Cryptonics

Audit Report

Terra Blockchain – Columbus-5

July 23, 2021

Table of Contents

Disclaimer

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/
info@cryptonics.consulting

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged by Terraform Labs to perform a security audit of the Terra blockchain implementation (Columbus-5 release) (https://www.terra.money/).

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/terra-money/core/tree/release/v0.5.x

Commit hash: `5ac5439e1d3ba17da0216181af9c09ca1155e63f`

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 code base implements the Cosmos SDK-based node software for the Terra blockchain, in its Columbus-5 iteration.

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 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 documentation**, and **test coverage**. We include a table with these criteria 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	Human/canon address conversion allows attack that may halt block production	Critical	Resolved
2	Code migration allows code creator to change behavior without contract admin's consent	Critical	Acknowledged
3	Slash windows may be skipped, leading to higher miss counters and higher slashing	Major	Resolved
4	Users might overpay for swap and swap send messages from Terra to Luna	Major	Resolved
5	Swap simulation may return different amount from actual swap	Minor	Acknowledged
6	Voters that have not voted or abstained from voting for the reference rate are excluded from the ballot	Minor	Acknowledged
7	Getting vested or locked coins ignores multiple vesting schedules for the same denomination	Minor	Resolved
8	Usage of alpha and beta dependencies	Minor	Acknowledged
9	Swaps between MNT and other Terra coins have a higher, undocumented Tobin tax	Informational	Resolved
10	REST API does not allow delegating the oracle feeder to an address other than the validator	Informational	Resolved
11	REST API allows aggregate prevote without a hash, exchange rates and a salt	Informational	Resolved
12	Oracle migration from v04 to v05 skips deprecated pre-votes and votes	Informational	Acknowledged
13	Validators that are not in the active set can vote and pre-vote within oracle	Informational	Resolved
14	Oracle's tally function contains unnecessary storage calls	Informational	Resolved
15	Sorting of vote targets in oracle is inefficient	Informational	Resolved
16	Static slash window results in uneven slashing at	Informational	Acknowledged

	window transition		
17	Epoch state migration retains empty entries for cumulative epochs	Informational	Resolved
18	Gov state migration retains empty entries for open tax rate and reward weight update proposals	Informational	Resolved
19	Code id provided in store code message is ignored	Informational	Resolved
20	WASM gas parameters might allow DOS attacks	Informational	Acknowledged

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	Medium	Documentation is outdated and diverges from the implementation in several places (e. g. in `x/market/spec/*` and `x/oracle/spec/*`)
Test Coverage	High	-

Detailed Findings

1. Human/canon address conversion allows attack that may halt block production

Severity: Critical

In `x/wasm/keeper/api.go:13`, the conversion function for canon to human addresses returns a gas cost of 0 if the canon address is not in the correct address format. Similarly, the human to canon address conversion function in `x/wasm/keeper/api.go:21` returns a gas cost of 0 if the human address is not valid Bech32. This allows an attacker to create, deploy and instantiate a CosmWasm contract that does many invalid address conversions in a loop. Such an attack may cause block production to halt.

Recommendation

We recommend returning a positive gas cost even in the error case of invalid address formats.

Status: Resolved

2. Code migration allows code creator to change behavior without contract admin's consent

Severity: Critical

The `MigrateCode` function in `x/wasm/keeper/contract.go:66` can be used by the code creator to migrate the code if the `CodeHash` is empty. The migration from v04 to v05 sets the `CodeHash` of all contracts to an empty slice in `x/wasm/legacy/v05/migrate.go:23`, allowing all stored codes to be migrated. This is a necessity due to breaking changes introduced by the upgrade of CosmWasm to v0.14.x. The issue here is that the code creator can unilaterally migrate the code without the consent of any of the admins that control contracts depending on that code. That could cause the code creator to accidentally introduce breaking changes or bugs, but, more severely, it also allows the code creator to intentionally add backdoors. Such bugs/backdoors might be unnoticed until value has been lost.

