn = usdc;
        input.netTokenIn = 5e6;
        input.tokenMintSy = 0x35E5dB674D8e93a03d814FA0ADa70731efe8a4b9;
        input.pendleSwap = 0x313e7Ef7d52f5C10aC04ebaa4d33CDc68634c212;
        input.swapData = swapData;

        ml.input = input;
        ml.SY = 0x4665d514e82B2F9c78Fa2B984e450F33d9efc842;
        ml.pendleMarket = 0x715509Bde846104cF2cCeBF6fdF7eF1BB874Bc45;

        // Bob is checking the mempool and notices that a tx will increase the
        // value of the shares,
        // so he front-runs it with a deposit
        vm.prank(bob);
        uint256 bobDepositAmount = 10_000e6;
        uint256 bobShares = vault.deposit(bobDepositAmount, bob);

        vm.prank(allocator);
        vault.loopToMorpho(ml);

        // Bob withdraws his shares making an instant profit, which would
        // correpond to Alice if he
        // didn't front-run the allocator's tx
        vm.prank(bob);
        uint256 assetsReceived = vault.redeem(bobShares, bob, bob);
        assert(assetsReceived > bobDepositAmount);
    }
}
```

Recommendations

- Use a vesting mechanism for profits resulting from the allocator's operations.
- Add a cooldown period for withdrawals.