VIDMA

SMART CONTRACT AUDIT

Project: Re:Water
Date: August 23rd, 2022

TABLE OF CONTENTS

SUMMARY

Vidma is pleased to present this audit report outlining our assessment of code, smart contracts, and other important audit insights and suggestions for management, developers, and users.

During the audit, we analyzed the project's on-chain EVM bridge logic. The Re:Water team has developed a cross-chain bridge that will allow users to transfer their tokens across the Ethereum and Cosmos chains. In order to claim their tokens on one of the desired blockchains, users will need to have a currency that is native to the chain on which they are claiming tokens.

For verification of cross-chain transfers involves off-chain logic run by the team. Please note, that we have not reviewed the off-chain logic related to the bridge and Cosmos version of the bridge smart contract.

During the audit process, the Vidma team found several issues. A detailed summary and the current state are displayed in the table below.

Severity of the issue	Total found	Resolved	Unresolved
Critical	0 issues	0 issues	0 issues
High	0 issues	0 issues	0 issues
Medium	2 issues	2 issues	0 issues
Low	2 issues	2 issues	0 issues
Informational	2 issues	1 issue	1 issue
Total	**6 issues**	**5 issues**	**1 issue**

After evaluating the findings in this report and the final state after fixes, the Vidma auditors can state that the contract is fully operational and secure. Under the given circumstances, we set the following risk level:

Our auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. This approach helps us adequately and sequentially evaluate the quality of the code. Code style, optimization of the contracts, the number of issues, and risk level of the issues are all taken into consideration. The Vidma team has developed a transparent scoring system presented below.

Severity of the issue	Resolved	Unresolved
Critical	1	10
High	0.8	7
Medium	0.5	5
Low	0.2	0.5
Informational	0	0.1

Please note that the points are deducted out of 100 for each and every issue on the list of findings (according to the current status of the issue). Issues marked as "not valid" are not subject to point deduction.

Based on the **overall result of the audit**, the Vidma audit team grants the following score:

In addition to manual check and static analysis, the auditing team has conducted a number of integrated autotests to ensure the given codebase has an adequate performance and security level.

The test results and the coverage can be found in the accompanying section of this audit report.

Please be aware that this audit does not certify the definitive reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by the Vidma audit team. If the code is still under development, we highly recommend running one more audit once the code is finalized.

SCOPE OF WORK

Re:Water is a unique independent mining protocol based on the ownership of Tiles — limited land areas. It is designed to provide the entire decentralized industry with a role and function. The Tiles aim to contain the predominant share of total industry liquidity.

Within the scope of this audit, two independent auditors thoroughly investigated the given codebase and analyzed the overall security and performance of the smart contract.

The audit was conducted from Aug 17, 2022 to Aug 23, 2022. The outcome is disclosed in this document.

The scope of work for the given audit consists of the following contract:

- MultiBridge.

In the current scope of work, we checked the bridge contract only, without the dependencies.

Repository:
https://gitlab.com/wwrs/bridge/eth-like-smart-contracts

Commit:
2a80f26de133d39e3d8a0a153171535ec65e0f55

WORKFLOW OF THE AUDITING PROCESS

Vidma audit team uses the most sophisticated and contemporary methods and well-developed techniques to ensure the contract is free of vulnerabilities and security risks. The overall workflow consists of the following phases:

Phase 1: The research phase

Research

After the Audit kick-off, our security team conducts research on the contract's logic and expected behavior of the audited contract.

Documentation reading

Vidma auditors do a deep dive into your tech documentation with the aim of discovering all the behavior patterns of your codebase and analyzing the potential audit and testing scenarios.

The outcome

At this point, the Vidma auditors are ready to kick off the process. We set the auditing strategies and methods and are prepared to conduct the first audit part.

Phase 2: Manual part of the audit

Manual check

During the manual phase of the audit, the Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract. The initial commit as stated in the agreement is taken into consideration.

Static analysis check

Static analysis tools are used to find any other vulnerabilities in smart contracts that were missed after a manual check.

The outcome

An interim report with the list of issues.

