Report

v. 2.0

Customer

RedStone

Smart Contract Audit

Oracle

22nd November 2022

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	21.11.22	A. Zveryanskaya	Initial Draft
0.2	21.11.22	A. Zveryanskaya	Minor revision
1.0	21.11.22	A. Zveryanskaya	Release
1.1	22.11.22	A. Zveryanskaya	Introduction and Our findings are fixed
2.0	22.11.22	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.

- https://docs.redstone.finance/
- https://app.redstone.finance/

3 Project scope

We were asked to review:

- Original Repository
- Fix Repository

Files:

```
/
    CalldataExtractor.sol          ProxyConnector.sol          RedstoneConstants.sol
    RedstoneConsumerBase.sol       RedstoneConsumerBytesBase.sol   RedstoneConsumerNumericBase.sol
    RedstoneDefaultsLib.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.

5 Our findings

We found 3 major, and a few less important issues.All identified Major issues have been fixed.

	Active	Fixed
Major	0	3
Moderate	1	9
Minor	6	22

Fixed 34 out of 41 issues

6 Major Issues

CVF-1. FIXED

- **Category** Flaw
- **Source** CalldataExtractor.sol

Description There is no check to ensure that the calculated signed metadata size don't exceed the calldata size.

Recommendation Consider adding such check.

```
21 return unsignedMetadataByteSize + UNSGINED_METADATA_BYTE_SIZE_BS +
      ↪ REDSTONE_MARKER_BS;
```

CVF-4. FIXED

- **Category** Suboptimal
- **Source** RedstoneConsumerBase.sol

Description Adding "BYTES_ARR_LEN_VAR_BS" here is redundant, as it was already added to "ptr" in the previous line.

```
163 mstore(FREE_MEMORY_PTR, add(ptr, add(BYTES_ARR_LEN_VAR_BS,
      ↪ bytesCount)))
```

CVF-5. FIXED

- **Category** Flaw
- **Source** RedstoneConsumerBase.sol

Recommendation There should be a "break" statement before this closing brace, as there couldn't be several indexed for the same feed ID.

```
218 }
```

7 Moderate Issues

CVF-2. FIXED

- **Category** Unclear behavior
- **Source** ProxyConnector.sol

Description Non-empty data returned by the inner call is forwarded to the original caller as is. Thus, if an inner call returns “Proxy calldata failed” message, then the original caller sees exactly the same output as it the inner call returned empty message.

Recommendation Consider adding some prefix to the error message returned by an inner call to make it possible to distinguish these two cases.

```
116   revert(add(32, result), result_size)
```

```
119 revert("Proxy calldata failed");
```

CVF-6. FIXED

- **Category** Overflow/Underflow
- **Source** CalldataExtractor.sol

Description Underflow is possible here.

Recommendation Consider explicitly checking that the calldata size is big enough. Also consider subtracting “UNSIGNED_METADATA_BYTE_SIZE_BS” instead of “STANDARD_SLOT_BS” and then shifting right the loaded value.

```
19 unsignedMetadataByteSize := calldataload(sub(calldataOffset,
     ↪ STANDARD_SLOT_BS))
```

CVF-7. FIXED

- **Category** Overflow/Underflow
- **Source** CalldataExtractor.sol

Description Underflow is possible here.

Recommendation Consider explicitly checking that the calldata size is big enough. Also consider subtracting “UNSIGNED_METADATA_BYTE_SIZE_BS” instead of “STANDARD_SLOT_BS” and then shifting right the loaded value.

```
18 let calldataOffset := sub(calldatasize(), REDSTONE_MARKER_BS)
```

```
28 let calldataOffset := sub(calldatasize(), calldataNegativeOffset)
   dataPackagesCount := calldataload(sub(calldataOffset,
      ↪ STANDARD_SLOT_BS))
```

```
44 let dataPointCalldataOffset := sub(
     calldatasize(),
     add(
       negativeOffsetToDataPoints,
       mul(add(1, dataPointIndex), add(defaultDataPointValueByteSize,
          ↪ DATA_POINT_SYMBOL_BS))
     )
50 )
```

CVF-8. FIXED

- **Category** Overflow/Underflow
- **Source** CalldataExtractor.sol

Description Overflow is possible here.

Recommendation Consider using safe math.

```
48      mul(add(1, dataPointIndex), add(defaultDataPointValueByteSize,
           ↪ DATA_POINT_SYMBOL_BS))
```

```
52  dataPointValue := calldataload(add(dataPointCalldataOffset,
       ↪ DATA_POINT_SYMBOL_BS))
```

```
69  let negativeCalldataOffset := add(
       ↪ calldataNegativeOffsetForDataPackage, SIG_BS)
```

```
73  negativeCalldataOffset := add(negativeCalldataOffset,
       ↪ DATA_POINTS_COUNT_BS)
```

CVF-9. FIXED

- **Category** Overflow/Underflow
- **Source** CalldataExtractor.sol

Description Underflow and overflow are possible here.

Recommendation Consider using safe math.

