Cryptonics

Audit Report

Mirror Protocol v2

June 22, 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 Terra Capitol to perform a security audit of the smart contracts for Version 2 of the Mirror Protocol (https://mirror.finance/)

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/Mirror-Protocol/mirror-contracts

Commit hash: `74b7a23021f9dca2b01e49e4b7e5dd5e09bb691d`

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 Version 2 of the Mirror protocol, a decentralize finance protocol aimed at providing representations of real world assets on the Terra blockchain. Mirrored assets are minted by collateralized debt positions (CDP).

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

No	Description	Severity	Status
1	Oracle feeders cannot be updated	Critical	Resolved
2	Withdraw, burn and auction will fail on short positions where funds were unlocked by owner	Critical	Resolved
3	Users can block liquidation of their short positions	Critical	Resolved
4	Burning of deprecated assets through users other than the position owner will fail if oracle stops reporting prices	Critical	Resolved
5	Update of the oracle address in the mint contract is not applied to the collateral oracle	Major	Resolved
6	Inflation reward distribution might fail	Minor	Acknowledged
7	Updating staking or lock contracts in the mint contract without migration may lead to failur1es of withdraw, burn and auction messages	Minor	Resolved
8	Withdrawal and staking of voting rewards fails if too many locked balance entries exist	Minor	Acknowledged
9	Withdrawal of voting tokens fails if too many locked balance entries exist	Minor	Acknowledged
10	End price is not used during burning of assets	Minor	Resolved
11	Protocol fee distribution can be blocked by opening many polls	Minor	Resolved
12	Protocol fee rate could be set to a value greater than 1	Minor	Resolved
13	Hardcoded “uusd” reference for liquidity token query	Informational	Resolved
14	Querying collateral asset infos is unbounded	Informational	Acknowledged
15	Weight changes are applied retroactively to last distribution	Informational	Acknowledged
16	Voter weight could be set to a value greater than 1	Informational	Resolved
17	Updated quorum value not validated	Informational	Resolved

18	Updated threshold value not validated	Informational	Resolved
19	Unlocking of funds uses currently configured lockup period	Informational	Resolved
20	Depositing of collateral and burning of assets is possible even if latest price is older than 60 seconds	Informational	Acknowledged
21	Attacker can hijack factory contract between initialization and post initialize message	Informational	Acknowledged
22	Decimal calculations use 9 decimal places, CosmWasm uses 18	Informational	Acknowledged
23	Migration functions contain unbounded loops	Informational	Acknowledged
24	Weights are different in documentation and implementation	Informational	Resolved
25	Documentation lists specific proposal types that are not used	Informational	Acknowledged
26	Participated polls field of token manager is unused	Informational	Acknowledged
27	Overflow checks not enabled for release profile in packages/mirror_protocol/Cargo.toml	Informational	Resolved

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, see below
Test Coverage	High	-

Detailed Findings

1. Oracle feeders cannot be updated

Severity: Critical

In order to remove misbehaving or even compromised oracles, there needs to be a way to update the feeder for an asset. This is even documented in the Mirror documentation, but no such message is currently implemented in `contracts/mirror_oracle/src/contract.rs:34`.

Recommendation

We recommend implementing a message to update the feeder for an asset.

Status: Resolved

Resolved in 1a09c0c

2. Withdraw, burn and auction will fail on short positions where funds were unlocked by owner

Severity: Critical

During a withdraw, burn and auction message, short positions might trigger a release of funds in `contracts/mirror_mint/src/positions.rs:319`, `645` and `847`. That release will lead to an error in `contracts/mirror_lock/src/contract.rs:171` or `196` if the user has previously unlocked all of the funds that can be unlocked through an `UnlockPositionFunds` message. An attacker can exploit this by denying liquidation of positions through the protocol.

Recommendation

We recommend allowing a graceful return of the release message such that withdraw, burn and auction messages succeed even if there are no locked funds or no funds to unlock.

Status: Resolved

Resolved in c4e010a

3. Users can block liquidation of their short positions

Severity: Critical

In `contracts/mirror_lock/src/contract.rs:185` an unbounded loop happens over `locked_funds`. A user could add many funds to the short position, causing that loop to run out of gas, effectively preventing withdrawal, burn and auction of short CDP positions.

Recommendation

We recommend changing the storage to remove the need for the loop.

Status: Resolved

Resolved in c4e010a

4. Burning of deprecated assets through users other than the position owner will fail if oracle stops reporting prices

Severity: Major

During burning of deprecated assets by users other than the position owner, the current asset price is queried from the oracle in `contracts/mirror_mint/src/positions.rs:574`. That query will return an error if the latest price reported by the oracle is older than 60 seconds, leading to a failure of the asset burn message. This is a major issue since prices might not be available for delisted/defaulted securities which prevents the protocol from removing liquidity for those assets.

