YieldFi PR19: vyToken Audit Report

Prepared by Cyfrin

Version 2.2

Lead Auditors

Immeas

June 17, 2025

Contents

1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

2 Disclaimer

The Cyfrin team 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

The new vyToken lets users earn additional yield by depositing underlying assets (like USDC), which are converted into yToken and then deployed into DeFi strategies. It acts as a higher-yield wrapper around yToken, with its own exchange rate and yield distribution.

5 Audit Scope

The changes in PR#19:

```
bridge/BridgeMB.sol
bridge/ccip/BridgeCCIP.sol
core/Manager.sol
core/l1/LockBox.sol
core/l1/Yield.sol
core/tokens/YToken.sol
core/tokens/YTokenL2.sol
core/tokens/dYTokenL1.sol
core/tokens/dYTokenL2.sol
```

6 Executive Summary

Over the course of 2 days, the Cyfrin team conducted an audit on the YieldFi PR19: vyToken smart contracts provided by YieldFi. In this period, a total of 5 issues were found.

No severe issues were identified during the audit. The codebase was well-designed and thoroughly tested. Two informational findings were reported: one regarding an unused function parameter, and another noting an unnecessary `virtual` modifier on a non-overridden function. Additionally, a minor gas optimization opportunity was identified.

During the mitigation phase, an additional commit, `6203b40`, was made to remove an unnecessary event, which was determined to pose no risk.

After the audit, two further commits were added:

- `b69e386`, which renamed `dYToken` to `vyToken`
- `2b32311`, which added an extra argument to the `AssetAndShareManage` event

Both changes were reviewed and determined to be safe.

A critical issue involving the over-minting of yUSD was discovered during post-deployment. When vyUSD was minted via the Manager contract, it ended up minting an abnormally large amount of yUSD due to a bug in how asset accounting was handled. This was fixed in commits `04f2c15`, `e43fa02`.

The above commit also introduced a new feature where smaller withdrawals could be done instantly if there's assets available in the Manager contract. One low and one informational finding were added from these changes which were both acknowledged by the protocol.

Summary

Project Name	YieldFi PR19: vyToken
Repository	contracts
Commit	702a931df3ad...
Audit Timeline	May 26th - May 27th, 2025
Methods	Manual Review

Issues Found

Critical Risk	0
High Risk	0
Medium Risk	0
Low Risk	1
Informational	3
Gas Optimizations	1
Total Issues	5

Summary of Findings

[L-1] Instant withdrawals via Manager bypass withdrawal fee	Acknowledged
[I-1] `isNewYToken` can be omitted in YToken contracts	Resolved
[I-2] Redundant `virtual` declaration in `YToken::_withdraw`	Acknowledged
[I-3] Price Change Sensitivity in Instant Withdrawals	Acknowledged
[G-1] Avoid unnecessary computation in `dYToken::mintYToken` when `isNewYToken == false`	Resolved

7 Findings

7.1 Low Risk

7.1.1 Instant withdrawals via Manager bypass withdrawal fee

Description: When a user performs a withdrawal, if the amount is small enough and sufficient assets are available in the `Manager`, the withdrawal can be completed instantly in `Manager::redeem`:

```
// if redeeming yToken.asset() and vaultAssetAmount is less than maxRedeemCap and balance of contract
↪  is greater than vaultAssetAmount, redeem immediately and return
if (_asset == IERC4626(_yToken).asset() && vaultAssetAmount <= maxRedeemCap[_yToken] &&
↪  IERC20(_asset).balanceOf(address(this)) >= vaultAssetAmount) {
    IERC20(_asset).safeTransfer(_receiver, vaultAssetAmount);
    emit InstantRedeem(caller, _yToken, _asset, _receiver, vaultAssetAmount);
    return;
}
```

However, this bypasses the fee applied in the asynchronous `_withdraw` flow.

Impact: Withdrawals can be initiated through both the `ERC4626` YToken vaults and directly via the `Manager` contract. This allows users to circumvent the fee applied in the YToken withdrawal path by opting for instant withdrawals directly through the `Manager`.

Recommended Mitigation: Consider taking the fee also in the instant withdrawal flow.

YieldFi: Acknowledged. Currently fees are set to 0 hence this doesn't affect the protocol fees.