```
77  extractedValue := calldataload(sub(calldatasize(), add(
       ↪ negativeOffset, STANDARD_SLOT_BS)))
```

CVF-10. FIXED

- **Category** Overflow/Underflow
- **Source** ProxyConnector.sol

Description Underflow is possible here.

Recommendation Consider using safe math.

```
67  sub(calldatasize(), redstonePayloadByteSize), // offset
```

CVF-11. FIXED

- **Category** Overflow/Underflow
- **Source** RedstoneConsumerBase.sol

Description Overflow is possible here.

Recommendation Consider using safe math.

153
```
add(calldataNegativeOffset,
    ↪ TIMESTAMP_NEGATIVE_OFFSET_IN_DATA_PACKAGE))
```

CVF-12. FIXED

- **Category** Overflow/Underflow
- **Source** RedstoneConsumerBase.sol

Description Overflow and underflow are possible here.

Recommendation Consider using safe math.

170
```
  sub(calldatasize(), add(offset, bytesCount)),
```

177
```
valueFromCalldata := calldataload(sub(calldatasize(), add(offset,
    ↪ STANDARD_SLOT_BS)))
```

CVF-13. INFO

- **Category** Suboptimal
- **Source** RedstoneConsumerBase.sol

Description The complexity of this algorithm is n*m. The complexity could be reduced to n+m by requiring the data points to be sorted by data feed indexes.

Client Comment *After internal discussion we decided not to add this requirement of sorting array, as some clients would require their custom order of requested data feeds. Additionally, we don't perform any "expensive" operations inside the loops, so the gas gains would not be significant.*

```
193 for (uint256 dataPointIndex = 0; dataPointIndex < dataPointsCount;
      ↪ dataPointIndex++) {
```

```
198   for (uint256 dataFeedIdIndex = 0; dataFeedIdIndex < dataFeedIds.
        ↪ length; dataFeedIdIndex++) {
```

CVF-14. FIXED

- **Category** Overflow/Underflow
- **Source** RedstoneConsumerBytesBase.sol

Description Overflow and underflow are possible here.

Recommendation Consider using safe math.

```
169 let negativeOffsetToDataPoints := add(
170   calldataNegativeOffsetForDataPackage,
    DATA_PACKAGE_WITHOUT_DATA_POINTS_BS
  )
  let dataPointCalldataOffset := sub(
    calldatasize(),
    add(
      negativeOffsetToDataPoints,
      mul(add(1, dataPointIndex), add(dataPointValueByteSize,
        ↪ DATA_POINT_SYMBOL_BS))
    )
  )
```

```
187   calldataPtr := add(shl(BITS_COUNT_IN_16_BYTES, calldataOffsetArg),
        ↪  valueByteSize)
```

8 Minor Issues

CVF-15. FIXED

- **Category** Procedural
- **Source** RedstoneDefaultsLib.sol

Recommendation Consider specifying as "^0.8.0" unless there is something special about this particular version. Also relevant for: CalldataExtractor.sol, ProxyConnector.sol, ProxyConnector.sol, RedstoneConstants.sol, RedstoneConsumerBase.sol, RedstoneConsumerBytesBase.sol, RedstoneConsumerNumericBase.sol.

```
3 pragma solidity ^0.8.4;
```

CVF-16. INFO

- **Category** Procedural
- **Source** RedstoneDefaultsLib.sol

Description We didn't review this file.

Client Comment *Ok.*

```
5 import "../libs/NumericArrayLib.sol";
```

CVF-17. FIXED

- **Category** Readability
- **Source** RedstoneDefaultsLib.sol

Recommendation This value could be rendered as "3 minutes".

```
12 uint256 constant DEFAULT_MAX_DATA_TIMESTAMP_DELAY_IN_SECONDS = 3 *
   ↪ 60;
```

CVF-18. FIXED

- **Category** Readability
- **Source** RedstoneDefaultsLib.sol

Recommendation This value could be rendered as “1 minutes”.

