CANTINA

Level Money contracts

Security Review

Cantina Managed review by:

Chris Smith, Lead Security Researcher
Om Parikh, Security Researcher

February 24, 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

Level is the first delta-neutral synthetic dollar with first-loss protection.

From Feb 11th to Feb 13th the Cantina team conducted a review of level-money-contracts on commit hash 2e1d122c. The team identified a total of **6** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	1	1	0
Gas Optimizations	0	0	0
Informational	5	3	2
Total	**6**	**4**	**2**

3 Findings

3.1 Low Risk

3.1.1 Implications of including 3rd party Eigenlayer strategy deposits

Severity: Low Risk

Context: LevelReserveLens.sol#L263-L270

Description: The `LevelReserveLens._getReserves` function sum includes the total collateral balance staked in the Eigen Strategy.

Since there is not currently a mechanism in EigenLayer to determine which tokens were deposited by the Level strategy manager or a 3rd party, including the total collateral balance could enable a 3rd party to make LevelUSD look more collateralized than it is and potentially allow them to kick-off liquidations.

The risk is that a user controls enough USDC/USDT collateral in Eigenlayer such that that they can move the price reported by the oracle enough to trigger a liquidation. In the case where part of the "up to $1" collateralization of LevelUSD is controlled by this 3rd party, they could use this to their advantage. By monitoring the lending pools for a borrow position to be close enough to being eligible to be liquidated, this actor could remove their collateral from one of the contracts, causing the oracle to report a lower price and making the borrow position eligible to be liquidated.

If the lending protocol offers some reward for actors who trigger valid liquidations, the attacker can generate a profit this way. If not, they could still grief borrowers (especially large ones) by triggering spurious liquidations.

Recommendation: Since you are waiting on clarity on Eigenlayer slashing before you deploy collateral there and there is not a good way at this point to exclude collateral that is not yours, it may be best to simply remove the Eigenlayer calculations from the Lens until it is more clear.

I appears as though Eigenlayer has a new feature that would allow the Lens to inspect the balance of a specific staker. When this code is deployed by Eigenlayer, it would provide a means to avoid the balance check. If you decide to wait for this upgrade, it would be recommended to remove the current Eigenlayer balance check so collateralization is correct.

If that upgrade is too far out for your purposes and you decide you still need to include it, an interim solution could be designed. A small contract that sits between the Eigenlayer strategy and the LevelUSD Eigenlayer Reserve manager could be used as a passthrough accounting contract that would track `totalDeposits` and `pendingWithdraws`. The oracle could then read the `totalDeposits` instead of the Eigenlayer strategy's token balance to only include the LevelUSD collateral and exclude 3rd party deposits. `pendingWithdraws` would be a way for Morpho lenders and borrowers to monitor pending withdraws so they are not surprised by large changes in the collateralization of LevelUSD when it is close to the $1 mark.

Level Money: Fixed in PR 34 by excluding Eigenlayer balances until future update.

Cantina Managed: This makes the most sense for now due to the uncertainty of the best way to interact with Eigenlayer at this point.

3.2 Informational

3.2.1 Implications of defaulting to $1 for Oracle Pricing

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The oracle will default to $1 (`1e18`) if:

- Lens contract uses an oracle that returns 0 or reverts.
- Oracle is paused by the `PAUSER` address/es.
- Bad oracle upgrade.

This could have implications for integrators. We will focus on Morpho lending markets as you have indicated that this oracle is designed for that use case. Additionally, you asked if the inclusion of Collateral

assets that were in the process of being withdrawn from Eigen Layer would have implications for the oracle.

Defaulting to `$1` for the oracle protects borrowers. Essentially, if something goes wrong with the oracle, borrowers will not be punished by being liquidated if the oracle is "down" and would report a 0 price.

While this is correct, the other side of this statement is that defaulting to `$1` favors borrowers over lenders. If there is a problem with the oracle but it should be reporting less than `$1`, then lenders are in a losing position because loans that are unsafe are not being liquidated.

Since it is impossible to tell why the oracle is experiencing one of these problems in its return value, protecting borrowers from liquidation during a temporary issue is likely the least impactful/best decision. However, if it is prolonged and/or lending continues to operate, it does disadvantage lenders and could even hurt borrowers if they think they are taking out safe loans and then the oracle reverts to reporting a lower than `$1` price when the revert or pause is resolved.

