Audit Report

Dora Vota Migration Contract

v1.0

September 23, 2024

Table of Contents

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security GmbH has been engaged by Matsushiba Factory Pte Ltd. to perform a security audit of Dora Vota Migration Contract.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/DoraFactory/dora-bridge-contract
Commit	`d92b30a5f46789f5dc3a35925aa0a264fb4ceb75`
Scope	The scope is restricted to the `contracts/DoraBridge.sol` contract.
Fixes verified at commit	`84117ea2a1dfecf156db1c28dd811e1a20cfa52e` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base’s intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The Dora Vota Migration Contract enables the migration of Ethereum ERC-20 DORA tokens (ethDORA) to the Dora Vota appchain. This contract is specifically designed for ethDORA holders who wish to move their tokens from Ethereum to the Dora Vota network. Only Ethereum addresses holding ethDORA can participate in this migration process.

Users interact directly with the Ethereum smart contract to initiate the migration, ensuring a non-custodial process. After the migration is complete, the ethDORA tokens are permanently removed from circulation by being sent to a null address on the Ethereum network.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged**, **Partially Resolved**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	**Low**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**Medium**	-
Test coverage	**High**	The test coverage reported by `truffle run coverage` is `97.67%`

Summary of Findings

No	Description	Severity	Status
1	Potential loss of funds due to missing address validation in `submit` function	Major	Acknowledged
2	Missing address validation in the `changeAdmin` function	Minor	Acknowledged
3	Possible duplicates in `txHashes`	Minor	Resolved
4	Missing address validation in the `constructor`	Minor	Acknowledged
5	Potential out-of-gas error in the `recordOf` function	Informational	Acknowledged
6	Functions called externally are defined as `public`	Informational	Resolved
7	Contracts should implement a two step ownership transfer	Informational	Acknowledged
8	The `DoraBridge` contract is not pausable	Informational	Acknowledged
9	The `Process` event is not indexed	Informational	Acknowledged
10	Miscellaneous comments	Informational	Partially Resolved

Detailed Findings

1. Potential loss of funds due to missing address validation in `submit` function

Severity: Major

In `contracts/DoraBridge.sol:95-113`, the `submit` function allows users to burn their tokens on Ethereum and emits an event to mint the same tokens on the Dora Vota chain at a specified `_votaAddr` address.

However, there is no validation to ensure that the `_votaAddr` is provided. If this address is not provided, the `Submit` event will be emitted without a receiver address.

This would result in a permanent loss of funds, as tokens would be burned without a reference to where they should be minted.

Recommendation

We recommend adding a validation check within the `submit` function to ensure that the `_votaAddr` is provided and the address is valid.

Status: Acknowledged

The client states that verifying the validity of a Cosmos account on the Ethereum network is complex. At present, they aim to link the Cosmos wallet verification address through their single front-end portal and ensure the validity of constructing Ethereum transactions.

2. Missing address validation in the `changeAdmin` function

Severity: Minor

In `contracts/DoraBridge.sol:49-53`, the `changeAdmin` function allows the current admin to transfer its role to another address.

However, it does not validate whether the provided `_admin` address is valid.

This omission could result in the admin being set to a nonexistent or incorrect address, potentially causing the protocol to become stuck since the admin would be unable to execute its required operations.

Recommendation

We recommend implementing a validation check within the `changeAdmin` function to ensure that the `_admin` address is not set to an invalid or incorrect address, such as `address(0)`.

Status: Acknowledged

The client states that in the current usage scenario, the `changeAdmin` method is almost always called once immediately after the contract is deployed, so there is no need to increase the security of this process for the time being.

3. Possible duplicates in `txHashes`

Severity: Minor

In `contracts/DoraBridge.sol:119-129`, the `process` function is called by the admin after the tokens are bridged on the Dora Vota chain. This function serves as a verification mechanism for the user by recording the `txHash` of the mint transaction on the Dora Vota chain.

However, the `process` function lacks validation to check if the `_txHash` provided is already present in the `_txHashes` array.

Although `txHash` values are expected to be unique for each transaction under normal circumstances, the absence of this validation allows admins to inadvertently or maliciously add duplicate entries to the `_txHashes` array.

Recommendation

We recommend implementing a validation check for duplicates in the contract and within the off-chain components. This will prevent the inclusion of duplicate transaction hashes and ensure the integrity of migration request tracking.