Recommendation

We recommend removing the `block_time` argument from `load_asset_price` call in `contracts/mirror_mint/src/positions.rs:574` or using the `end_price` of the asset to allow burning with outdated prices.

Status: Resolved

Resolved in ea36552

5. Update of the oracle address in the mint contract is not applied to the collateral oracle

Severity: Major

During asset registration in the mint contract, the current oracle address is used in a stored query in `contracts/mirror_mint/src/contract.rs:342`. The `update_config` function of the mint contract allows an update of the oracle address in `contracts/mirror_mint/src/contract.rs:227`. While that update sets the current

oracle address correctly, it does not update the stored queries of all existing assets. Consequently, calling the stored query on existing assets will keep using the previous oracle.

The stored query is used in several places:

1. In the `CollateralPrice` query in `contracts/mirror_collateral_oracle/src/contract.rs:228`, which is used in the CDP operations (open, mint, withdraw, burn and auction) and can be used by external users.
2. In the `CollateralAssetInfo` query in `contracts/mirror_collateral_oracle/src/contract.rs:229`, which can be used by external users.

This issue can be mitigated by sending a `UpdateCollateralQuery` message for every asset in the system, but this process is error prone and complex since it needs to be dispatched through governance.

Consequently, we consider this issue to be a major security concern. An oracle that has been compromised or is misbehaving can not effectively be removed from the system with a simple message.

Recommendation

We recommend changing the architecture to no longer rely on a stored query for fetching prices from the oracle. Alternatively, the oracle updating logic could be rewritten to include a migration for existing assets. Depending on the number of assets, such a migration could run out of gas though.

Status: Resolved

Resolved in b191a0e

6. Inflation reward distribution might fail

Severity: Minor

The `Distribute` message of the factory contract uses an unbounded call for asset weights through the `read_all_weight` function in `contracts/mirror_factory/src/state.rs:104`. After that call, the `DepositReward` message is sent to the staking contract, which internally also loops over all the assets in `contracts/mirror_staking/src/contract.rs:119` and `contracts/mirror_staking/src/rewards.rs:66`. As more and more assets are added to Mirror over time, this might lead to reward distribution becoming impossible.

Likewise, the `DistributionInfo` query message is unbounded since it calls the same `read_all_weight` function.

Recommendation

We recommend profiling the amount of gas used for different numbers of assets. If a change is necessary, we recommend inverting the logic such that the factory contract only stores the amounts to be distributed and maintains an index of distributed rewards per asset, while the staking contract can be called with a specific asset as a parameter to collect undistributed rewards.

Additionally, we recommend adding `start_after`, `limit` and `order_by` parameters to the `DistributionInfo` query.

Status: Acknowledged

Message has been split in chunks to prevent a single message containing too many assets. This mitigates the issue for the foreseeable future, but does not solve the underlying problem. To scale Mirror to more assets, the gas usage of this operation will have to be addressed further. The Mirror team committed to monitor gas usage and take measures if this issue becomes a problem.

7. Updating staking or lock contracts in the mint contract without migration may lead to failures of withdraw, burn and auction messages

Severity: Minor

Through the `update_config` call of the mint contract, the staking contract and the lock contract can be changed without a data migration in `contracts/mirror_mint/src/contract.rs:239` and `247`.

A change of the staking contract without migration will lead to an error in the `_decrease_bond_amount` function in `contracts/mirror_staking/src/staking.rs:343`, which is called from the burn or auction messages in the case of short positions.

A change of the lock contract without migration will lead to an error in the `unlock_position_funds` function in `contracts/mirror_lock/src/contract.rs:167`, which is called from the withdraw, burn or auction messages in the case of short positions.

Both of those bugs will lead to the protocol not functioning as expected, which is an issue especially if liquidation cannot be performed anymore.

Recommendation

We recommend removing the ability to update staking or lock contracts, adding migration or allowing the contracts outlined above to pass without returning an error in the cases described.

Status: Resolved

Resolved in dfe5b86

8. Withdrawal and staking of voting rewards fails if too many locked balance entries exist

Severity: Minor

In the `get_withdrawable_polls` function in `contracts/mirror_gov/src/staking.rs:317`, an unbounded iteration over the entries in `locked_balance` is performed. In each iteration, two storage entries are read. If the transaction runs out of gas, the `WithdrawVotingRewards` and `StakeVotingRewards` messages would fail, and it would be impossible for a user to recover and withdraw their rewards.

This issue only affects individual users and the likelihood of the `locked_balance` list growing to the point of the described issue is very low. Hence we classify this issue as minor.