Since Morpho's lenders are permissioned and expected to monitor the health of their loans and their market, they should be advised of this design decision such that they can notice a "*default price*" reporting situation and take appropriate actions to pause their markets if they deem that is the best course.

Recommendation: There is inherent risk of the Oracle reporting incorrect values, but these risks can be managed within the Morpho markets through monitoring the lenders should be doing. Clear documentation should be made in this contract and in any LevelUSD public docs about this design decision as there could be risks to the health of other lending markets an d especially to trading markets if they used this oracle and were served the fallback value.

Since the `PAUSER` role has special powers in this situation, it is important to design a secure system to protect this address from attackers and prevent malicious actions and to clearly document how the PAUSER could affect the price in the event the system becomes temporarily undercollateralized.

Level Money: Inline Docs added in PR 34.

Cantina Managed: Fixed.

3.2.2 Differing versions of OZ contracts are used in the repo

Severity: Informational

Context: lvlUSD4626.sol#L4

Description: As was noted in the previous Cantina audit (see 3.4.4), the repo contains an old import of Open Zeppelin's contracts.

This contract imports the latest version of OpenZeppelin's contracts whereas the previous 4626 vault `StakedlvlUSD` still uses OZ 4.9. The differences between the 4626 contracts are mainly related to the addition and use of these errors:

```
 * @dev Attempted to deposit more assets than the max amount for `receiver`.
 */
error ERC4626ExceededMaxDeposit(address receiver, uint256 assets, uint256 max);

/**
 * @dev Attempted to mint more shares than the max amount for `receiver`.
 */
error ERC4626ExceededMaxMint(address receiver, uint256 shares, uint256 max);

/**
 * @dev Attempted to withdraw more assets than the max amount for `receiver`.
 */
error ERC4626ExceededMaxWithdraw(address owner, uint256 assets, uint256 max);

/**
 * @dev Attempted to redeem more shares than the max amount for `receiver`.
 */
error ERC4626ExceededMaxRedeem(address owner, uint256 shares, uint256 max);

/**
```

However, there are some differences in the imported ERC-20 and SafeERC20 contracts.

Recommendation: Since both versions of the 4626 implementation will exist in the Level system, any error monitoring must take both the old errors and the new named errors into account.

We recommend ensuring the changes the the base ERC-20 and other imported contracts are not problematic with how the Level system plans on interacting with either 4626 vault and that appropriate updates are made to any documentation you create for integrators.

Level Money: Acknowledged, and (as discussed) will have internal systems account for this system.

Cantina Managed: Acknowledged.

3.2.3 Decimal conversion could be simpler

Severity: Informational

Context: LevelReserveLens.sol#L236-L244

Description: The only time this is used in the repo is in this contract with USDC and with USDT. In these instances, the `fromDecimals` and `toDecimals` are guaranteed to be `6` and `18` respectively.

Recommendation: Unless there is benefit to having a more open calculation, this can be simplified to:

```
return amount * (10 ** (toDecimals - fromDecimals));
```

Level Money: Fixed in PR 34.

Cantina Managed: Code simplification looks good.

3.2.4 `getEigenStake` simplification

Severity: Informational

Context: LevelReserveLens.sol#L200-L203

Description: It is not necessary to declare the `collateralToken` variable.

Recommendation:

```
function getEigenStake(IERC20Metadata collateral, address strategy) public view returns (uint256) {
    return IERC20Metadata(collateral).balanceOf(strategy);
}
```

Level Money: Fixed in PR 34.

Cantina Managed: Entire function no longer needed due to exclusion of Eigenlayer at this time.

3.2.5 `BoringVault` solidity version upgrade

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: You requested if we foresaw issues upgrading the BoringVault from Solidity `0.8.21` to `0.8.29`. While, a full audit of the BoringVault was not in-scope, it does not appear there would any issues with its code based on the changelog from `0.8.21` $\rightarrow$ `0.8.29` that would arise from the upgrade.

Recommendation: While it appears safe to change solidity versions, we would recommend a little extra testing around any changes to the low level call performed by the two `manage` functions. There are some changes to gas estimation and error handling that might touch those low level calls indirectly.

Level Money: Acknowledged.

Cantina Managed: Acknowledged.