Recommendation

We recommend changing the code migration process to require an explicit opt-in from contract admins/deployers. Alternatively, we recommend supporting multiple CosmWasm versions to remove the need to redeploy any code.

Status: Acknowledged

The Terra team acknowledges this issue but states that almost all projects deploy their own contracts, which implies that there should be very few cases where this issue could pose a security threat.

3. Slash windows may be skipped, leading to higher miss counters and higher slashing

Severity: Major

In `x/oracle/abci.go:21`, the end blocker exits early if the `VotePeriod` is not ending in the current block. In `x/oracle/abci.go:106`, slashing happens if the `SlashWindow` is ending in the current block. If `SlashWindow` is not a multiple of `VotePeriod`, these conditions imply that some slash windows are skipped, leading to higher miss counters and higher slashing.

Recommendation

We recommend performing the check for an ending slash window in every block.

Status: Resolved

4. Users might overpay for swap and swap send messages from Terra to Luna

Severity: Major

In `x/market/keeper/params.go:17`, the `BurnBasePool` method erroneously returns the mint base pool, instead of the burn base pool. That causes wrong spreads to be calculated, which leads to users potentially paying too much for their swaps.

Recommendation

We recommend correcting the param key from `types.KeyMintBasePool` to `types.KeyBurnBasePool`.

Status: Resolved

5. Swap simulation may return different amount from actual swap

Severity: Minor

The implementation of decimal truncation differs between swap simulation and the actual swap. In the swap simulation in `x/market/keeper/swap.go:184`, the swap amount is

truncated after applying the fee, while in the actual swap in `x/market/keeper/msg_server.go:103`, truncation happens to the swap amount, then the decimals are added to the fee, and then the fee is subtracted.

This difference in truncation may lead to a difference in the amount.

Example: `swapCoin = 10.7, fee = 1.6`. In the simulation, that would result in `truncate(10.7 - 1.6) = truncate(9.1) = 9`, while in the actual swap, it would result in `truncate(10 - truncate(1.6 + 0.7)) = truncate(10 - truncate(2.3)) = truncate(10-2) = 8`.

Recommendation

We recommend using the same truncation method in both functions.

Status: Acknowledged

The Terra team acknowledges this difference but states that it is insignificant because it is caused by decimal truncation only, and will be handled correctly in the actual swap.

6. Voters that have not voted or abstained from voting for the reference rate are excluded from the ballot

Severity: Minor

The logic in `x/oracle/types/ballot.go:56` implies that any voter that has either not voted on the reference rate or abstained from voting on it is excluded from the ballot.

This implies a lower amount of turnout in the ballot, which could lead to less efficient voting results.

Recommendation

We recommend allowing different reference rates to increase the information efficiency of ballots.

Status: Acknowledged

In practice, most validators report prices most of the time, so information inefficiency is insignificant. Additional detail on the change to use a reference rate has been provided in PR #345.

7. Getting vested or locked coins ignores multiple vesting schedules for the same denomination

Severity: Minor

In the `GetVestedCoins` function in `x/vesting/types/vesting_account.go:77`, `GetVestingSchedule` is called, which returns the first entry in `VestingSchedules` that matches a `denom`. Any additional entries for that `denom` are ignored. At the same time, `cmd/terrad/genaccounts.go:153` allows creation of multiple vesting schedules with the same `denom`. That implies that users will be unable to access vested coins of any additional entries with the same denomination. We classify this issue as minor, since it can only be caused during genesis.

Recommendation

We recommend either changing the `GetVestedCoins` function to consider all vesting schedules for the same `denom`, or changing vesting generation to prevent adding multiple vesting schedules with the same `denom`.

Status: Resolved

8. Usage of alpha and beta dependencies

Severity: Minor

Terra Core depends on a beta release of Cosmos SDK (`github.com/cosmos/cosmos-sdk v0.43.0-beta1`) and an alpha release of IBC (`github.com/cosmos/ibc-go v1.0.0-alpha2`), which might still contain security issues.

Recommendation

We recommend using only stable releases of dependencies to decrease the probability of vulnerabilities.

Status: Acknowledged

The Terra team plans to upgrade those dependencies to stable releases in the near future.