Phase 3: Testing part of the audit

Integration tests

Within the testing part, Vidma auditors run integration tests using the Truffle or Hardhat testing framework. The test coverage and the test results are inserted in the accompanying section of this audit report.

The outcome

Second interim report with the list of new issues found during the testing part of the audit process.

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For simplicity in reviewing the findings in this report, Vidma auditors classify the findings in accordance with the severity level of the issues. (from most critical to least critical).

All issues are marked as “Resolved” or “Unresolved”, depending on if they have been fixed by Re:Water or not. The issues with “Not Valid” status are left on the list of findings but are not eligible for the score points deduction.

The latest commit with the fixes reviewed by the auditors is indicated in the “Scope of Work” section of the report.

The Vidma team always provides a detailed description of the issues and recommendations on how to fix them.

Classification of found issues is graded according to 6 levels of severity described below:

Critical

The issue affects the contract in such a way that funds may be lost or allocated incorrectly, or the issue could result in a significant loss.

Example: Underflow/overflow, precisions, locked funds.

High

The issue significantly affects the ability of the contract to compile or operate. These are potential security or operational issues.

Example: Compilation errors, pausing/unpausing of some functionality, a random value, recursion, the logic that can use all gas from block (too many iterations in the loop), no limitations for locking period, cooldown, arithmetic errors which can cause underflow, etc.

Medium

The issue slightly impacts the contract's ability to operate by slightly hindering its intended behavior.

Example: Absence of emergency withdrawal of funds, using assert for parameter sanitization.

Low

The issue doesn't contain operational or security risks, but are more related to optimization of the codebase.

Example: Unused variables, inappropriate function visibility (public instead of external), useless importing of SCs, misuse or disuse of constant and immutable, absent indexing of parameters in events, absent events to track important state changes, absence of getters for important variables, usage of string as a key instead of a hash, etc.

Informational

Are classified as every point that increases onboarding time and code reading, as well as the issues which have no impact on the contract's ability to operate.

Example: Code style, NatSpec, typos, license, refactoring, naming convention (or unclear naming), layout order, functions order, lack of any type of documentation.

MANUAL REPORT

Unchecked tokens transfer

Medium | Resolved

Functions *upload ()* and *dispense()* ignore return value by token transfer.

Recommendation:

Use SafeERC20, or ensure that the *transfer*/*transferFrom* return value is checked.

Re-Audit:

Re:Water team considered not using the SafeERC20 from OpenZeppelin instead a check for the returned result of the token transfer was added. SafeERC20 provides:

1) Check the boolean return values of ERC20 operations and revert the transaction if they fail;
2) At the same time allowing you to support some non-standard ERC20 tokens that don't have boolean return values.

We assume that the team is aware of this fact.

Additional check is required for *upload()* function

Medium | Resolved

In function *upload()* there is no check for the imputed amount of tokens. Also, in function is missing validation for empty target parameter data. As this function provides data for BE to validate the address of the recipient on other chains it can lead to issues.

Recommendation:

Add additional checks for function parameters:

```
require(amount > 0, "Wrong amount");
require(bytes(target).length != 0, " Incorrect target");
```

Additional check is required for constructor of MultiBridge.sol

Low | Resolved

There is no verification for the zero address for the token.

Recommendation:

Consider adding a requirement for token address to not be equal to the zero address or check that token is a valid contract with help of OpenZeppelin Address library.

Additional check is required for *dispense()* function

Low | Resolved

In function *dispense()* there is no check for the imputed amount of tokens and zero address recipient.

Recommendation:

Consider adding checks for function parameters.

NatSpec annotations

Informational | Unresolved

Smart contracts are not covered by NatSpec annotations.

Recommendation:

Consider covering by NatSpec all contract`s methods.

Style Guide

Informational | Resolved

Ordering helps readers identify which functions they can call and find the constructor and fallback definitions easier.

Functions should be grouped according to their visibility and ordered:

- Constructor;
- Receive function (if exists);
- Fallback function (if exists);
- External;
- Public;
- Internal;
- Private.

