Cryptonics

Anchor Token and Distributions Smart Contracts - Audit Report

April 6, 2021

This audit has been performed by

Philip Stanislaus and **Stefan Beyer**

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

https://cryptonics.consulting/

Table of Contents

Disclaimer

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

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

THIS AUDIT REPORT IS NOT A SECURITY WARRANTY, INVESTMENT ADVICE, OR AN ENDORSEMENT OF THE CLIENT OR ITS PRODUCTS. THIS AUDIT DOES NOT PROVIDE A SECURITY OR CORRECTNESS GUARANTEE OF THE AUDITED SOFTWARE.

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged by Terraform Labs to perform a security audit of the Anchor token and distribution smart contracts.

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 code submitted in the following GitHub repository:

https://github.com/Anchor-Protocol/anchor-token-contracts

Commit no: `ef29e3e635adbdb4c73925065b02a2c50f7b0cc2`

In addition the following pull request to the Anchor money market repository has been audited:

https://github.com/Anchor-Protocol/money-market-contracts/pull/16

Commit no: `60a71be800a673a091aa9deaa5baa2188cae2c9f`

Methodology

The audit has been performed by a mixed team of smart contract and full-stack auditors.

The following steps were performed:

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
    d. Permissioning issues
    e. Logic errors
4. Report preparation

The results were then discussed between the auditors in a consensus meeting and integrated into this joint report.

Functionality Overview

The submitted code implements the smart contracts for the Anchor protocol's token implementation and distribution 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. 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 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

The Anchor smart contracts were found to contain 0 critical issues, 1 major issue, 6 minor issues and 6 informational notes:

No	Description	Severity	Status
CosmWasm anchor-token-contracts Smart Contracts			
1	Ending of polls can be blocked	Major	Resolved
2	Poll indexer and poll voter storage conflicts will cause overwritten data	Minor	Resolved
3	Overwriting an airdrop merkle root will cause users being unable to claim updated amounts	Minor	Resolved
4	Storing an airdrop merkle root for a future stage may cause users being unable to claim updated amounts	Minor	Resolved
5	Withdrawing voting tokens will panic if amount is not set	Minor	Resolved
6	Polls cannot be ended if no votes exists in the system	Minor	Resolved
7	Allowing updates to gov config values for ongoing polls can disturb users	Informational	Acknowledged
8	Invalid merkle roots can cause panics during airdrop claims	Informational	Resolved
9	After gov contract initialization, anyone can set the anchor token contract	Informational	Acknowledged
10	Overflow checks not set for profile release in `packages/anchor_token/Cargo.toml`	Informational	Resolved

CosmWasm money-market-contracts Smart Contract PR			
11	Rewards cannot be claimed after repaying a loan	Minor	Resolved
12	Querying borrower may not consider the block height parameter for interest and rewards	Informational	Acknowledged
13	Overflow checks not set for profile release in `packages/moneymarket/Cargo.toml`	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium-high	-
Level of Documentation	Medium-high	-
Test Coverage	Medium-high	-

Detailed Findings

CosmWasm anchor-token-contracts Smart Contracts

1. Ending of polls can be blocked

Severity: Major

The `poll_all_voters` call in `contracts/gov/src/contract.rs:547` is unbounded, as is the `voter_store.remove` call in line `555`. A poll could be blocked by an attacker by adding lots of votes such that ending the poll runs out of gas, implying that a poll can never be ended.

Recommendation

We recommend changing the behaviour to let users unlock their own tokens and remove themselves from the voter store after a poll ends instead of automatically unlocking all tokens and removing all voters, such that there are no loops needed here.

Status: Resolved

2. Poll indexer and poll voter storage conflicts will cause overwritten data

Severity: Minor

`poll_indexer_*` and `poll_voter_*` store in `contracts/gov/src/state.rs` do both use the same storage prefix `PREFIX_POLL_VOTER`. That will cause conflicts and overwritten data.

Recommendation

We recommend using distinct storage prefixes for `poll_indexer_*` and `poll_voter_*` stores.

Status: Resolved

3. Overwriting an airdrop merkle root will cause users being unable to claim updated amounts

Severity: Minor

The `store_merkle_root` function in `contracts/airdrop/src/contract.rs:73` does not prevent overwriting merkle roots for existing stages. If a user has already claimed tokens and a new merkle root from a tree with updated amounts is stored, the user will not be able to claim those up with updated amounts.

Recommendation

We recommend either changing the claim behaviour to store the already claimed amount per user and stage and only transferring the unclaimed amount or disallowing overwrites of merkle roots for existing stages.

UPDATE

This issue has been resolved by refactoring and removing trusted code, reducing possible admin control.

Status: Resolved

4. Storing an airdrop merkle root for a future stage may cause users being unable to claim updated amounts

Severity: Minor

The `store_merkle_root` function in `contracts/airdrop/src/contract.rs:73` does not prevent storing a merkle root for a future stage and does also not update the `latest_stage`. That allows the owner to set a merkle root for a stage in the future, which could be used in a claim by a user. That merkle root may be overwritten at some point by the register_merkle_root function. A user that has already claimed tokens for the stage will not be able to claim the amount from the updated merkle tree.

