TRUFY

Smart Contract Audit Report
for
Orchai Protocol

October, 2022

Contents

1 Introduction

Trufy has been engaged by what to perform a security audit of the Orchai Protocol smart contracts. This current report covers the implementation of the CosmWasm smart contract of the Orchai Protocol. The purpose of this audit is to achieve the followings:

- Ensure that smart contract functions work as intended.
- Identify possible vulnerabilities, which could be exploited by an attacker.
- Identify smart contract bugs, which might lead to unexpected behavior.
- Make recommendations to improve code safety and readability.

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.

1.1 About Orchai Protocol

Orchai is a lending and borrowing platform that implements AI functions via Oraichain's distinctive AIOracle mechanism. This project will utilize AI models to optimize protocol parameters in a dynamic market environment. The AI models are voted through DAO proposals and connected to smart contracts via AI Oracle. With that, Orchai will enhance the transparency and decentralization of the platform while ensuring a friendly user experience. The Orchai protocol consists of Orchai Liquid Staking and Orchai Money Market.

The Orchai Liquid Staking protocol is a liquid staking solution that tokenizes staked ORAI to grant users the ability to utilize the yield-bearing asset within Decentralized Finance applications.

The Orchai Money Market protocol defines a money market between lenders, providing liquidity to the protocol to earn yield, while borrowers look to borrow on stakeable assets. To be able to borrow, the borrower locks up staked assets as collateral in an over-collateralized manner.

1.2 Project Summary

1.2.1 Codebase

This audit has performed on the code submitted in the following GitHub repositories:

- Orchai-sAsset-contracts:
  - https://github.com/OrchaiLendingPool/Orchai-sAsset-contracts
  - Commit no: a990ab8bb222e8d2b2ac487755f9b7ddadfedfd7
- Orchai-money-market:
  - https://github.com/OrchaiLendingPool/Orchai-money-market
  - Commit no: fc9c625f5c758f9dfd12e739d0934c625bb5bcde

1.2.2 Methodology

Trufy performed a combination of manual review of the code and automated security testing.

The following phases and associated tools were used throughout the term of the audit:

- Research into architecture, purpose, and use of the platform by reading the available documentation.
- Manual code read and walk-through.
- Checking the test coverage by tool Cargo tarpaulin.
- Scanning of Rust files for vulnerabilities by tool Cargo audit.

1.3 Vulnerability Summary

Severity	# of Findings
Critical	0
Medium	1
Low	0
Informational	4

2 Findings

Orchai-sAsset-contracts

ID	Title	Severity	Status
ID-01	Losing coin while send other than `underlying_coin_denom`	Informational	Resolved
ID-02	Performing redundant calculations	Informational	Resolved
ID-03	Unused parameter	Informational	Resolved

Orchai-money-market

ID	Title	Severity	Status
ID-04	Overflow checks not set for profile release in all packages	Informational	Resolved
ID-05	Handler of message `RegisterContracts` does not check permission	Medium	Resolved

3 Detailed Results

3.1 ID-01: Losing coin while send other than `underlying_coin_denom`

Severity: Informational

In several places in the codebase, send_funds are iterated to find out if there is `underlying_coin_denom`, but any other coins sent are not prevented or returned to the sender. That make those coins frozen in the contract.

Those places are: contracts/orchai_stake_hub/bond.rs:47 contracts/orchai_stake_hub/contract.rs:43

Recommend

We recommend either returning unused coins or reverting the transaction if unexpected coins are sent.

3.2 ID-02: Performing redundant calculations

Severity: Informational

Function `execute_bond()` at contracts/orchai_stake_hub/bond.rs:17:

- Line 44, function slashing was called, this make STATE.total_bond_amount was updated to sum of `delegation.amount.amount`.
- But at line 81, value actual_bond_amount is recalculated by sum of `delegation.amount.amount`, this lead to redundant calculation.

Recommend

We recommend using value of `STATE.total_bond_amount` for variable `actual_bond_amount`.

3.3 ID-03: Unused parameter

Severity: Informational

At contracts/orchai_stake_hub/unbond.rs:353: in declaration of function `pick_validator()`, parameter `block_height` is not used.

3.4 ID-04: Overflow checks not set for profile release in all packages

Severity: Informational

While set in all other packages, but packages/oraiswap/Cargo.toml does not enable `overflow-checks` for the release profile.

Recommend

While 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 when the project is refactored to prevent unintended consequences.

3.5 ID-05: Handler of message `RegisterContracts` doese not check permission

Severity: Medium

Message `RegisterContracts` at contracts/market/contract.rs:132:

- Handler is function `register_contracts()` at contracts/market/contract.rs:259. This handler will set value variables `overseer_contract`, `interest_model`, `collector_contract`, `distributor_contract`. This handler should be restricted to admin or owner of the contract market.
- But in this handler, there is no permission check. So any one send message `RegisterContracts` to contract market.

Recommend

We recommend adding permission check to the handler `register_contract`.

4 Appendix

4.1 Severity Definitions

Critical

This level vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.

Medium

This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical-risk severity.

Low

This level vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.

Informational

This level vulnerabilities can be ignored. They are code style violations and informational statements in the code. They may not affect the smart contract execution.

4.2 Scanning vulnerabilities by `Cargo audit`

We use Cargo audit (https://crates.io/crates/cargo-audit) to scan codebase. Warnings are raised by Cargo audit:

- Orchai-money-market:
  - RUSTSEC-2020-0025 (https://rustsec.org/advisories/RUSTSEC-2020-0025)

```
Crate:     bigint
Version:   4.4.3
Warning:   unmaintained
Title:     bigint is unmaintained, use uint instead
Date:      2020-05-07
ID:        RUSTSEC-2020-0025
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0025
Dependency tree:
bigint 4.4.3
└── oraiswap 2.4.1
    ├── overseer 0.1.0
    ├── orchai-token 0.3.0
    │   └── money-market 0.1.0
    ├── moneymarket-oracle-2 0.1.0
    ├── moneymarket-liquidation-queue 0.1.0
    ├── moneymarket 0.3.1
    │   ├── overseer 0.1.0
    │   ├── moneymarket-oracle-2 0.1.0
    │   ├── moneymarket-liquidation-queue 0.1.0
    │   ├── money-market 0.1.0
    │   ├── custody_sc_orai 0.1.0
    │   ├── custody_borai_reward 1.1.0
    │   └── custody-borai 0.1.0
    ├── money-market 0.1.0
    └── custody-borai 0.1.0
└── moneymarket-liquidation-queue 0.1.0
```