Electisec CAP Pre-Mainnet Vault Review

Review Resources:

- None beyond the code repositories

Auditors:

- Spalen
- HHK

Table of Contents

Review Summary

CAP Pre-Mainnet Vault

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. Once the campaign concludes, users can withdraw their USDC.

The contracts of the CAP Pre-Mainnet Vault Repo were reviewed over 1.5 days. Two auditors performed the code review between March 4th and March 6th, 2025. The repository was under active development during the review, but the review was limited to the latest commit 46b0dda18490f74c81204185a11bd36fe0e66647 of the cap-contracts repo.

Scope

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

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

After the findings were presented to the TODO_protocol_name 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.

Electisec and the auditors make no warranties regarding the security of the code and do not warrant that the code is free from defects. Electisec 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, TODO_protocol_name 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 - Smart contracts calling deposit can revert

Smart contracts calling `deposit()` can revert if they don't implement `receive()` function.

Technical Details

`_lzSend()` function is called with the last parameter `refundAddress` set to `msg.sender`. The send fee is set to be paid only in native token ETH. If the sender is a smart contract that didn't implement `receive()` function, deposit calls with excess fees will revert, making it hard to implement deposit calls for smart contracts.

LayerZero docs also mention that if the refund address is a smart contract, you will need to implement a fallback function for it to receive the refund.

Impact

Low. A Smart contract that calls a deposit with excess gas will revert if it doesn't implement the `receive()` function.

Recommendation

Add parameter `refundAddress` to the `deposit()` function to enable smart contracts without `receive()` to use it. Or explicitly state in NatSpec that if the `msg.sender` is a smart contract, it must implement `receive()` function.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/71.

2. Low - Missing verification for `_destReceiver`

Technical Details

Function `deposit()` is missing verification for input parameter `_destReceiver`. If `address(0)` is passed, there is no validation before sending it to LayerZero. How the destination network will handle `address(0)` is unclear, but the user won't get tokens.

Impact

Low. Users can deposit without receiving assets on the MegaETH testnet, but the collateral will be safe.

Recommendation

Validate that the input parameter `_destReceiver` is set.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/75.

3. Low - Turn off the deposit when the campaign ends

Technical Details

The `maxCampaignEnd` variable stores the campaign's end date, and users should be able to withdraw from the vault.

Since this date will probably be when the MegaETH mainnet is available, there is little interest in leaving deposits open.

It would also allow users to deposit and withdraw to mint an infinite amount of tokens on the testnet.

The `deposit()` function could be modified to revert if `block.timestamp > maxCampaignEnd` or if `transferEnabled() == true`.

Impact

Informational.

Recommendation

Consider turning off deposits when the campaign ends, or transfers are enabled.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/76.

Gas Saving Findings

1. Gas - Converting amount to shared decimals can be optimized

Technical Details

Function `_toSD()` is calculating `10 ** (decimals - sharedDecimals())` on every call. This constant value can be calculated only once to save gas on each deposit call.

Impact

Gas savings.

Recommendation

Change function `_toSD()` implementation to:

```
    function _toSD(uint256 _amountLD) internal view virtual returns (uint64 amountSD)
{
        return uint64(_amountLD / decimalConversionRate);
    }
```

And set: `decimalConversionRate = 10 ** (_localDecimals - sharedDecimals());` only once in constructor.

LayerZero official implementation also follows this pattern.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/73.

2. Gas - Parameter `_gas` can be removed

Technical Details

In function `_buildMsgAndOptions()`, parameter `_gas` can be removed because every time constant `lzReceiveGas` is passed as a parameter.

Impact

Gas savings.

Recommendation

Remove `_gas` parameter to save gas.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/74.

3. Gas - Make variables `immutable`

Technical Details

- The variables `asset` and `maxCampaignEnd` are only set in the `constructor()` and never modified.
- The function `decimals()` returns `asset.decimals()`, since it will never change it could be stored inside an `immutable` variable and returned to save an external call.

Consider making them `immutable` to save gas.

Impact

Gas.

Recommendation

Make the variables `immutable`.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/77.

Informational Findings

1. Informational - Incorrect NatSpec

Technical Details

Function `_toSD()` has incorrect NatSpec. It states: "Convert amount in shared decimals to amount in local decimals", but the function implementation is converting amount in local decimals to amount in shared decimals.

Impact

Informational.

Recommendation

Change NatSpec to: "Convert amount from local decimals to amount in shared decimals"

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/72.

2. Informational - Verify `lzReceiveGas` value

Technical Details

Constant `lzReceiveGas` is set to value `100_000`, it is used in `addExecutorLzReceiveOption()` which instructs for how much gas should be used when calling `lzReceive` on the destination endpoint. If this value is too high, users will overpay deposit calls because there is no default way to refund the difference to the sender.

Check LayerZero docs tip for more info: ”Application developers need to thoroughly profile and test gas amounts to ensure consumed gas amounts are correct and not excessive.”

Impact

Informational.

Recommendation

Verify the set value for `lzReceiveGas` is not too big so the user won't overpay each deposit.

Developer Response

Fixed in https://github.com/cap-labs-dev/cap-contracts/pull/78.

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 simple, with minimal functionalities. Users can deposit and withdraw(after the campaign ends) from the vault without any conversions or loss of assets. The CAP team promptly addressed the identified issues and found no severe vulnerabilities.