ChainAudit

SMART CONTRACT AUDIT REPORT
for
Volmex

Prepared By : Manmeet Singh Parmar
ChainAudit
09 March 2023

Smart Contract Final Audit Report for Volmex Contract

Document Properties

Client	Volmex Protocol
Title	Smart Contract Audit Report
Version (code freeze hash)	4f1a90b36e53dd52dd7e3bac8868c9b03a90ab9e
Author	Manmeet Singh Parmar
Auditor	Manmeet Singh Parmar

Overview

The audited commit is 4f1a90b36e53dd52dd7e3bac8868c9b03a90ab9e and all Solidity contracts in the `contracts` folder were in scope. The diff specifically contains changes related to integration with LayerZero to bridge Volmex tokens between Layer1 and Layer2

Classification of issues

1. CRITICAL: Bugs leading to Ether or token theft, fund access locking, or any other loss of Ether/tokens to be transferred to any party (for example, dividends)

2. HIGH: Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement

3. MEDIUM: Bugs that can break the intended contract logic or expose it to DoS attacks.

4. COMMENTS: Other issues and recommendations

Security Assessment Methodology

Engagement Goals

Specifically, I sought to answer the following questions:

- Is it possible for the protocol to lose money?
- Are access controls well-defined?

- Is it possible to manipulate the market by using specially crafted parameters or front-running transactions?
- Is it possible for participants to steal or lose tokens?
- Can participants perform denial-of-service attacks against any of the contracts?
- Is it possible for users to lose money?

Stages of audit

- Structural Analysis
- Static Analysis
- Code Review / Manual Analysis

Detected Issues

CRITICAL

[C01] Accounting and Checks in `Layer1VolmexPositionToken` can lead to incorrect accounting of tokens when Bridging from Layer 2 to Layer 1 (can lead to overminting)

Background :

`Layer1VolmexPositionToken` is designed to function as a BaseOFT by inheriting LayerZero's `OFTCoreUpgradeable.` In the context of token bridging, tokens transferred from layer 1 will be locked in the contract and minted on the destination chain. Conversely, tokens transferred from other chains will be burned on that chain. If there is a shortage of locked layer 1 tokens, additional tokens will be minted to users, which may occur when there is an increase in layer 2 supply.

Problem :

`Layer1VolmexPositionToken` is maintaining a mapping of the number of tokens locked by an address, user's tokens are locked in this contract itself. When the tokens are locked by a user (calling the `_debitFrom` function internally) the amount is added to _lockedTokens and in the current implementation when the tokens are bridged back to layer1 (invoking the `_creditTo` function), first the amount of tokens locked by the `_toAddress` address is checked, if the `_amount <= _lockedTokens[_toAddress]` the amount is subtracted from this mapping and the amount is transferred. In case the `_amount > _lockedTokens[_toAddress]` the remaining amount of tokens are minted to the address and the key value pair is deleted from the mapping.

Breaking Scenario :

User1 locks 500 tokens in layer1, the tokens are bridged to layer2 and minted to the user's address on layer2. Now, User1 transfers these tokens to another address, lets call it User2. User2 decides to bridge the received tokens back to layer1. When `_creditTo` is invoked. It will check, `_lockedToken[user_2_address]` which will result in 0 tokens locked and thus this function will **overmint** the tokens.

Ref :
https://github.com/volmexfinance/volmex-amm/blob/0e7127f2eefee8ece1de2f836137f4f00c545959/contracts/protocol/layerZero/Layer1VolmexPositionToken.sol#L63-L78

Ideally first the locked tokens should have been released for a consistent total and circulating supply, and the remaining tokens should be minted.

Suggested fix :

The `_lockedToken` mapping is not required and the related checks can be removed. The token should work like LayerZero's BaseOFT Token
Ref :
https://github.com/LayerZero-Labs/solidity-examples/blob/main/contracts/token/oft/extension/BasedOFT.sol

HIGH

NA

MEDIUM

[M01] Incorrect implementation of Circulating Supply function in `Layer1VolmexPositionToken`

Background :

`Layer1VolmexToken` has a circulating supply function, The circulating supply refers to the coins that are accessible to the public and should not be confused with the total supply or max supply.

Problem :

Tokens transferred from layer 1 will be locked in the contract, this reduces the circulating supply as the tokens are locked. Circulating supply = Total Supply - Amount of tokens locked

Fix :

```
function circulatingSupply() public view virtual override returns (uint) {
    unchecked {
        return totalSupply() - balanceOf(address(this));
    }
}
```

COMMENTS

[Co01] Missing Events in tokens bridging

Events should be emitted in the COFT contracts when tokens are getting bridged. I.e. in `_creditTo` and `_debitFrom` methods.

Findings List

Level	Amount
CRITICAL	1
HIGH	0
MEDIUM	1
COMMENTS	1