Audit Report

Osmosis Transmuter v3

v1.0

May 15, 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 Osmosis Grants Company to perform a security audit of the Osmosis Transmuter CosmWasm smart contract v3.

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/osmosis-labs/transmuter
Commit	`56e0e0b4a5f88758ecfcfd9413553e4184816a62`
Scope	All contracts were in scope.
Fixes verified at commit	`8fb69299e786d64886e3f4a78b2e722917df1ec0` 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 transmuter contract is designed to interact with the `cosmwasmpool` module, allowing for liquidity provision in exchange for a share token "alloyed asset" and swapping between multiple tokens with an X:Y relation and no fees.

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	Medium	-
Code readability and clarity	Medium-High	-
Level of documentation	High	The provided documentation was comprehensive.
Test coverage	Medium-High	Tarpaulin reported test coverage of 94%.

Summary of Findings

No	Description	Severity	Status
1	Corrupted assets may not be removed when liquidity reaches zero	Major	Resolved
2	Resetting change limiters creates an opportunity window for economic attacks	Major	Resolved
3	Incorrect spot price calculation	Minor	Resolved
4	Outdated libraries with known vulnerabilities in use	Minor	Resolved
5	Alloyed asset normalization factor lacks validation	Minor	Resolved
6	ChangeLimiter missing validation upon update	Minor	Resolved
7	Migration lacks validation steps	Minor	Resolved
8	Adminship can be renounced, leading to inability to maintain the protocol	Minor	Resolved
9	Outdated documentation	Informational	Resolved
10	Instantiation without a moderator is allowed	Informational	Resolved
11	Miscellaneous performance optimizations	Informational	Resolved
12	Missing current status validation during updates may mislead users	Informational	Resolved
13	Unused code	Informational	Resolved

Detailed Findings

1. Corrupted assets may not be removed when liquidity reaches zero

Severity: Major

The `transmuter` contract's specification states that corrupted assets should be removed from the contract when liquidity is zero. This is done when all the corrupted assets are redeemed at the same time in `contracts/transmuter/src/swap.rs:267-276`, but not in any other cases such as pool exits and transmutes during `unchecked_exit_pool` in `contracts/transmuter/src/transmuter_pool/exit_pool.rs:12-43` and `update_pool_assets` in `contracts/transmuter/src/transmuter_pool/transmute.rs:38-75`.

Note that the current implementation does not have any other way of removing corrupted assets from the pool.

Recommendation

We recommend checking if the liquidity of a corrupted asset has reached zero and, if so, removing it from the pool.

Status: Resolved

2. Resetting change limiters creates an opportunity window for economic attacks

Severity: Major

In `contracts/transmuter/src/contract.rs:225-228`, the states of all change limiters are reset. This is done when new assets are added because weights are no longer correct and should be recalculated.

However, change limiters implement protection from rapid market changes and pool imbalances. Removing them, even temporarily, creates an opportunity window for economic attacks. A malicious party capable of joining the pool with a large position can simply listen to the contract's on-chain events and deploy the attack as soon as new assets are added.

Recommendation

We recommend implementing a change limiters upgrade mechanism that would create a new division with a recalculated safe average for each existing asset.

Status: Resolved

3. Incorrect spot price calculation

Severity: Minor

The `spot_price` function returns a fixed value of `Decimal:one()` in `contracts/transmuter/src/contract.rs:673`. This is incorrect as the assets in the pool can have different normalization factors, resulting in prices other than 1.

We classify this issue as minor because the affected function does not impact the logic of the contract, as it is exclusively used in a query. However, note that it could have a severe impact on external contracts relying on this information.

This issue was independently reported by the Osmosis team during the audit.

Recommendation

We recommend taking the normalization factor of the assets into account to calculate the spot price.

Status: Resolved

4. Outdated libraries with known vulnerabilities in use

Severity: Minor

The `transmuter` contract makes use of outdated libraries that contain known vulnerabilities. Although we did not find these vulnerabilities to be exploitable in the current codebase, updates to the codebase may lead to security risks in the future. The following libraries are affected:

- `sylvia 0.7.0`
  - Affected by RUSTSEC-2024-0012
- `cosmwasm-std 1.3.1`
  - Affected by CWA-2024-002

Recommendation

We recommend updating the affected crates to their latest version.

Status: Resolved

5. Alloyed asset normalization factor lacks validation

Severity: Minor

The `set_normalizaton_factor` function in `contracts/transmuter/src/alloyed_asset.rs:67-73` saves the normalization

factor for the alloyed asset. However, this value does not undergo any validation as opposed to the normalization factor of the pool assets.

If, by mistake, a value of zero is assigned to the normalization factor of the alloyed asset the features that operate with the alloyed asset will error, making them unusable.

Recommendation

We recommend ensuring the normalization factor is greater than zero before saving it.

