Audit Report

Wormhole Token Translator

v1.1

July 6, 2023

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

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

Introduction

Purpose of This Report

Oak Security has been engaged by Gladiolus Labs Limited to perform a security audit of the Wormhole Token Translator CosmWasm smart 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/everstake/sei-token-translator
Commit	`4ba99e7e2d84deec01d2d2671040d798ed55085b`
Scope	All code was in scope.

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 token translator contract enables the conversion of CW20 tokens to native tokens of Sei's tokenfactory module and vice-versa. Additionally, through integration with the Wormhole token bridge contract, it enables foreign assets to be minted as native tokens through Sei's `tokenfactory` module rather than CW20 tokens.

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**, 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	**High**	-
Level of documentation	**High**	The comments in the contract code sufficiently explain the logic of the contract, and the client has supplied a high-level overview of the core functionality.
Test coverage	**High**	`cargo tarpaulin` reports a test coverage for the contracts in scope of 92.27% (179/194 lines covered).

Summary of Findings

No	Description	Severity	Status
1	Truncated CW20 token transfer amount for CW20 tokens with more than 8 decimals	Minor	Acknowledged
2	Lack of configuration parameter validation	Minor	Acknowledged
3	Contracts are not compliant with the CW2 Migration specification	Minor	Acknowledged
4	Redundant check on the submessage reply	Informational	Acknowledged
5	Unused `WORMHOLE_CONTRACT` storage variable	Informational	Acknowledged
6	Missing attributes on some message handlers' responses	Informational	Acknowledged
7	Unhandled zero-amount transfer	Informational	Acknowledged

Detailed Findings

1. Truncated CW20 token transfer amount for CW20 tokens with more than 8 decimals

Severity: Minor

The Wormhole token bridge contract truncates CW20 token amounts to 8 decimals.

This implies that a small amount of tokens (the so-called “dust”) stays in the `token-translator` contract and is not transferred to the Wormhole token bridge. The user will hence receive fewer bridged CW20 tokens than initially transferred tokenfactory tokens.

Recommendation

We recommend refunding the "dust" token amount as tokenfactory tokens to the user.

Status: Acknowledged

The client noted that Wormhole CW20 contracts are always instantiated with at most 8 decimal places, so this will not affect any foreign tokens that are bridged in/out of Sei. While this could affect other kinds of CW20 contracts on Sei, it is not a big concern since the amount is so small, and most Wormhole integrators configure their UIs to truncate to 8 decimals.

2. Lack of configuration parameter validation

Severity: Minor

During contract instantiation, the storage variables `TOKEN_BRIDGE_CONTRACT` and `WORMHOLE_CONTRACT` are initialized in `src/contract.rs:33-39` with the `token_bridge_contract` and `wormhole_contract` values supplied in the `InstantiateMsg` message.

However, the provided values are missing address validation. Invalid strings would lead to a non-functioning contract.

Recommendation

We recommend validating both `token_bridge_contract` and `wormhole_contract` to ensure they are valid addresses.

Status: Acknowledged

The client noted that this is not a concern since the contract deployment and instantiation is a trusted and checked process.

3. Contracts are not compliant with the CW2 Migration specification

Severity: Minor

The `token-translator` contract does not adhere to the CW2 Migration specification standard. This may lead to unexpected problems during contract migration and code version handling.

Recommendation

We recommend following the CW2 standard in all the contracts. For reference, see the CosmWasm Migration documentation.

The migration process can be further improved by adding validation to ensure the migration is only performed if the supplied version is newer. It is recommended to follow the migrate "only if newer" pattern defined in the CosmWasm documentation.

Status: Acknowledged

The client noted that this is not a concern since the canonical token translator contract will be immutable.

4. Redundant check on the submessage reply

Severity: Informational

The `token-translator` validates whether the submessage has been successful by checking the reply's result in `src/contract.rs:94-97`. However, this check is unnecessary as the submessage is declared as `SubMsg::reply_on_success`. For `SubMsg::reply_on_success` submessages, the reply handler is only called upon the success of the submessage, making it impossible that the result is an `Err` variant.

Although not a security issue, unnecessary code can negatively impact maintainability and slightly increase gas consumption.

Recommendation

We recommend removing the redundant success validation.

Status: Acknowledged

5. Unused `WORMHOLE_CONTRACT` storage variable

Severity: Informational

During contract instantiation, the `WORMHOLE_CONTRACT` storage variable is initialized with the `wormhole_contract` value in `src/contract.rs:37-39`, which is supplied within

the `InstantiateMsg` message. However, the value of `WORMHOLE_CONTRACT` is never read from storage or used in any other way.

Recommendation

We recommend removing the unused `WORMHOLE_CONTRACT` storage variable.

Status: Acknowledged

6. Missing attributes on some message handlers' responses

Severity: Informational

The `token-translator` contract does not make use of informative attributes when returning a response in the `convert_and_transfer`, `convert_bank_to_cw20`, and `convert_cw20_to_bank` message handlers. This could negatively impact off-chain services that try to monitor the state of the protocol.

In addition, the instantiation function response includes a misleading `owner` attribute, although the contract does not save any `owner` field in the storage.

Recommendation

We recommend adding relevant information as attributes to responses so the performed action and outcome can be clearly identified by off-chain services.

Status: Acknowledged

7. Unhandled zero-amount transfer

Severity: Informational

Using the `ConvertAndTransfer` and `ConvertBankToCw20` messages, a user can send tokens to the contract. They are later validated in the context of whether more than one type of coin has been sent, but there is no validation enforcing that the transferred amount is greater than zero.

As a consequence, the functions may unnecessarily perform operations, ending up with a `panic`, because both `TokenBridgeExecuteMsg::InitiateTransfer` and `Cw20ExecuteMsg::Transfer` do not support transferring zero-amounts.

This leads to unnecessary gas consumption. Also, panics degrade the user experience since they do not provide any context why an error has occurred and how it can be resolved.

Recommendation

We recommend adding a validation step when transferring funds to ensure the amount is greater than zero and returning an error with a clear error message otherwise.

Status: Acknowledged