CANTINA

DeFi Integrations v1.2.2

Security Review

Cantina Managed review by:

Optimum, Lead Security Researcher
Akshay Srivastav, Security Researcher

April 10, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Kiln is a staking platform you can use to stake directly, or whitelabel staking into your product. It enables users to stake crypto assets, manually or programmatically, while maintaining custody of your funds in your existing solution, such Fireblocks, Copper, or Ledger.

From Apr 4th to Apr 5th the Cantina team conducted a review of defi-integrations-v1.2.2 on commit hash 9df7d77f. The team identified a total of **9** issues:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	1	1	0
Low Risk	4	2	2
Gas Optimizations	0	0	0
Informational	4	0	4
Total	**9**	**3**	**6**

3 Findings

3.1 Medium Risk

3.1.1 `MetamorphoConnector/VenusConnector.reinvest()` will always revert when `swapTarget` is `address(0)`

Severity: Medium Risk

Context: MetamorphoConnector.sol#L102

Description: The `constructor` of `MetamorphoConnector`, `VenusConnector` contracts accepts `address(0)` as the `swapTarget`. The `reinvest` function is also expected to work when `swapTarget` is `address(0)`.

However when `swapTarget` is `address(0)` the `reinvest` function still tries to provide ERC20 token approval of `rewardsAsset` to `address(0)` which will likely lead to a `revert` as most ERC20 tokens do not allow approval to `address(0)`. See OpenZeppelin's `ERC20.approve()`.

Impact: Due to this the connector's `reinvest` function will always revert when `swapTarget` is `address(0)`. This is likely bound to happen when `MorphoDistributor` distributes the rewards in asset tokens of Kiln vault (`rewardToken == vault.asset()`) so there is no need to swap reward tokens into asset tokens. In that scenario reinvesting logic will fail.

Recommendation: Consider adding this change:

```diff
- // Approve the swap target
- rewardsAsset.forceApprove(address(swapTarget), type(uint256).max);

  // Swap the rewardsAsset to the underlying asset
  if (swapPayload.length != 0 && swapTarget != address(0)) {
+     // Approve the swap target
+     rewardsAsset.forceApprove(address(swapTarget), type(uint256).max);
      swapTarget.functionCall(swapPayload);
  }
  // ...
```

Kiln: Fixed in 2a0f82cf by enforcing `swapTarget` is non-zero in the constructor.

Cantina Managed: Fix verified.

3.2 Low Risk

3.2.1 Returned data of multiple external calls is ignored

Severity: Low Risk

Context: MetamorphoConnector.sol#L60, MetamorphoConnector.sol#L65, MetamorphoConnector.sol#L77, MetamorphoConnector.sol#L98, MetamorphoConnector.sol#L106, MetamorphoConnector.sol#L114

Description: The connector smart contracts perform multiple external calls to other contracts, but at various instances the returned data of those external calls is blindly ignored. Some of those instances are:

1. `MetamorphoConnector.sol#L60`: Returned data of `metamorpho.deposit` call is ignored.
2. `MetamorphoConnector.sol#L65`: Returned data of `metamorpho.withdraw` call is ignored.
3. `MetamorphoConnector.sol#L77`: Returned data of `distributor.claim` call is ignored.
4. `MetamorphoConnector.sol#L98`: Returned data of `distributor.claim` call is ignored.
5. `MetamorphoConnector.sol#L106`: Returned data of `swapTarget.functionCall` call is ignored.
6. `MetamorphoConnector.sol#L114`: Returned data of `metamorpho.deposit` call is ignored.

Recommendation: It is always advised to explicitly verify returned data of external calls. So either verify the returned data or add an explicit comment on each instance explaining the reason for ignored returned data.

Kiln: Acknowledged. The functions we interact with have explicit revert conditions in case of errors, and we do not need to use the returned data.

Cantina Managed: Acknowledged.

3.2.2 `Vault.claimAdditionalRewards()` call can be forcefully reverted

Severity: Low Risk

Context: MetamorphoConnector.sol#L77, MetamorphoConnector.sol#L98

Description: The `MorphoDistributor.claim()` call can revert. On Morpho's `UniversalRewardsDistributor` contract anyone can claim rewards on behalf of an account. When the `CLAIM_MANAGER_ROLE` initiates a claim or reinvest call (via `claimAdditionalRewards`), any malicious user can extract the transaction's input values (merkle proofs) and can initiate the claim for Kiln `Vault`, essentially front-running the `CLAIM_MANAGER_ROLE`'s txn. As claiming the same rewards twice is not allowed on `UniversalRewardsDistributor`, the `CLAIM_MANAGER_ROLE`'s original `claimAdditionalRewards` txn will get reverted.

After this to reinvest the claimed rewards the `CLAIM_MANAGER_ROLE` will need to initiate a new `claimAdditionalRewards` call with `RewardClaimInfo.distributor` address as `0`. Hence causing minor inconvenience to normal vault operations.

Recommendation: Consider wrapping the `MorphoDistributor.claim()` call in a `try-catch` block.

```
if (address(claimInfo.distributor) != address(0)) {
    try claimInfo.distributor.claim(...) catch {}
}
```

Kiln: Acknowledged. No economic incentives for the people reverting our TX. We just have to re-call without claiming.

Cantina Managed: Acknowledged.