Recommendation

We recommend profiling the application to check how severe this problem is. If needed, the issue could be solved by adding another call to only withdraw/stake voting rewards for a specific poll. That would allow a user to recover.

Status: Acknowledged

The Mirror team committed to monitor gas usage and take measures if this issue becomes a problem.

9. Withdrawal of voting tokens fails if too many locked balance entries exist

Severity: Minor

The `WithdrawVotingTokens` message calls `compute_locked_balance`, which contains an unbounded iteration over the entries in `locked_balance` in `contracts/mirror_gov/src/staking.rs:129`. Since the `WithdrawVotingTokens` message is the only place where the `locked_balance` is cleaned, a user could never recover from that, and could never get their staked tokens back.

This issue only affects individual users and the likelihood of the `locked_balance` list growing to the point of the described issue is very low. Hence we classify this issue as minor.

Recommendation

We recommend profiling the application to check how severe this problem is. There are different approaches to fix it, e. g. a separate message to remove locked balances, a change of the storage such that the largest locked balance is stored rather than computed, or more places where `locked_balance` is cleaned.

Status: Acknowledged

The Mirror team committed to monitor gas usage and take measures if this issue becomes a problem.

10. End price is not used during burning of assets

Severity: Minor

In `contracts/mirror_mint/src/positions.rs:571`, a check is done whether the burnt asset has an `end_price` is set, which if true allows anyone to burn assets for that position. That end_price is however not used in any pricing calculation. Without the end_price being used, there is no incentive for external parties to actually burn assets.

Recommendation

We recommend using the `end_price` during burning of assets through anyone except the position owner.

Status: Resolved

Resolved in 7fccafe

11. Protocol fee distribution can be blocked by opening many polls

Severity: Minor

The `Distribute` message of the collector contract sends the `DepositReward` message to the gov contract. Within that contract, all polls are read in `contracts/mirror_gov/src/staking.rs:153`. That storage read is unbounded and could run out of gas, reverting the distribution. Theoretically, an attacker can block protocol fee distribution by opening many polls. Since opening polls has an economic cost and distribution can be re-triggered at any time by any user, this is a minor security concern.

Recommendation

We recommend modelling the cost of such an attack. If necessary, we recommend adjusting the economic parameters of polls to mitigate this attack, e. g. by increasing the cost per poll quadratically in the number of polls currently in progress.

Status: Resolved

Resolved in 23b1fa1

12. Protocol fee rate could be set to a value greater than 1

Severity: Minor

In the current implementation, the `protocol_fee_rate` could be set to a value greater than 1 in `contracts/mirror_mint/src/contract.rs:255`, which would lead to an underflow panic in `contracts/mirror_mint/src/positions.rs:335` and `contracts/mirror_mint/src/positions.rs:809`.

Recommendation

We recommend adding an assertion to validate that the `protocol_fee_rate` is set to a value smaller than 1 to prevent this issue.

Status: Resolved

Resolved in 3416a23

13. Hardcoded “uusd” reference for liquidity token query

Severity: Informational

In `contracts/mirror_factory/src/contract.rs:413`, a hardcoded reference to `"uusd"` is used to query the liquidity token of the trading pair, while other queries in the codebase use `config.base_denom`.

Recommendation

We recommend using `config.base_denom` in this query.

Status: Resolved

14. Querying collateral asset infos is unbounded

Severity: Informational

The `CollateralAssetInfos` query message is unbounded in `contracts/mirror_collateral_oracle/src/contract.rs:295`, which could

cause calling transactions to run out of gas. Since the query is not used in the current mirror contracts, this issue is just reported here for informational purposes. Nevertheless, external contracts cannot use this query without being exposed to out of gas errors.

Recommendation

We recommend adding `start_after`, `limit` and `order_by` parameters to the query.

Status: Acknowledged

15. Weight changes are applied retroactively to last distribution

Severity: Informational

The distribution logic always uses the latest weight. Within the `UpdateWeight` message, no distribution is triggered. That means that the next distribution will retroactively use the latest weight for the whole period, including the time before the call to update the weight. This could confuse users.

Recommendation

We recommend calling distribute before the weight is updated.

Status: Acknowledged

16. Voter weight could be set to a value greater than 1

Severity: Informational

In the current implementation, the `voter_weight` could be set to a value greater than 1 in `contracts/mirror_gov/src/contract.rs:205`, which would imply that more protocol fee rewards are distributed to voters than have been deposited by the collector contract.

Recommendation

We recommend adding an assertion to validate that the `voter_weight` is set to a value smaller than or equal to 1 to prevent this issue.

