MiloTruck

Celo

Security Review

May 7, 2025

Contents

1 Introduction

1.1 About MiloTruck

MiloTruck is an independent security researcher, primarily working as a Lead Security Researcher at Spearbit and Cantina. Previously, he was part of the team at Renascence Labs and a Lead Auditor at Trust Security.

For private audits or security consulting, please reach out to him on Twitter *@milotruck*.

1.2 Disclaimer

A smart contract security review **can never prove the complete absence of vulnerabilities**. Security reviews are a time, resource and expertise bound effort to find as many vulnerabilities as possible. However, they cannot guarantee the absolute security of the protocol in any way.

2 Risk Classification

Severity Level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2.1 Impact

- High - Funds are **directly** at risk, or a **severe** disruption of the protocol's core functionality.
- Medium - Funds are **indirectly** at risk, or **some** disruption of the protocol's functionality/availability.
- Low - Funds are **not** at risk.

2.2 Likelihood

- High - Highly likely to occur.
- Medium - Might occur under specific conditions.
- Low - Unlikely to occur.

3 Executive Summary

3.1 About Celo

Celo is an emerging Ethereum Layer-2 designed to make blockchain technology accessible to all. With its focus on scalability, low fees, and ease of use, Celo is ideal for building blockchain products that reach millions of users around the globe.

To learn more about Celo, please visit https://celo.org/.

3.2 Overview

Project Name	Celo
Project Type	L2
Language	Solidity
Repository	celo-monorepo
Commit Hash	b0335fc9cb3a80c8fc228c1d7668e8909c88a2a2

3.3 Issues Found

High	0
Medium	0
Low	0
Informational	2

4 Findings

4.1 Informational

4.1.1 Additional fork test for `SuperBridgeETHWrapper`

Context: SuperBridgeETHWrapper.sol

Description: The following Foundry test forks mainnet to verify that the L1 -> L2 deposit works as intended:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import "forge-std/Test.sol";
import {SuperBridgeETHWrapper} from "src/Wrapper.sol";

contract SuperBridgeETHWrapperTest is Test {
    // Event emitted by L1StandardBridge
    event ERC20BridgeInitiated(
        address indexed localToken,
        address indexed remoteToken,
        address indexed from,
        address to,
        uint256 amount,
        bytes extraData
    );

    // Contract addresses
    address L1_STANDARD_BRIDGE = 0x9C4955b92F34148dbcfDCD82e9c9eCe5CF2badfe;
    address WETH_L1 = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;
    address WETH_L2 = 0xD221812de1BD094f35587EE8E174B07B6167D9Af;

    // Wrapper contract
    SuperBridgeETHWrapper wrapper;

    function setUp() public {
        wrapper = new SuperBridgeETHWrapper(WETH_L1, WETH_L2, L1_STANDARD_BRIDGE);
    }

    function testDepositEthToL2() public {
        // L1StandardBridge should emit the following event
        vm.expectEmit();
        emit ERC20BridgeInitiated(WETH_L1, WETH_L2, address(wrapper), address(0xb0b), 10 ether, "");

        // Bridge 10 ETH from L1 to L2
        wrapper.wrapAndBridge{value: 10 ether}(address(0xb0b), 200_000);
    }
}
```

It can be run with:

```
forge test -vvv --match-path test/SuperBridgeETHWrapperTest.t.sol --fork-url $ETHEREUM_RPC_URL
```

4.1.2 Minor improvements to code and comments

Context: See below.

Description/Recommendation:

1. SuperBridgeETHWrapper.sol#L4-L5 - The `Initializable` and `Ownable` imports are unused and can be removed.
2. SuperBridgeETHWrapper.sol#L35 - `wrapAndBridge()` can be declared `external`.
3. SuperBridgeETHWrapper.sol#L47 - Casting `wethAddressRemote` to `address` is unnecessary as it is already stored as `address`.