Report

v. 1.0

Customer

RedStone

Smart Contract Audit

Oracles Monorepo

8th December 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog **3**

2 Introduction **4**

3 Project scope **5**

4 Methodology **6**

5 Our findings **7**

6 Minor Issues **8**

CVF-1. INFO 8
CVF-2. INFO 8
CVF-3. FIXED 8
CVF-4. FIXED 9
CVF-5. INFO 9
CVF-6. INFO 9
CVF-7. FIXED 10
CVF-8. FIXED 10
CVF-9. INFO 10
CVF-10. INFO 11
CVF-11. INFO 11
CVF-12. INFO 11

1 Changelog

#	Date	Author	Description
0.1	08.12.23	A. Zveryanskaya	Initial Draft
0.2	08.12.23	A. Zveryanskaya	Minor revision
1.0	08.12.23	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

RedStone builds a novel oracle that delivers data feeds for 1,100+ assets, available on Ethereum, Polygon, zkEVM, Avalanche, Arbitrum, and 30+ chains with 10 second update time. Thanks to RedStone's StorageLess architecture, which bypasses expensive EVM-storage, projects can integrate it once and use on all EVM chains.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

packages/on-chain-relayer/contracts/price-feeds/

MergedPriceFeedAdapterCommon.sol

packages/on-chain-relayer/contracts/price-feeds/with-rounds/

MergedPriceFeedAdapterWithRounds.sol

packages/on-chain-relayer/contracts/price-feeds/without-rounds/

MergedPriceFeedAdapterWithoutRounds.sol

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 12 minor issues.

	Info	Fixed
Minor	8	4

Fixed 4 out of 12 issues

6 Minor Issues

CVF-1. INFO

- **Category** Procedural
- **Source** MergedPriceFeedAdapter-WithoutRounds.sol

Description Consider specifying as "^0.8.0" unless there is something special about this particular version.

Client Comment *We use custom errors, that are available starting from 0.8.4. But we'll actually start reuqring ^0.8.14, because of the following bug in evm assembly solidity optimiser in 0.8.13. https://medium.com/certora/overly-optimistic-optimizer-certora-bug-isclosure-2101e3f7994d*

```
3 pragma solidity ^0.8.14;
```

CVF-2. INFO

- **Category** Procedural
- **Source** MergedPriceFeedAdapter-WithoutRounds.sol

Description We didn't review these files.

```
5 import {PriceFeedsAdapterBase, PriceFeedsAdapterWithoutRounds} from
    ↪ "./PriceFeedsAdapterWithoutRounds.sol";
  import {PriceFeedBase, PriceFeedWithoutRounds} from "./
    ↪ PriceFeedWithoutRounds.sol";
  import {IRedstoneAdapter} from "../../core/IRedstoneAdapter.sol";
```

CVF-3. FIXED

- **Category** Documentation
- **Source** MergedPriceFeedAdapter-WithoutRounds.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

```
16 function initialize() public override(PriceFeedBase,
    ↪ PriceFeedsAdapterBase) initializer {}
```

CVF-4. FIXED

- **Category** Readability
- **Source** MergedPriceFeedAdapter-WithoutRounds.sol

Description Should be ”else revert” for readability.

```
31 revert DataFeedIdNotFound(dataFeedId);
```

CVF-5. INFO

- **Category** Procedural
- **Source** MergedPriceFeedAdapter-WithRounds.sol

Description Consider specifying as ”^0.8.0” unless there is something special about this particular version.

Client Comment *We use custom errors, that are available starting from 0.8.4. But we’ll actually start reuqring ^0.8.14, because of the following bug in evm assembly solidity optimiser in 0.8.13. https://medium.com/certora/overly-optimistic-optimizer-certora-bug-isclosure-2101e3f7994d*

```
3 pragma solidity ^0.8.14;
```

CVF-6. INFO

- **Category** Procedural
- **Source** MergedPriceFeedAdapter-WithRounds.sol

Description We didn’t review these files.

```
5 import {PriceFeedBase, PriceFeedWithRounds} from "./
    ↪ PriceFeedWithRounds.sol";
  import {PriceFeedsAdapterBase, PriceFeedsAdapterWithRounds} from "./
    ↪ PriceFeedsAdapterWithRounds.sol";
  import {IRedstoneAdapter} from "../../core/IRedstoneAdapter.sol";
```

CVF-7. FIXED

- **Category** Documentation
- **Source** MergedPriceFeedAdapter-WithRounds.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

```
16 function initialize() public override(PriceFeedBase,
     ↪ PriceFeedsAdapterBase) initializer {}
```

CVF-8. FIXED

- **Category** Readability
- **Source** MergedPriceFeedAdapter-WithRounds.sol

Description Should be ”else revert” for readability.

```
31 revert DataFeedIdNotFound(dataFeedId);
```

CVF-9. INFO

- **Category** Procedural
- **Source** MergedPriceFeedAdapterCommon.sol

Description Consider specifying as ”^0.8.0” unless there is something special about this particular version.

Client Comment *We use custom errors, that are available starting from 0.8.4. But we’ll actually start reuqring ^0.8.14, because of the following bug in evm assembly solidity optimiser in 0.8.13. https://medium.com/certora/overly-optimistic-optimizer-certora-bug-isclosure-2101e3f7994d*

```
2 pragma solidity ^0.8.14;
```

CVF-10. INFO

- **Category** Procedural
- **Source** MergedPriceFeedAdapterCommon.sol

Description We didn't review this file.

```
3 import {IRedstoneAdapter} from "../core/IRedstoneAdapter.sol";
```

CVF-11. INFO

- **Category** Bad naming
- **Source** MergedPriceFeedAdapterCommon.sol

Description Events are usually named via nouns, such as "Answer" or "AnswerUpdate".

Client Comment *We wanted to have exactly the same event as Chainlink emits in their contracts.*

```
7 event AnswerUpdated(int256 indexed current, uint256 indexed roundId,
    ↪ uint256 updatedAt);
```

CVF-12. INFO

- **Category** Suboptimal
- **Source** MergedPriceFeedAdapterCommon.sol

Description This conversion wouldn't be necessary if this contract would implement the "IRedstoneAdapter" interface.

Client Comment *True, but it would introduce additional complexity to the MergedPriceFeedAdapterWithRounds and MergedPriceFeedAdapterWithoutRounds contracts. This is because we would need to explicitly specify the overridden contracts 'IRedstoneAdapter' and 'RedstoneAdapterBase' for common functions such as getDataFeedIndex and getDataFeedIds. In my opinion, the benefits of this modification do not sufficiently outweigh the added complications*

```
12 return IRedstoneAdapter(address(this));
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting