yAudit CAP Pre-Mainnet Vault Update Review

Review Resources:

- None beyond the code repositories

Auditors:

- Spalen
- HHK

Table of Contents

Review Summary

CAP Pre-Mainnet Vault Update

The CAP Pre-Mainnet Vault is the core codebase for the CAP Pre-Mainnet campaign. It lets users deposit USDC in exchange for boosted cUSD on the MegaEth testnet using LayerZero. Pre-Mainnet Vault deposits USDC into the CAP vault and then stakes those shares for stcUSD. Once the campaign concludes, users can withdraw stcUSD.

The contracts of the CAP Pre-Mainnet Vault Repo were reviewed over one day. Two auditors performed the code review on May 26th, 2025. Only a small part of the repository was reviewed and limited to the latest commit 5ddb865c429c6c9216e60c5e5e3d70f695bc9285 of the cap-contracts repo.

Scope

The scope of the review consisted of the following contracts at the specific commit:

```
- contracts/testnetCampaign/PreMainnetVault.sol
```

After the findings were presented to the CAP team, fixes were made and included in several PRs.

This review is a code review to identify potential vulnerabilities in the code. The reviewers did not investigate security practices or operational security and assumed that privileged accounts could be trusted. The reviewers did not evaluate the security of the code relative to a standard or specification. The review may not have identified all potential attack vectors or areas of vulnerability.

yAudit and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. yAudit and the auditors do not represent nor imply to third parties that the code has been audited nor that the code is free from defects. By deploying or using the code, CAP and users of the contracts agree to use the code at their own risk.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Good	Access control mechanisms are implemented appropriately where necessary.
Mathematics	Good	No complex calculations are involved.
Complexity	Good	The codebase is simple and easy to understand.
Libraries	Good	Utilizes OpenZeppelin and LayerZero libraries for security and best practices.
Decentralization	Good	Users can withdraw funds even if the owner does not unlock transfers.
Code stability	Good	The codebase was stable during the audit.
Documentation	Good	Well-documented with NatSpec comments, with only minor omissions.
Monitoring	Good	Events are emitted within state-changing functions.
Testing and verification	Average	Includes unit tests but lacks invariant testing and fuzzing.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
  - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas savings
  - Findings that can improve the gas efficiency of the contracts.
- Informational
  - Findings including recommendations and best practices.

Critical Findings

None.

High Findings

None.

Medium Findings

None.

Low Findings

1. Low - Incorrect preview deposit

Technical Details

The function `previewDeposit()` is used to preview the amount of shares to user will receive for depositing a given amount. After the campaign has ended, deposits are not possible anymore, but the preview function will still return non-zero values.

Impact

Low. An incorrect preview function can lead to broken integrations or users burning gas because of failed transactions.

Recommendation

The function `previewDeposit()` should return zero after the campaign has ended and deposits are disabled. Explain in NatSpec that a zero value can mean that the deposits are disabled.

Developer Response

Fixed in PR#154.

Gas Saving Findings

1. Gas - Skip casting

Technical Details

Variables `asset` , `cap` and `stakedCap` are cast to address types. Instead of casting variables to address type, input parameters that are of the correct type can be used instead.

Impact

Gas savings.

Recommendation

Use address variable `_asset` instead of `asset`, _cap `instead of` cap `and` _stakedCap `instead of` stakedCap` when approving to save gas.

Developer Response

Fixed in PR150.

Informational Findings

1. Informational - Minting zero shares

Technical Details

Input parameter `_minShares` defines the minimum amount of shares the user is willing to accept. If this value is unset or set to zero, the users can lose their assets by depositing and receiving zero shares.

From the current implementation, it is not clear if the `cap` contract handles this case.

Impact

Low. The user can deposit assets but receive zero shares in return, resulting in lost assets for the user.

Recommendation

Validate that the `_minShares` is not zero.

Developer Response

Fixed in PR#155.

2. Informational - Multiple variable casting

Technical Details

Immutable `cap` variable is defined as `IVault`. Later in the contract, it is cast to `IMinter`.

Impact

Informational.

Recommendation

Use only one interface for `cap` variable. Add missing functions to interface `IVault`.

Developer Response

Fixed in PR#152.

3. Informational - Incorrect NatSpec

Technical Details

The function `deposit()` is missing a return value in NatSpec.

Impact

Informational.

Recommendation

Add missing values to NatSpec.

Developer Response

Fixed in PR#151.

4. Informational - Unclear `_minShares` param usage

Technical Details

The param `_minShares` in the `deposit()` is described in the natspec as `Minimum amount of shares to mint`.

However, this value is used when minting Cap tokens, not when minting the final shares to the user. The minted Caps are then used to mint staked Caps, and finally, the user receives shares minted equal to the amount of staked Caps received by the contract.

This means there is no check on the staked cap amount received, and the final amount of shares minted may be smaller than the `_minShares` param.

Impact

Low. The parameter usage may be unclear and may not accurately reflect the final amount of shares received.

Recommendation

Update the NatSpec documentation or apply the `_mintShares` check to the staked cap amount.

Developer Response

Fixed in PR#149.

Final remarks

The CAP Pre-Mainnet Vault provides users with a simple vault contract to participate in the CAP Pre-Mainnet campaign. The codebase is small and straightforward, with minimal functionality. Users can deposit and withdraw(after the campaign ends) staked funds. The staked Cap vault holds deposited USDC, and the user withdraws those staked vault tokens. It is recommended that users review the Staked Cap implementation or documentation to understand how to withdraw USDC from received staked funds. This review didn't cover Cap Vault and ERC4626 Staked Cap implementations. The CAP team promptly addressed the identified issues and found no vulnerabilities.