7.2 Informational

7.2.1 `isNewYToken` can be omitted in YToken contracts

Description: To support the accounting of underlying assets, a new parameter `isNewYToken` was introduced in `mintYToken`. This parameter is used in the `dYTokenL1::mintYToken` and `dYTokenL2::mintYToken` contracts to determine whether minting `dYTokens` should also update the balances of the underlying `YTokens`.

However, the parameter is unused in the `YToken` and `YTokenL2` implementations:

```
function mintYToken(address to, uint256 shares, bool isNewYToken) external virtual {
    require(msg.sender == manager, "!manager");
    _mint(to, shares);
}
```

Consider omitting the parameter to make its unused status explicit:

```
- function mintYToken(address to, uint256 shares, bool isNewYToken) external virtual {
+ function mintYToken(address to, uint256 shares, bool ) external virtual {
```

YieldFi: Fixed in commit a3a9bad

Cyfrin: Verified. `isNewYToken` is now removed from the above function parameter declarations.

7.2.2 Redundant `virtual` declaration in `YToken::_withdraw`

Description: In the pull request, the function `YToken::_withdraw` was updated to be declared `virtual`, allowing it to be overridden in derived contracts. However, it is never actually overridden in any of the `dYToken` implementations.

Consider removing the `virtual` modifier from both `YToken::_withdraw` and `YTokenL2::_withdraw` to clarify intent and avoid misleading extensibility.

YieldFi: Acknowledged.

7.2.3 Price Change Sensitivity in Instant Withdrawals

Description: Since instant withdrawals allow users to withdraw the underlying asset immediately, they can potentially react to price changes in a way that introduces economic inefficiencies. Because prices are delivered via an oracle on L2, this creates two potential vectors for abuse:

1. Cross-chain arbitrage: Large price discrepancies between L1 and L2 can be exploited by users performing arbitrage across chains.
2. Sandwiching price updates: If a user observes a significant price movement, they can deposit just before the change and withdraw immediately after—capturing the gain at the expense of existing holders. This also works in reverse: a user can withdraw just before a large drop, avoiding losses that others would bear.

This behavior is already mitigated to some extent by the cap on instant withdrawals, which limits the amount a user can redeem at once, and by keeping only a limited balance available for instant redemptions.

However, it may be beneficial to monitor the price delta between L1 and L2 or to detect significant price swings. In such cases, consider pausing the contract to prevent potential abuse during volatile conditions.

YieldFi: Acknowledged. The price differences between L1 and L2, as well as short-term price movements, are typically small. Under normal conditions, this behavior is unlikely to be profitable. In the case of a catastrophic event, the affected vaults can be paused while changes are addressed.

7.3 Gas Optimization

7.3.1 Avoid unnecessary computation in `dYToken::mintYToken` when `isNewYToken == false`

Description: In the new `dYToken::mintYToken`, there is special logic for handling newly minted `dYTokens`, i.e., tokens generated through deposits or accrued fees:

```
function mintYToken(address to, uint256 shares, bool isNewYToken) external override {
    require(msg.sender == manager, "!manager");
    uint256 assets = convertToAssets(shares);

    // if isNewYToken i.e external deposit has triggered minting of dyToken, we mint yToken to this
    ↪   contract
    if(isNewYToken) {
        // corresponding shares of yToken based on assets
        uint256 yShares = YToken(yToken).convertToShares(assets);
        // can pass isNewYToken here as it is not used in yToken
        ManageAssetAndShares memory manageAssetAndShares = ManageAssetAndShares({
            yToken: yToken,
            shares: yShares,
            assetAmount: assets,
            updateAsset: true,
            isMint: true,
            isNewYToken: isNewYToken
        });
        IManager(manager).manageAssetAndShares(address(this), manageAssetAndShares);
    }
    // minting dYToken to receiver
    _mint(to, shares);
}
```

The `assets` variable is only used within the `if (isNewYToken)` block. Moving its declaration inside the block would save gas when `isNewYToken == false`, by avoiding unnecessary computation:

```
function mintYToken(address to, uint256 shares, bool isNewYToken) external override {
    require(msg.sender == manager, "!manager");
-   uint256 assets = convertToAssets(shares);

    // if isNewYToken i.e external deposit has triggered minting of dyToken, we mint yToken to this
    ↪   contract
    if(isNewYToken) {
+       uint256 assets = convertToAssets(shares);
        // corresponding shares of yToken based on assets
        uint256 yShares = YToken(yToken).convertToShares(assets);
```

YieldFi: Fixed in commit `f1f6996`

Cyfrin: Verified. `convertToAssets` now moved inside the if-statmement.