Within a grouping, place the view and pure functions last.

Inside each contract, library or interface, use the following order:

- Library declarations (using statements);
- Constant variables;
- Type declarations;
- State variables;
- Events;
- Modifiers;
- Functions.

Ordering helps readers to navigate the code and find the elements more quickly.

Also, the state variable *Token* is not in *mixedCase*.

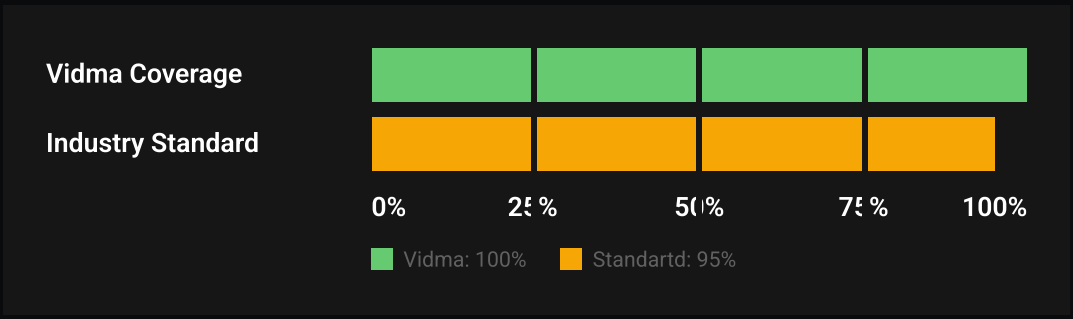
Recommendation:

Consider changing order of layout according to solidity documentations. For local and state variables use *mixedCase* names.

TEST RESULTS

To verify the security of the contract and the performance, a number of integration tests were carried out using the HardHat testing framework.

In this section, we provide tests written by Vidma auditors.



It is important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the Re:Water repository. We write totally separate tests with code coverage of a minimum of 95% to meet the industry standards.

Tests written by Vidma auditors

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	100.00	100.00	100.00
MultiBridge.sol	100.00	100.00	100.00	100.00
All Files	**100.00**	**100.00**	**100.00**	**100.00**

Test Results

```
Contract: MultiBridge
  MultiBridge Test Cases
    MultiBridge Deployment Test Cases
      ✓ shouldn't deploy with zero address token
      ✓ should deploy with correct token (111ms)
      ✓ should deploy with correct owner
      ✓ should deploy with correct initial block number
    MultiBridge Transfer Ownership Test Cases
      ✓ shouldn't transfer ownership by not the current owner
      ✓ shouldn't transfer ownership to the zero address
      ✓ should transfer ownership correctly (61ms)
    MultiBridge Upload Test Cases
      ✓ shouldn't allow to upload with zero amount
      ✓ shouldn't allow to upload with empty target
      ✓ shouldn't allow to upload if transfer returns false (139ms)
      ✓ should upload tokens correctly (52ms)
    MultiBridge Dispense Test Cases
      ✓ shouldn't dispense by not the current owner
      ✓ shouldn't allow to dispense with zero amount
      ✓ shouldn't allow to dispense with zero address recipient
      ✓ shouldn't allow to dispense if transfer returns false (87ms)
      ✓ should dispense tokens correctly (40ms)
```

```
    MultiBridge Info Bundle Test Cases
      ✓ should return correct information for chosen user

  17 passing (1s)
```

We are delighted to have a chance to work with the Re:Water team and contribute to your company's success by reviewing and certifying the security of your smart contracts.

The statements made in this document should be interpreted neither as investment or legal advice, nor should its authors be held accountable for decisions made based on this document.

Vidma is a security audit company helping crypto companies ensure their code and products operate safely and as intended, enabling founders to sleep soundly at night. We specialize in auditing DeFi protocols, layer one protocols, and marketplace solutions. Our team consists of experienced and internationally trained specialists. Our company is based in Ukraine, known for its strong engineering, cryptography, and cybersecurity culture.

Website: **vidma.io**
Email: **security@vidma.io**