Status: Resolved

6. `ChangeLimiter` missing validation upon update

Severity: Minor

The `set_change_limiter_boundary_offset` function in `contracts/transmuter/src/limiter/limiters.rs:416-447` does not validate the `boundary_offset` value through `ensure_boundary_offset_constrain` before saving it.

Recommendation

We recommend ensuring the value is larger than zero before saving it.

Status: Resolved

7. Migration lacks validation steps

Severity: Minor

The `add_normalization_factor_to_pool_assets` function in `contracts/transmuter/src/migrations/v3_0_0.rs:67-73` and the `set_alloyed_asset_normalization_factor` function in lines `113-121` save the normalization factor for the alloyed asset and the pool assets upon migration. However, these values do not undergo any validation.

If a value of zero is mistakenly assigned to the assets' normalization factor, most of the main features would become unusable.

Recommendation

We recommend ensuring that the normalization factors are greater than zero before saving them.

Status: Resolved

8. Adminship can be renounced, leading to inability to maintain the protocol

Severity: Minor

The `transmuter` contract implements a `renounce_adminship` entry point in `contracts/transmuter/src/contract.rs:776-785`, which leaves the contract without any address holding admin privileges.

Given that several functions needed for the correct long-term operation of the contract are restricted to the admin address only, it will not be possible to maintain the protocol if the adminship is dropped at any point.

We classify this issue as minor since it can only be caused by the admin, who is a trusted entity.

Recommendation

We recommend removing the `renounce_adminship` entry point.

Status: Resolved

9. Outdated documentation

Severity: Informational

The `transmuter` contract restricts access to its `set_active_status` function to moderator users only, following security best practices. However, the access control table in the repository's `README.md` file incorrectly states that this function should be restricted to admin users only.

Recommendation

We recommend updating the documentation to reflect the current implementation.

Status: Resolved

10. Instantiation without a moderator is allowed

Severity: Informational

In `contracts/transmuter/src/contract.rs:81`, the transmuter contract instantiation is parameterized by an optional address for the moderator.

The moderator role is responsible for pausing the contract if any security issue arises and marking assets as corrupt. There are other ways to pause the contract, either by governance or by the admin, but they are less straightforward and potentially slower to react. Specifically,

governance would need to pass the proposal, which takes time. On the other hand, the admin would need to grant itself moderator privileges first, and this is one more step that can count in critical scenarios.

As a consequence, the contract would ideally not allow a configuration with the moderator set to `None`.

Recommendation

We recommend making the moderator field non-optional. During instantiation, either the deployer should explicitly specify the moderator address or a fallback could be implemented to set the moderator address to be equal to the admin address.

Status: Resolved

11. Miscellaneous performance optimizations

Severity: Informational

The codebase contains minor inefficiencies that affect performance and hence gas costs:

- In `contracts/transmuter/src/transmuter_pool/add_new_assets.rs:9-10`, the assets vector is checked for duplicates first, and then its length is checked for being below the limit. However, duplicate search is a more computationally intensive operation and should be performed after the more efficient length check.
- In `contracts/transmuter/src/transmuter_pool/corrupted_assets.rs:91`, the `pool_assets` vector is iterated to discover any corrupted assets. If such has not been found, the control flow prematurely returns. Then, in lines `97-101`, the same vector is iterated again to filter corrupted assets and construct a `HashMap`. Redundant iteration could be avoided by performing the check between the filtering and creating the `HashMap`.
- In `contracts/transmuter/src/swap.rs:431`, the code responsible for checking if `token_out.denom` differs from `token_in.denom` is not necessary as two lines later, during the execution of the `swap_variant` function, the same check is performed as well.
- In `contracts/transmuter/src/swap.rs:498`, the code responsible for checking if `token_out.denom` differs from `token_in.denom` is not necessary, as two lines later, during the execution of the `swap_variant` function, the same check is performed as well.

Recommendation

We recommend removing the inefficiencies described above.

Status: Resolved

12. Missing current status validation during updates may mislead users

Severity: Informational

In `contracts/transmuter/src/contract.rs:456`, the `set_active_status` function, as well as `SetActive` sudo message in `contracts/transmuter/src/sudo.rs:48`, no validation ensures that the current and updated statuses are not equal.

Consequently, even if these values are equal, information will be emitted stating that the status has been changed. This might be misleading and may even have a security impact on contract management during an incident.

Recommendation

We recommend ensuring that the current and updated values are not equal.

Status: Resolved

13. Unused code

Severity: Informational

The codebase contains unused code, which may indicate that the logic has changed or that assumptions do not match the current implementation. Instances of unused code have been found as follows:

- `SingleTokenExpected` error in `contracts/transmuter/src/error.rs:24`
- `InvalidTokenInAmount` error in `contracts/transmuter/src/error.rs:80`
- `UnexpectedDenom` error in `contracts/transmuter/src/error.rs:100`

Recommendation

We recommend using or removing unused code from the codebase.

Status: Resolved