```
13 uint256 constant DEFAULT_MAX_DATA_TIMESTAMP_AHEAD_IN_SECONDS = 60;
```

CVF-19. INFO

- **Category** Unclear behavior
- **Source** RedstoneDefaultsLib.sol

Description As we didn’t review the “pickMedian” function, it is unclear, how it reacts on an empty array.

Recommendation Consider checking this case carefully and covering it with tests.

Client Comment *Ok.*

```
33 return NumericArrayLib.pickMedian(values);
```

CVF-20. INFO

- **Category** Suboptimal
- **Source** CalldataExtractor.sol

Recommendation This contract could be turned into a library.

Client Comment *We decided to skip this recommendation with the following reasons: 1. The requirement of specifying the Library name with the constants before each constant would significantly increase the size of the code (as we use constants a lot), thus making it less readable 2. A lot of constants are used inside the inline assembly blocks, so to make them work properly we would need to use approach with copying constants to local variables, which will not work because of the ”Stack too deep” error and would complicate the code without the real need.*

```
13 contract CalldataExtractor is RedstoneConstants {
```

CVF-21. FIXED

- **Category** Suboptimal
- **Source** CalldataExtractor.sol

Description This code subtracts “REDSTONE_MARKER_BS” from the calldata size, and then subtracts “STANDARD_SLOT_BS” from the result.

Recommendation Consider subtracting the sum of these two constants using a single “sub” opcode.

```
18 let calldataOffset := sub(calldatasize(), REDSTONE_MARKER_BS)
   unsignedMetadataByteSize := calldataload(sub(calldataOffset,
      ↪ STANDARD_SLOT_BS))
```

CVF-22. FIXED

- **Category** Suboptimal
- **Source** CalldataExtractor.sol

Recommendation This variable is redundant. Just give a name to the returned value and use it instead.

```
26 uint16 dataPackagesCount;
```

CVF-23. FIXED

- **Category** Suboptimal
- **Source** CalldataExtractor.sol

Recommendation Explicit conversions are redundant, as compiler would do such conversions automatically.

```
82 dataPointsCount = uint256(_dataPointsCount);
   eachDataPointValueByteSize = uint256(_eachDataPointValueByteSize);
```

CVF-24. FIXED

- **Category** Suboptimal
- **Source** ProxyConnector.sol

Recommendation This code could be simplified to: (success, result) = contractAddress.call{value: forwardValue ? msg.value : 0}(message);

```
18 if (forwardValue == true) {
     (success, result) = contractAddress.call{value: msg.value}(message
        ↪ );
20 } else {
     (success, result) = contractAddress.call(message);
   }
```

CVF-25. FIXED

- **Category** Suboptimal
- **Source** ProxyConnector.sol

Recommendation It would be more efficient to copy like this: for { let from := add(BYTES_ARR_LEN_VAR_SS, encodedFunction) let fromEnd := add(from, encodedFunctionBytesCount) let to := add(BYTES_ARR_LEN_VAR_SS, message) } lt (from, fromEnd) { from := add (from, 0×20) to := add (to, 0×20) } { mstore (to, mload (from)) }

```
54 for { encodedFunctionOffset := 0 } lt(encodedFunctionOffset,
      ↪ encodedFunctionBytesCount) {
     encodedFunctionOffset := add(encodedFunctionOffset,
        ↪ STANDARD_SLOT_BS) // going with 32 bytes steps
   } {
     // Copying data from encodedFunction to message 32 bytes at a time
     mstore(
       add(add(BYTES_ARR_LEN_VAR_BS, message), encodedFunctionOffset),
          ↪ // address in memory
60     mload(add(add(BYTES_ARR_LEN_VAR_BS, encodedFunction),
          ↪ encodedFunctionOffset)) // 32 bytes to copy
     )
   }
```

CVF-26. FIXED

- **Category** Suboptimal
- **Source** ProxyConnector.sol

Description The expression “add(BYTES_ARR_LEN_VAR_SS, message)” is calcualted several times.

Recommendation Consider calculating once before the loop.

```
59  add(add(BYTES_ARR_LEN_VAR_BS, message), encodedFunctionOffset), //
       ↪  address in memory
```

```
66 add(message, add(BYTES_ARR_LEN_VAR_BS, encodedFunctionBytesCount)),
      ↪ // address
```

```
74  add(message, add(redstonePayloadByteSize,
       ↪ encodedFunctionBytesCount)),
    BYTES_ARR_LEN_VAR_BS
```

CVF-27. FIXED

- **Category** Suboptimal
- **Source** ProxyConnector.sol

Description The expression “add(BYTES_ARR_LEN_VAR_SS, encodedFunction)” is calculated on every loop iteration.

Recommendation Consider calculating once before the loop.

```
60 mload(add(add(BYTES_ARR_LEN_VAR_BS, encodedFunction),
      ↪ encodedFunctionOffset)) // 32 bytes to copy
```

CVF-28. INFO

- **Category** Procedural
- **Source** RedstoneConstants.sol

Recommendation This contract could be turned into a library.

Client Comment *It would require too many changes and would significantly complicate our code.*

```
11 contract RedstoneConstants {
```

CVF-29. FIXED

- **Category** Suboptimal
- **Source** RedstoneConstants.sol

Description There is no access level specified for these constants, so internal access will be used by default.

Recommendation Consider explicitly specifying an access level.

