Audit Report

TeFi Oracle Contracts

v1.0

January 14, 2022

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Terraform Labs Pte. Ltd. to perform a security audit of the TeFi Oracle smart contracts.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, 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/tefi-oracle-contracts

Commit hash: `0853b934ff20daf9ab28de8474c4f0a445376e9e`

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 smart contracts implement TeFi Oracle Contracts which aim to provide generalized oracle functionality for TeFi projects running on the Terra blockchain.

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 below.

Note, that high complexity or low 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	Conflict of permissioning prevents feeder from submitting prices	Major	Resolved
2	Incorrect price list query in oracle hub	Minor	Resolved
3	Unbounded number of oracle proxies	Minor	Resolved
4	Oracle query functions contain unbounded loops	Informational	Acknowledged
5	Sorting proxy list on every query is inefficient	Informational	Resolved
6	Legacy price response returns last updated value for quote price in the future	Informational	Acknowledged
7	Overflow checks not enabled for release profile	Informational	Resolved
8	Oracle price query variable naming may be confusing	Informational	Acknowledged
9	Legacy price query does not support different bases	Informational	Acknowledged
10	Legacy price only queries the highest priority proxy	Informational	Acknowledged
11	Price query response of band oracle proxy is misleading	Informational	Acknowledged

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Low-Medium	-
Code readability and clarity	Medium	-
Level of documentation	Low-Medium	-
Test coverage	Medium-High	-

Detailed Findings

1. Conflict of permissioning prevents feeder from submitting prices

Severity: Major

When feeding prices into a price feed the contract checks that the owner is the sender of the message in `contracts/oracle-proxy-feed/src/contract.rs:125`. Subsequently when pushing prices to the relevant asset feeds the contract checks whether the message sender is a registered feeder in line `143`. This means that only the contract owner who also is the feeder can submit prices to the feeder contracts.

Recommendation

We recommend removing the condition on line `125` to allow all registered feeders to submit prices.

Status: Resolved

Resolved in `fa7d5a4`

2. Incorrect price list query in oracle hub

Severity: Minor

The oracle hub query message `PriceList` in `contracts/oracle-hub/src/contract.rs:70` calls the `query_proxy_list` function. This incorrectly returns a list of the registered proxies for a specific asset token and not the latest prices.

Recommendation

We recommend replacing `query_proxy_list` with `query_price_list` in `contracts/oracle-hub/src/contract.rs:70`.

Status: Resolved

Resolved in `fa7d5a4`

3. Unbounded number of oracle proxies

Severity: Minor

The number of proxies that can be registered to the oracle hub is unbounded in `contracts/oracle-hub/src/state.rs:39`. Having a potentially large number of

proxies would make execution functions costly and query functions unperformant. In extreme cases, this could cause the querier to run out of gas.

Recommendation

We recommend setting an upper bound on the number of proxies that can be registered upon deployment with an additional governance function to update the max proxy number should it be necessary in the future.

Status: Resolved

Resolved in `fa7d5a4`

4. Oracle query functions contain unbounded loops

Severity: Informational

The Band and Chainlink query functions contain unbounded loops in `contracts/oracle-proxy-band/src/contract.rs:172` and `contracts/oracle-proxy-chainlink/src/contract.rs:164`. They may use significant computational resources leading to issues in the calling contracts.

Recommendation

We recommend adding pagination to these query functions.

Status: Acknowledged

5. Sorting proxy list on every query is inefficient

Severity: Informational

Every call to `query_price` in `contracts/oracle-hub/src/query.rs:54`, `84` and `118` sorts the proxy list. Doing so on every call is computationally inefficient.

Recommendation

We recommend sorting the proxy list on registration of a new proxy and storing it sorted.

Status: Resolved

Resolved in `fa7d5a4`

6. Legacy price response returns last updated value for quote price in the future

Severity: Informational

The `LegacyPriceResponse` contains `u64::MAX` for the `last_updated_quote` in `contracts/oracle-hub/src/query.rs:126`. A last updated quote value in the future may be unexpected to callers of the query and may cause errors in calling contract if that edge case is not handled.

Recommendation

We recommend returning the `proxy_price.last_updated` for the quote, as is done for the base in line `125`.

Status: Acknowledged

7. Overflow checks not enabled for release profile

Severity: Informational

Packages `contracts/oracle-hub/Cargo.toml`, `contracts/oracle-proxy-band/Cargo.toml`, `contracts/oracle-proxy-chainlink/Cargo.toml`, `contracts/oracle-proxy-feed/Cargo.toml` and `packages/tefi-oracle/Cargo.toml` do not enable `overflow-checks` for the release profile.

While enabled implicitly through the workspace manifest, a future refactoring might break this assumption.

Recommendation

We recommend enabling overflow checks in all packages, including those that do not currently perform calculations, to prevent unintended consequences if changes are added in future releases or during refactoring. Note that enabling overflow checks in packages other than the workspace manifest will lead to compiler warnings.

Status: Resolved

Resolved in [fa7d5a4]

8. Oracle price query variable naming may be confusing

Severity: Informational

The oracle hub price query in `contracts/oracle-hub/src/query.rs:38` exposes a field named `timeframe`, which does not clearly communicate the intent of the field. Moreover, it is not obvious whether `timeframe` should be provided in seconds, blocks or another unit. Since the oracle may be used by many projects, the API should be as self-explanatory as possible.

Recommendation

We recommend using an API that is easier to understand, renaming `timeframe` to `max_age` and documenting that it is measured in seconds. Examples of similar APIs can be found in `cw-plus/packages/utils/src/expiration.rs`.

Status: Acknowledged

The Mirror team states that this naming has been chosen to stay consistent with Anchor contracts.

9. Legacy price query does not support different bases

Severity: Informational

The oracle hub price query in `contracts/oracle-hub/src/query.rs:102` takes both a base and quote asset as an argument. However, if the quote asset is not `base_denom` the contract throws an error in line `108`. This makes the API less user-friendly as requests for prices the oracle has access to would be rejected unexpectedly.

Recommendation

We recommend the function to query both the base and quote assets and then calculate and return the price as the caller expects.

Status: Acknowledged

The Mirror team acknowledges this issue stating that the legacy price query is only supposed to be used by Anchor to prevent migrating interfaces.

10.Legacy price only queries the highest priority proxy

Severity: Informational

The oracle hub legacy price query only queries the highest priority proxy contract in `contracts/oracle-hub/src/query.rs:118`. Therefore the query will fail if the proxy with highest priority does not have a price the query regardless if the other proxy contracts have prices.

Recommendation

We recommend querying the registered proxies by priority as is performed for `query_price` in `contracts/oracle-hub/src/query.rs:54`.

Status: Acknowledged

The Mirror team acknowledges this issue stating that the legacy price query is only supposed to be used by Anchor to prevent migrating interfaces.

11. Price query response of band oracle proxy is misleading

Severity: Informational

The Band Protocol proxy contract price query response returns the rate and time that the rate was last updated. However, the response in fact returns simply the time the base asset was updated in `contracts/oracle-proxy-band/src/contract.rs:211`. This is misleading as the quote asset may not have been updated in the same timeframe.

Recommendation

We recommend the response return the older of `last_updated_base` and `last_updated_quote` on line `contracts/oracle-proxy-band/src/contract.rs:211`.

Status: Acknowledged