Cryptonics

Audit Report

TerraSwap Liquidity Bootstrap Pool

August 5, 2021

Table of Contents

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 has been performed by

Philip Stanislaus, **Iraklis Leontiadis** and **Stefan Beyer**

Cryptonics Consulting S.L.
Ramiro de Maeztu 7
46022 Valencia
SPAIN

https://cryptonics.consulting/
info@cryptonics.consulting

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged by Terra Capital to perform a security audit of the liquidity bootstrap pool for version 2 of the TerraSwap protocol (https://terraswap.io/).

The objectives of the audit are as follows:

1. Determine the correct functioning of the system, 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 GitHub repository:

https://github.com/attic-terra/terraswap-lbp/tree/master

Commit hash: `378de03102eb82327d45d70d16ebce151459a5df`

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 submitted contracts implement a liquidity bootstrap pool based on the Balancer model.

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**. Informational notes do not have a status, since we consider them optional recommendations.

Note, that audits are an important step to improve 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. To help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentatio**n, and **test coverage**. We include a table with these criteria for each module, in the corresponding findings section.

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

Summary of Findings

No	Description	Severity	Status
1	Erroneous spot price calculation leads to wrong reported spreads and may cause swaps to fail	Major	Resolved
2	Contract queries rely on storage implementation	Minor	Resolved
3	Use of system time could cause panics	Minor	Resolved
4	Outdated crate raw-cpuid v7.0.4	Minor	Acknowledged
5	Swap operations simulation and execution might provide different results	Minor	Resolved
6	Overflow checks not set for profile release in contracts/terraswap_token/Cargo.toml and packages/terraswap/Cargo.toml	Informational	Resolved
7	Duplicate read of pair info	Informational	Resolved
8	No check for providing zero liquidity	Informational	Resolved
9	Avoid usage of foo/bar naming in production code	Informational	Resolved
10	Calc out given in could return non zero value when in is zero	Informational	Resolved
11	Swapping is no longer possible after pair's end time passed	Informational	Acknowledged

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Low-Medium	-
Code readability and clarity	High	-
Level of Documentation	Medium	Documentation is outdated and diverges from the implementation. Examples for that are the following files: - `packages/terraswap/README.md`, `contracts/terraswap_factory/README.md`, `contracts/terraswap_router/README.md` - `contracts/terraswap_pair/README.md` describes a commision of 0.3%, while the actual implemented commision is 0.15% - `contracts/terraswap_pair/src/contract.rs:443` describes sending of commission to collector, which does not happen in the code
Test Coverage	Medium	-

Detailed Findings

1. Erroneous spot price calculation leads to wrong reported spreads and may cause swaps to fail

Severity: Major

The spot price calculation in the `get_ask_by_spot_price` function in `contracts/terraswap_pair/src/contract.rs:626` uses a wrong formula, which leads to incorrect spreads. That leads to wrong reported spreads in `contracts/terraswap_pair/src/contract.rs:457` and line `556`, and might cause swaps to fail due to a failing max spread assertion in line `426`. The implemented formula calculates a spot price as follows:

```
ask_amount = (ask_pool * ask_weight) / (offer_pool * offer_weight)
* offer_amount
```

Recommendation

We recommend updating the formula to:

```
ask_amount = (ask_pool / ask_weight) / (offer_pool / offer_weight)
* offer_amount
```

Status: Resolved

2. Contract queries rely on storage implementation

Severity: Minor

In multiple places in the codebase, raw queries are used to read storage from other contracts. That ties the calling contract to the implementation of the called contract, which can lead to major issues if the implementation changes, potentially even a loss of funds. Occurrences of this pattern in the codebase are:

- `packages/terraswap/src/querier.rs:46`
- `packages/terraswap/src/querier.rs:63`
- `contracts/terraswap_factory/src/querier.rs:13`

Recommendation

We recommend using query interfaces as opposed to raw storage queries.

Status: Resolved

3. Use of system time could cause panics

Severity: Minor

In the `simulate_swap_operations` function in `contracts/terraswap_router/src/contract.rs:261`, `SystemTime::now()` is used as the current time. This function is not available during smart contract execution. If another contract is using that query, the calling contract will always receive a panic. (Actually, an out of gas error will show up, see https://github.com/CosmWasm/cosmwasm/issues/530 for details.)

Recommendation

We recommend using a user provided time instead.

Status: Resolved

4. Outdated crate raw-cpuid v7.0.4

Severity: Minor

Vulnerable to memory corruption: https://rustsec.org/advisories/RUSTSEC-2021-0013

Recommendation

Update to the latest version

Status: Acknowledged

raw-cpuid v7.0.4 is a dependency from cosmwasm-vm v0.10.1 and cannot be fixed without an upgrade of the underlying CosmWasm SDK.

5. Swap operations simulation and execution might provide different results

Severity: Minor

During swap operations execution, the `assert_operations` function is called in `contracts/terraswap_router/src/contract.rs:103`. This assertion is not performed during swap operations simulation in `contracts/terraswap_router/src/contract.rs:202` though. Consequently, some `operations` inputs might succeed during simulation, but fail during execution, for example if operations have multiple outputs during simulation.

Additionally, the simulation function assumes that the output denom of the previous operation equals the input denom of the next operation. There is no check for that assumption in the code though. If that assumption was not fulfilled, an actual execution would lead to a zero

token amount as the result of the operation, while the simulation would give an actual number.

Recommendation

We recommend adding the `assert_operations` function to the swap operations simulation function.

We also recommend adding a check for the input denom being equal to the previous output denom to the operations simulation function.

Status: Resolved

6. Overflow checks not set for profile release in `contracts/terraswap_token/Cargo.toml` and `packages/terraswap/Cargo.toml`

Severity: Informational

While set in all other packages, `contracts/terraswap_token/Cargo.toml` and `packages/terraswap/Cargo.toml` do not enable `overflow-checks` for the release profile.

Recommendation

Even though this check is implicitly applied to all packages from the workspace `cargo.toml`, we recommend also explicitly enabling overflow checks in every individual package. That helps prevent unintended consequences when the codebase is refactored in the future.

Status: Resolved

7. Duplicate read of pair info

Severity: Informational

During withdrawal of liquidity, pair info is read twice in `contracts/terraswap_pair/src/contract.rs:179` and `314`. This increases computation consumption.

Recommendation

We recommend reusing the value from the first pair info read.

Status: Resolved

8. No check for providing zero liquidity

Severity: Informational

Providing liquidity with an asset amount of `0` will lead to the full execution of the `try_provide_liquidity` function in `contracts/terraswap_pair/src/contract.rs:220`. That leads to unnecessary resource consumption and events of zero transfers.

Recommendation

We recommend adding a check for zero and return an error.

Status: Resolved

9. Avoid usage of foo/bar naming in production code

Severity: Informational

In `contracts/terraswap_pair/src/math.rs:44` and `contracts/terraswap_pair/src/math.rs:65`, variable names foo and bar are used.

Recommendation

We recommend using more descriptive variable names to improve readability and documentation of the code.

Status: Resolved

10. Calc out given in could return non zero value when in is zero

Severity: Informational

In `contracts/terraswap_pair/src/math.rs:30`, a zero `amount_in` value could result in a positive `amount_out` value due to rounding issues.

Recommendation

We recommend checking whether `amount_in` equals zero and return a zero `amount_out`.

Status: Resolved

11. Swapping is no longer possible after pair's end time passed

Severity: Informational

In contracts/terraswap_pair/src/contract.rs:759, an error is returned if the current block time is after the pair's end time. That implies that swaps cannot be executed anymore after the end time.

Status: Acknowledged

This behaviour is intended.