Status: Resolved

4. Missing address validation in the `constructor`

Severity: Minor

In `contracts/DoraBridge.sol:25-29`, the contract's constructor does not check whether the provided `_admin` and `_token` addresses are valid.

This lack of validation could lead to unintended behavior and misconfigurations, as it could allow the instantiation of the contract without having references to the token contract or the administrator.

Recommendation

We recommend implementing a validation check within the constructor to ensure the provided addresses are valid.

Status: Acknowledged

The client states that they believe, in the current scenario, it is sufficient to manually confirm the accuracy of the address once during deployment.

5. Potential out-of-gas error in the `recordOf` function

Severity: Informational

In `contracts/DoraBridge.sol:66-77`, the `recordOf` view function iterates through all the `_usersRecords` of the provided address and returns for each of them a `Record`.

However, the iteration over the `_usersRecords` array is unbounded and could result in a computationally expensive operation if the user executes the `submit` function multiple times.

Although `recordOf` is a view function, the gas limit is still enforced by the RPC nodes, potentially leading to out-of-gas errors if the array grows too large.

Recommendation

We recommend implementing pagination for the `recordOf` function.

Status: Acknowledged

6. Functions called externally are defined as `public`

Severity: Informational

The `changeAdmin`, `recordOf`, `getUnprocessedRecords`, `submit`, and `process` functions defined in `contracts/DoraBridge.sol` are called only externally, but defined as `public`, which is inefficient.

Recommendation

We recommend defining these functions as `external`.

Status: Resolved

7. Contracts should implement a two-step ownership transfer

Severity: Informational

The contracts within the scope of this audit allow the current owner to execute a one-step ownership transfer. While this is common practice, it presents a risk for the ownership of the contract to become lost if the owner transfers ownership to the incorrect address.

A two-step ownership transfer will allow the current owner to propose a new owner, and then the account that is proposed as the new owner may call a function that will allow them to claim ownership and actually execute the config update.

Recommendation

We recommend implementing a two-step ownership transfer. The flow can be as follows:

1. The current owner proposes a new checksummed owner address.
2. The new owner account claims ownership, which applies the configuration changes.

Status: Acknowledged

8. The `DoraBridge` contract is not `pausable`

Severity: Informational

Bridge contracts typically will implement `pausable` functionality to protect users from exploits that can cause a loss of funds.

In addition, due to the design of the bridge, it may be necessary to halt the execution of the `submit` function immediately after the off-chain actor responsible for handling events is suspended. This precaution is important to prevent users from burning their tokens.

Recommendation

We recommend implementing the ability to pause and unpause the bridge.

Status: Acknowledged

9. The `Process` event is not indexed

Severity: Informational

Indexing event fields significantly enhances the accessibility of these fields for off-chain tools that parse events.

However, in `contracts/DoraBridge.sol:47`, the `Process` event is missing the `indexed` keyword for the `count` and `processed` fields.

This omission could slow down event processing and make data retrieval less efficient for off-chain tools.

Recommendation

We recommend Indexing the `count` and `processed` fields in the `Process` event by adding the `indexed` keyword to these fields in the event definition.

Status: Acknowledged

10. Miscellaneous comments

Severity: Informational

Miscellaneous recommendations can be found below.

Recommendation

The following are some recommendations to improve the overall code quality and readability:

- In `contracts/DoraBridge.sol:13`, the `token` storage variable should be marked as `immutable` since it does not change following initialization in the constructor.

- In `contracts/DoraBridge.sol:17`, the `amountThreshold` storage variable should be defined as a `constant` to save gas in the `submit` function.

- The Token migration Q&A document states that *"after migration (which takes 48 hours), your ethDORA tokens will be burnt"*. However, in `contracts/DoraBridge.sol:99`, the tokens are burnt immediately via a `transferFrom` call in the `submit` function. We recommend revising the documentation to state that tokens are burnt immediately upon sending to the bridge contract.

- In `contracts/DoraBridge.sol:115-117`, all the `revert` statements are missing descriptive reason strings or custom errors. We recommend adding descriptive reason strings or custom errors to all `revert` statements.

- In `contracts/DoraBridge.sol:5-9`, the `ERC20` interface methods like `transfer` and `decimals` are not used in the code. We recommend removing that unused code in favor of readability.

- Publishing events on state changes in a smart contract is crucial for ensuring transparency and enabling off-chain monitoring, however, in `contracts/DoraBridge.sol:49`, no event is emitted during the execution of the `changeAdmin` function. We recommend implementing an `adminChanged` event.

Status: Partially Resolved