9. Swaps between MNT and other Terra coins have a higher, undocumented Tobin tax

Severity: Informational

Swaps between Terra coins have a default Tobin tax of 0.25% applied. There is one exception in `x/oracle/types/params.go:42`, MNT, which has a Tobin tax of 2%, which is 8 times

the default Tobin tax. That difference is not mentioned in the documentation. Any swap between MNT and another Terra coin will be subject to that higher Tobin tax.

The rationale for the higher tax for MNT swaps is a higher volatility of that currency.

This is not a security concern, but might be unexpected to users.

Recommendation

We recommend documenting any Tobin tax deviating from the default, or linking to the current on-chain Tobin taxes in the documentation.

Status: Resolved

10. REST API does not allow delegating the oracle feeder to an address other than the validator

Severity: Informational

In `x/oracle/client/rest/tx.go:64`, an error is returned from the REST handler for feeder delegation if the feeder is different from the voter/validator. Consequently, only the voter/validator can be set as the feeder.

This is a bug in the REST implementation, rather than a security concern, since feeders can be delegated correctly via the CLI.

Recommendation

We recommend removing the condition in `x/oracle/client/rest/tx.go:63`.

Status: Resolved

11. REST API allows aggregate prevote without a hash, exchange rates and a salt

Severity: Informational

The REST API for submitting an aggregate prevote in `x/oracle/client/rest/tx.go:79` does not return an error if there is no hash as well as no exchange rates and no salt.

This is not a security concern, but rather an inconvenience for API users.

Recommendation

We recommend changing the else statement in `x/oracle/client/rest/tx.go:112` to `} else if len(req.Hash) > 0 {` and adding an additional else clause to return an error if there is no hash provided.

Status: Resolved

12. Oracle migration from v04 to v05 skips deprecated pre-votes and votes

Severity: Informational

In `x/oracle/legacy/v05/migrate.go:26` the deprecated fields `ExchangeRatePrevotes` and `ExchangeRateVotes` from v04 are skipped and not added as entries to the v05 genesis fields `AggregateExchangeRatePrevotes` and `AggregateExchangeRateVotes`. That implies that existing valid (albeit deprecated) votes are lost in the migration.

Recommendation

We recommend adding existing votes in `ExchangeRatePrevotes` and `ExchangeRateVotes` from v04 to the v05 genesis fields `AggregateExchangeRatePrevotes` and `AggregateExchangeRateVotes`.

Status: Acknowledged

The Terra team acknowledges the missing migration for deprecated `ExchangeRatePrevotes` and `ExchangeRateVotes`, but states that those votes are not used anymore since they have long been deprecated.

13. Validators that are not in the active set can vote and pre-vote within oracle

Severity: Informational

In `x/oracle/keeper/msg_server.go:50` and `94`, validators can pre-vote and vote for aggregate exchange rates even if those validators are not in the active validator set.

This issue has been classified as informational since votes from those validators are not considered in the oracle ballot process.

Recommendation

We recommend checking whether the validator is bonded to prevent any previous validators to pre-vote or vote and hence reduce the amount of stored information on-chain.

Status: Resolved

14. Oracle's tally function contains unnecessary storage calls

Severity: Informational

The `ballotIsPassing` function called in `x/oracle/tally.go:72` runs in a loop and contains multiple storage queries to fetch the power reduction and the vote threshold. Both of those values are independent of the items being looped over.

This is not a security concern, but causes unnecessary computation overhead.

Recommendation

We recommend moving the following lines out of the `ballotIsPassing` function to `x/oracle/tally.go:59`:

```
totalBondedPower := sdk.TokensToConsensusPower(
    k.StakingKeeper.TotalBondedTokens(ctx),
    k.StakingKeeper.PowerReduction(ctx))
voteThreshold := k.VoteThreshold(ctx)
thresholdVotes := voteThreshold.MulInt64(
    totalBondedPower).RoundInt()
```

Status: Resolved

15. Sorting of vote targets in oracle is inefficient

Severity: Informational

The ballot is sorted multiple times during the end blocker: The `Tally` function in `x/oracle/tally.go:14` is called once per denomination, and the `WeightedMedian` function in `x/oracle/types/ballot.go:78` is called three times per denomination.

