Zellic

Cega Vault

Smart Contract Patch Review

March 31, 2023

Prepared for:

Victor Zhang

Cega

Prepared by:

Aaron Esau

Zellic Inc.

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than to simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please email us at hello@zellic.io or contact us on Telegram at https://t.me/zellic_io.

1 Introduction

We were engaged to perform a review of a minor patch for the Cega Vault, which introduced a new leveraged vault.

1.1 Scope

The engagement involved a review of the following targets:

Cega Vault

Repository	https://github.com/cega-fi/cega-eth-v1/
Versions	`0257c6dec89b40a6d99245390e239acedc4614ae`
Type	Solidity
Platform	EVM-compatible
Programs	LOVProduct, LOVCalculations

Contact Information

The following consultants were engaged to conduct the assessment:

Aaron Esau, Engineer
aaron@zellic.io

1.2 Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any additional code added to the assessed project after the audit version of our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.

2 Smart Contract Patch Review

Please note that while we are reporting the findings against the scoped contracts, some of the identified issues may also apply to out-of-scope contracts as a considerable portion of the logic has been copied, for instance, between LOVProduct and FCNProduct.

2.1 Funds can be locked using minimum withdrawals

- **Target**: LOVProduct
- **Category**: Coding Mistakes
- **Likelihood**: Low
- **Severity**: Medium
- **Impact**: High

Description

There is a risk of funds being locked if the `minWithdrawalAmount` is greater than a user's deposit size. This can happen because the `minDepositAmount` can be set to be less than the `minWithdrawalAmount`. These product settings can be adjusted using the `setMinWithdrawalAmount` function or by setting the value in the LOVProduct constructor.

Impact

Users' funds can be locked in the vault if the `minWithdrawalAmount` is greater than a deposit size.

Recommendations

The `minWithdrawalAmount` should always be less than the `minDepositAmount`, even if the owner calls `setMinDepositAmount` to lower the deposit limit.

That is, the vault needs to either track:

- The lowest-ever deposit in the vault; or
- The lowest-ever `minDepositAmount` value.

Since vault shares are fungible tokens, shares could be split up into amounts that do not pass the minimum withdrawal check. So, we prefer the second solution.

Also, consider requiring that the `minWithdrawalAmount` is less than the `minDepositAmount` in the LOVProduct constructor.

2.2 Reliance on central roles

- **Target**: LOVProduct
- **Category**: Business Logic
- **Likelihood**: Low
- **Severity**: Medium
- **Impact**: High

Description

To ensure the issuance of shares after creating a deposit, the trader admin needs to process the deposit queue through the `processDepositQueue` function. However, if the trader admin fails to do so, funds could potentially be locked in the system. In this case, the protocol relies on the default admin to fix the roles and resolve the issue.

Impact

There is a potential for funds to be locked in the protocol if keys are lost or roles turn malicious.

Recommendations

Add support for cancelling deposits or withdrawals before they are processed as a failsafe.

2.3 Vaults with withdrawals can be removed

- **Target**: LOVProduct
- **Category**: Coding Mistakes
- **Likelihood**: Low
- **Severity**: Medium
- **Impact**: High

Description

Withdrawal request arrays are stored in the following mapping, keyed by vault address:

```
mapping(address => Withdrawal[]) public withdrawalQueues;
```

As a consequence, if a default admin removes a vault using the `removeVault` function, the withdrawal records are lost, and the vault shares remain locked in the LOVProduct contract.

Impact

If an admin removes a vault with withdrawal records, the funds will be irrecoverably locked.

Recommendations

Require that `withdrawalQueue[vaultAddress].length == 0` before removing a vault in the `removeVault` function.

2.4 Removing a `leverage` from the allowlist leaves records queued

- **Target**: LOVProduct
- **Category**: Coding Mistakes
- **Likelihood**: Low
- **Severity**: Medium
- **Impact**: High

Description

Removing a `leverage` value from the `leverageAllowlist` using the `updateAllowedLeverage` function leaves deposits queued.

Note that it also leaves vaults with a given `leverage` value configured with it.

Impact

An operator admin could lock deposited funds by removing a `leverage` value with deposits from the allowlist.

We consider this a bug because creating a vault or processing these records requires interaction from the trader admin, but re-allowing a `leverage` value requires the operator admin's interaction.

Recommendations

Do not allow `leverage` values to be removed from the `leverageAllowlist` if they have corresponding deposits, or otherwise, support cancelling deposits.

2.5 The `removeVault` function improperly checks index

- **Target**: LOVProduct
- **Category**: Coding Mistakes
- **Likelihood**: N/A
- **Severity**: Informational
- **Impact**: N/A

Description

The `removeVault` function does not check if the `i` argument exists in the array. Passing an invalid value reverts with `reverted with panic code 0x32 (Array accessed at an out-of-bounds or negative index)`.

```
function removeVault(uint256 i) public onlyDefaultAdmin {
    address vaultAddress = vaultAddresses[i];
    vaultAddresses[i] = vaultAddresses[vaultAddresses.length - 1];
    vaultAddresses.pop();
    delete vaults[vaultAddress];
    emit VaultRemoved(vaultAddress);
}
```

Recommendations

Require that the value is less than the length of the array.