Recommendation

We recommend rejecting the update of merkle roots that are for a stage later than `latest_stage` or updating `latest_stage` to the newly stored stage.

UPDATE

This issue has been resolved by refactoring and removing trusted code, reducing possible admin control.

Status: Resolved

5. Withdrawing voting tokens will panic if amount is not set

Severity: Minor

Unwrapping the `amount` in the `withdraw_voting_tokens` function in `contracts/gov/src/contract.rs:303` will panic if the `amount` is `None`.

Recommendation

We recommend to calculate the amount by using `withdraw_share * total_balance` instead.

Status: Resolved

6. Polls cannot be ended if no votes exists in the system

Severity: Minor

The `end_poll` function in `contracts/gov/src/contract.rs:479` returns an error if `total_share` equals `0`. That implies that if no votes exist in the gov contract at all, a poll that passed `end_height` cannot be ended. At the same time, a vote cannot be cast for a poll that has passed `end_height`.

The only workaround is to create a new poll, vote on that poll, and then end the previous poll.

Recommendation

We recommend removing the condition on the `total_share` value.

Status: Resolved

7. Allowing updates to gov config values for ongoing polls can disturb users

Severity: Informational

In multiple places in the gov contract, current config values are used. When those config vales is updated through `update_config` in `contracts/gov/src/contract.rs`, ongoing polls will be affected. That behaviour can be unexpected for users and disturb them. In particular:

- Using the current quorum from the config in `contracts/gov/src/contract.rs:506` might have unintended consequences. For rational voters, it does not make sense to participate in a vote if quorum and threshold are already reached. Their vote would cost them fees, without affecting the outcome. If the quorum is changed after the voting period ended though (or even shortly before it), they will (might) not be able to vote.

- Likewise, using the current threshold from the config in `contracts/gov/src/contract.rs:511` might have unintended consequences. For rational voters, it does not make sense to participate in a vote if the threshold is already reached. Their vote would cost them fees, without affecting the outcome. If the threshold is changed after the voting period ended though (or even shortly before it), they will (might) not be able to vote.
- Using the current expiration period from the config in `contracts/gov/src/contract.rs:506` can have unintended consequences. Users might start a poll that's time critical and should be executed before a certain block height. A change in the global expiration period makes any planning for such a time critical poll useless, since the expiration period could be changed through governance at any time. A coordinated change that includes the interests of all ongoing polls might get impossible if there are many ongoing polls.
- Likewise, using the current timelock period from the config in `contracts/gov/src/contract.rs:585` can have unintended consequences. Users might start a poll that's time critical and should be executed before a certain block height. A change in the global timelock period makes any planning for such a time critical poll useless, since the timelock period could be changed through governance at any time. A coordinated change that includes the interests of all ongoing polls might get impossible if there are many ongoing polls.

Recommendation

We recommend storing the `quorum`, `threshold`, `expiration_period` and `timelock_period` at poll creation and using those stored values instead of the current ones.

Status: Acknowledged

This behaviour is intentional since it allows speeding up pending polls in critical situations.

8. Invalid merkle roots can cause panics during airdrop claims

Severity: Informational

Since the merkle root is not validated in the `register_merkle_root` and `update_merkle_root` functions in `contracts/airdrop/src/contract.rs`, decoding in the `claim` function in `contracts/airdrop/src/contract.rs:175` could panic.

Recommendation

We recommend validating the decodability of the merkle root before storing it in `register_merkle_root` and `update_merkle_root`.

Status: Resolved

9. After gov contract initialization, anyone can set the anchor token contract

Severity: Informational

During the `init` function of the market contract in `contracts/gov/src/contract.rs:37`, the `anchor_token` variable is assigned to `CanonicalAddr::default()`. After that initialization, anyone can send the `RegisterContracts` message, since there is no permission check in the `register_contracts` handler.

Recommendation

We recommend adding permissioning to the `register_contracts` handler for the `RegisterContracts` message.

Status: Acknowledged

Contract deployment is done in a script and can be verified after deployment.

10. Overflow checks not set for profile release in `packages/anchor_token/Cargo.toml`

Severity: Informational

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

Recommendation

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.

Status: Resolved

CosmWasm money-market-contracts Smart Contract PR

11. Rewards cannot be claimed after repaying a loan

Severity: Minor

The check in `contracts/market/src/borrow.rs:202` prevents reward claims if a loan is repaid already.

Recommendation

We recommend allowing reward claims even after repaying a loan.

Status: Resolved

12. Querying borrower may not consider the block height parameter for interest and rewards

Severity: Informational

With the changes in this PR, interest is and rewards are only recomputed if the passed block height is newer than the one from the last computation within a call that stores the state. That change implies that the passed block height in `query_borrower_info` in `contracts/overseer/src/querier.rs:27` will only be considered if it is greater than the stored one.

Recommendation

While not a security concern, this behaviour may confuse users. We recommend removing the ability to query borrower info by a specific block height.

Status: Acknowledged

13. Overflow checks not set for profile release in `packages/moneymarket/Cargo.toml`

Severity: Informational

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

Recommendation

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.

Status: Resolved