This is not a security concern but causes unnecessary computation overhead.

Recommendation

We recommend sorting the ballot only once, ideally after creating it within the `OrganizeBallotByDenom` function in `x/oracle/keeper/ballot.go:11`. That change also removes the need to check whether the ballot is already sorted in various places.

Status: Resolved

16. Static slash window results in uneven slashing at window transition

Severity: Informational

In the current implementation, slashing only happens at the end of the `SlashWindow` in `x/oracle/abci.go:106`. That implies that validators that miss enough votes within the slash window will be slashed, while validators that miss votes between windows will not be slashed. For example, imagine we have a `SlashWindow` of `100` periods, with a `MinValidPerWindow` fraction of `0.05`. A validator that votes in the first 5 periods of slash window 1, and then again in the last 5 periods of window 2 will miss 190 consecutive periods without being slashed.

Recommendation

We recommend implementing a rolling slash window to ensure at least `MinValidPerWindow` is voted on on an ongoing basis.

Status: Acknowledged

The Terra team plans to improve the slashing window implementation in a future update.

17. Epoch state migration retains empty entries for cumulative epochs

Severity: Informational

In the treasury migration code for v05 in `x/treasury/legacy/v05/migrate.go:33`, a slice is initialized with the length of tax rewards from v04:

```
epochStates := make([]v05treasury.EpochState,
     len(treasuryGenState.TRs))
```

The previous tax rewards `treasuryGenState.TRs` are then iterated over, and the slice is filled with values that come after cumulative rewards by direct assignment:

```
epochStates[i] = v05treasury.EpochState{ // …
```

This approach leaves slice elements `0` to `cumulativeEpochs - 1` empty, which will cause empty entries for epoch `0` in the store when init genesis runs in `x/treasury/genesis.go:34`.

Recommendation

We recommend initializing the slice with:

```
epochStates := make([]v05treasury.EpochState, 0,
     len(treasuryGenState.TRs))
```

And then appending to that slice instead of assigning values to skip cumulative values from the migrated state.

Status: Resolved

18. Gov state migration retains empty entries for open tax rate and reward weight update proposals

Severity: Informational

In the gov migration code for v05 in `custom/gov/legacy/v043/migrate.go:181`, a slice is initialized with the length of proposals from v04:

```
newProposals := make([]v043gov.Proposal,
len(oldGovState.Proposals))
```

The previous proposals `oldGovState.Proposals` are then iterated over, and the slice is filled with values except tax rate and reward weight update proposals by direct assignment:

```
newProposals[i] = v043gov.Proposal{ // …
```

This approach leaves slice elements that were former tax rate or reward weight proposals empty, which will cause proposals with status nil to be inserted when init genesis runs as part of Cosmos SDK.

Recommendation

We recommend initializing the slice with:

```
newProposals := make([]v043gov.Proposal, 0,
     len(oldGovState.Proposals))
```

And then appending to that slice instead of assigning values to skip tax rate and reward weight update proposals from the migrated state.

Status: Resolved

19. Code id provided in store code message is ignored

Severity: Informational

In the `StoreCode` function in `x/wasm/keeper/msg_server.go:25`, the code is stored at the next available code id. The `CodeID` provided in `MsgStoreCode` is ignored.

Recommendation

We recommend removing `CodeID` from `MsgStoreCode`.

Status: Resolved

20. WASM gas parameters might allow DOS attacks

Severity: Informational

In `x/wasm/types/params.go:45`, WASM gas parameters are set. Those values are not determined through benchmarks/simulations. This might lead to certain calls being mispriced, such that their execution would be relatively cheap for the computation they need. If that is the case, an attacker may be able to flood the with messages, causing congestion with relatively low costs.

Recommendation

We recommend running benchmarks/simulations to determine adequate WASM gas parameters.

Status: Acknowledged

The Terra team plans to run benchmarks in an upgrade in the future.