Status: Resolved

Resolved in b191a0e and d3f47f4

17. Updated quorum value not validated

Severity: Informational

In `contracts/mirror_gov/src/contract.rs:181`, an update of the `quorum` value is not validated.

Recommendation

We recommend calling `validate_quorum(msg.quorum)?;` before updating the value.

Status: Resolved

18. Updated threshold value not validated

Severity: Informational

In `contracts/mirror_gov/src/contract.rs:185`, an update of the `threshold` value is not validated.

Recommendation

We recommend calling `validate_threshold(msg.threshold)?;` before updating the value.

Status: Resolved

19. Unlocking of funds uses currently configured lockup period

Severity: Informational

Unlocking of funds by the user is only possible if the lockup period is over. The check for that condition in `contracts/mirror_lock/src/state.rs:186` uses the current lockup period, read from the config. Changes to the config will be applied to any funds, even to funds locked in the past. While not a security issue, this might confuse users.

Recommendation

We recommend calculating and store the unlock time, rather than storing the lock time and calculate whether the lockup period is over when unlock is called.

Status: Resolved

20. Depositing of collateral and burning of assets is possible even if latest price is older than 60 seconds

Severity: Informational

According to the documentation, CDP operations (mint, burn, deposit and withdraw) should be disabled if no price update has been received for 60 seconds or more. In the function for depositing collateral in `contracts/mirror_mint/src/positions.rs:206` and in the function for burning assets that are not deprecated in `contracts/mirror_mint/src/positions.rs:527`, that condition is not checked.

Recommendation

We recommend calling `load_asset_price` to implement the check or adjusting the documentation.

Status: Acknowledged

21. Attacker can hijack factory contract between initialization and post initialize message

Severity: Informational

Since `init` and `post_initialize` in `contracts/mirror_factory/src/contract.rs:125` are separate messages, an attacker can hijack the contract by setting itself as owner between those messages. This is not really a security concern, since the honest `post_initialize` message would fail and the contracts could just be redeployed.

Recommendation

We recommend setting the owner in the `init` message and checking whether the initialization has been completed by checking one of the other config values against the `CanonicalAddr::default()` value. Alternatively, the `init` and `post_initialize` messages as well as all the messages in between could be batched into one transaction to ensure they are executed in an atomic way.

Status: Acknowledged

22. Decimal calculations use 9 decimal places, CosmWasm uses 18

Severity: Informational

Decimal functions in `contracts/*/src/math.rs` use a `DECIMAL_FRACTIONAL` of `1_000_000_000`, while the underlying CosmWasm functions use a `DECIMAL_FRACTIONAL`

of `1_000_000_000_000_000_000`. While there is no security risk in this difference above, a loss in precision in calculations might be unexpected for users.

Recommendation

We recommend using a `DECIMAL_FRACTIONAL` of `1_000_000_000_000_000_000` as CosmWasm does.

Status: Acknowledged

Intended by Mirror to prevent overflow during multiplications

23. Migration functions contain unbounded loops

Severity: Informational

The migration functions contain unbounded loops in `contracts/mirror_gov/src/migrate.rs:87`, `160`, `contracts/mirror_mint/src/migration.rs:65` and `contracts/mirror_staking/src/migration.rs:33`. This might cause a security issue if some migrations work, but others fail – leaving the whole protocol in an inconsistent state.

Recommendation

We recommend batching all migrations into one call to ensure they are executed in an atomic way.

Status: Acknowledged

24. Weights are different in documentation and implementation

Severity: Informational

In `contracts/mirror_factory/src/contract.rs:30` and `31`, MIR tokens have a weight of `300`, while other assets have a weight of `30`. The documentation states that MIR tokens have a weight of `3`, while other assets have a weight of `1`.

Recommendation

We recommend updating the documentation for consistency.

Status: Resolved

25. Documentation lists specific proposal types that are not used

Severity: Informational

The documentation at https://docs.mirror.finance/protocol/governance/proposal-types lists a number of proposal types, which are not implemented in `contracts/mirror_gov/src/contract.rs:284`.

Recommendation

We recommend updating the documentation for consistency.

Status: Acknowledged

26. Participated polls field of token manager is unused

Severity: Informational

The `participated_polls` field of the token manager struct defined in `contracts/mirror_gov/src/state.rs:47` is currently unused.

Recommendation

We recommend removing the `participated_polls` field.

Status: Acknowledged

27. Overflow checks not enabled for release profile in `packages/mirror_protocol/Cargo.toml`

Severity: Informational

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

Recommendation

We recommend enabling overflow checks in every package, even if no calculations are currently performed in those packages. That prevents unintended consequences when features are added in the future or when the project is refactored.

Status: Resolved

Resolved in 6e09898