3.2.3 Discrepancy in Morpho's `UniversalRewardsDistributor` interface

Severity: Low Risk

Context: MetamorphoConnector.sol#L21-L23

Description: The Morpho's `UniversalRewardsDistributor.claim` function has this interface:

```
function claim(address account, address reward, uint256 claimable, bytes32[] memory proof)
    external
    returns (uint256 amount);
```

But the `claim` function interface of `MorphoDistributor` present in `MetamorphoConnector.sol` is:

```
function claim(address account, address reward, uint256 claimbale, bytes32[] calldata proof) external;
```

which does not includes the `uint256` data returned by the `claim` function. Also the spelling of `claimbale` parameter is incorrect, it should be `claimable`.

Recommendation: Consider correcting the `MorphoDistributor` interface by including the return value.

Kiln: Fixed in 2a0f82cf by implementing the reviewer's recommendation.

Cantina Managed: Fix verified.

3.2.4 Unsafe infinite approval granted to `swapTarget`

Severity: Low Risk

Context: MetamorphoConnector.sol#L102

Description: The `MetamorphoConnector.reinvest` function grants infinite approval for `swapTarget` address for any reward token that the `CLAIM_MANAGER_ROLE` provides as input. The reward token can be `Vault.asset()` token as well. This infinite approval seems unnecessary and somewhat risky, hence should be avoided to restrict potential unknown impacts. Please note that unlimited approvals were found in other connectors as well.

Recommendation: Consider only providing a limited approval, or consider resetting the approval to 0 at the end of reinvest. rewardsAsset.forceApprove(address(swapTarget), 0);.

Kiln: Fixed in PR 333 by implementing the reviewer's recommendation.

Cantina Managed: Fix verified.

3.3 Informational

3.3.1 MetamorphoConnector.reinvest() : No explicit slippage check for potentially swapped tokens

Severity: Informational

Context: MetamorphoConnector.sol#L106

Description: The reinvest() function swaps reward tokens for the vault's underlying asset using an arbitrary call to a contract controlled by an address with the CLAIM_MANAGER_ROLE. However, the current implementation lacks an explicit check to ensure that the received amount of tokens meets a minimum threshold. This opens the door to sandwich attacks or front-running, where an attacker manipulates the price just before the swap occurs, resulting in unfavorable execution and poor returns.

```
// Approve the swap target
rewardsAsset.forceApprove(address(swapTarget), type(uint256).max);

// Swap the rewardsAsset to the underlying asset
if (swapPayload.length != 0 && swapTarget != address(0)) {
    swapTarget.functionCall(swapPayload);
}

uint256 _received = asset.balanceOf(address(this)) - _balanceBefore;
if (_received == 0) revert NothingToClaim();
```

Recommendation: Introduce an explicit minimum expected output check in the reinvest() logic. This would help protect against front-running and ensure that swaps result in meaningful returns. While the swapTarget contract might already perform such a check internally, relying on external contracts for critical validation introduces risk. Adding this safeguard within reinvest() improves resilience and clarity. It's also worth noting that similar logic appears in other connector contracts, which should be reviewed for the same issue.

Kiln: Acknowledged.

Cantina Managed: Acknowledged.

3.3.2 Reward asset cannot be Kiln or MetaMorpho vault token

Severity: Informational

Context: MetamorphoConnector.sol#L69-L71, MetamorphoConnector.sol#L89-L91

Description: The Morpho's UrdFactory allows anyone to deploy a UniversalRewardsDistributor contract to distribute any reward token to recipients.

However due to the restrictions present on rewardsAsset address in MetaMorphoConnector's claim and reinvest functions, the rewardsAsset cannot be Kiln or MetaMorpho vault token. Hence limiting the potential additional rewards strategies for Kiln vaults.

Recommendation: No change is recommended. From a security perspective it is better to prevent rewardsAsset from being Kiln or MetaMorpho vault token unless that functionality is explicitly required.

Kiln: Acknowledged. We don't see any reason to send shares of neither of the vaults through a MorphoDistributor.

Cantina Managed: Acknowledged.

3.3.3 ERC20 tokens held by MetamorphoConnector can be stolen by anyone

Severity: Informational

Context: MetamorphoConnector.sol#L69

Description: The `MetamorphoConnector.claim()` function can be called by anyone.

While unlikely, in case the `MetamorphoConnector` contract holds any ERC20 tokens (sent as rewards or transfers) then anyone can call the `claim()` function with `claimInfo.distributor` as `0` and appropriate multisend inputs and steal all the ERC20 tokens held by `MetamorphoConnector`.

Recommendation: Consider adding an `onlyDelegateCall` modifier and add that to all external functions of `MetamorphoConnector`.

Kiln: Acknowledged. We don't see any reason to send shares of neither of the vaults through a `MorphoDistributor`.

Cantina Managed: Acknowledged.

3.3.4 Natspec comment of `IConnector.totalAssets` can be improved

Severity: Informational

Context: IConnector.sol#L17-L20

Description: The natspec comment of `IConnector.totalAssets` function says that:

```
    /// @notice Get the total balance (deposit + rewards).
    function totalAssets(IERC20 asset) external view returns (uint256);
```

It does not clearly specifies whether additional rewards are also included in the result or not, hence creating confusion that unclaimed additional rewards might also be being accounted in `totalAssets` returned value.

Recommendation: Consider adding explicit comment saying that unclaimed additional rewards are not included.

Kiln: Acknowledged.

Cantina Managed: Acknowledged.