```
18 uint256 constant STANDARD_SLOT_BS = 32;
   uint256 constant FREE_MEMORY_PTR = 0x40;
20 uint256 constant BYTES_ARR_LEN_VAR_BS = 32;
   uint256 constant FUNCTION_SIGNATURE_BS = 4;
```

```
24 uint256 constant SIG_BS = 65;
   uint256 constant TIMESTAMP_BS = 6;
   uint256 constant DATA_PACKAGES_COUNT_BS = 2;
   uint256 constant DATA_POINTS_COUNT_BS = 3;
   uint256 constant DATA_POINT_VALUE_BYTE_SIZE_BS = 4;
   uint256 constant DATA_POINT_SYMBOL_BS = 32;
30 uint256 constant DEFAULT_DATA_POINT_VALUE_BS = 32;
   uint256 constant UNSGINED_METADATA_BYTE_SIZE_BS = 3;
   uint256 constant REDSTONE_MARKER_BS = 9; // byte size of 0
      ↪ x000002ed57011e0000
```

```
35 uint256 constant TIMESTAMP_NEGATIVE_OFFSET_IN_DATA_PACKAGE = 72; //
      ↪ SIG_BS + DATA_POINTS_COUNT_BS + DATA_POINT_VALUE_BYTE_SIZE_BS
   uint256 constant DATA_PACKAGE_WITHOUT_DATA_POINTS_BS = 78; //
      ↪ DATA_POINT_VALUE_BYTE_SIZE_BS + TIMESTAMP_BS +
      ↪ DATA_POINTS_COUNT_BS + SIG_BS
   uint256 constant DATA_PACKAGE_WITHOUT_DATA_POINTS_AND_SIG_BS = 13;
      ↪ // DATA_POINT_VALUE_BYTE_SIZE_BS + TIMESTAMP_BS +
      ↪ DATA_POINTS_COUNT_BS
```

CVF-30. FIXED

- **Category** Documentation
- **Source** RedstoneConstants.sol

Description The word “dynamic” sounds odd for constants.

Recommendation Consider describing as “derived constants”.

```
34 // "Dynamic" values (based on consts)
```

CVF-31. INFO

- **Category** Procedural
- **Source** RedstoneConsumerBase.sol

Description We didn't review these files.

Client Comment *Ok*

```
8 import "../libs/BitmapLib.sol";
  import "../libs/SignatureLib.sol";
```

CVF-32. FIXED

- **Category** Bad datatype
- **Source** RedstoneConsumerBase.sol

Description The comment says a returned value should be in the range 0..255, while the return type is "uint256".

Recommendation Consider changing the return type to "uint8".

```
28  * @return Unique index for a signer in the range [0..255]
```

```
30 function getAuthorisedSignerIndex(address receviedSigner) public
     ↪ view virtual returns (uint256);
```

CVF-33. FIXED

- **Category** Documentation
- **Source** RedstoneConsumerBase.sol

Description This call may actually have three possible outcomes: true, false, and revert.

Recommendation Consider explaining the difference between "false" and "revert" cases.

```
39 return RedstoneDefaultsLib.isTimestampValid(receivedTimestamp);
```

CVF-34. FIXED

- **Category** Documentation
- **Source** RedstoneConsumerBase.sol

Recommendation “from from”

```
58 * validation, and aggregating values from from different authorised
      ↪ signers into a
```

CVF-35. FIXED

- **Category** Suboptimal
- **Source** RedstoneConsumerBase.sol

Description If an empty bitmap is just a zero value, then this assignment is redundant as all newly allocated arrays are filled with zeros.

```
79 signersBitmapForDataFeedIds[i] = BitmapLib.EMPTY_BITMAP;
```

CVF-36. FIXED

- **Category** Unclear behavior
- **Source** RedstoneConsumerBase.sol

Recommendation The official Solidity documentation doesn’t have this condition and recomments using the following code: function allocate(length) → pos { pos := mload(0×40) mstore(0×40, add(pos, length)) } Also it states that the initial value for the free memory pointer is 0×80, rather than 0×60: https://docs.solidity-lang.org/en/v0.8.17/assembly.html?highlight=0×60#memory-management

```
157 // TODO: check why this condition is added in the official yul
       ↪ documentation
    // if iszero(ptr) {
    //   ptr := 0x60
160 // }
```

CVF-37. INFO

- **Category** Suboptimal
- **Source** RedstoneConsumerBytesBase.sol

Description Implementing this functionality as an abstract contract increases the bytecode size of all the consumers.

Recommendation Consider implementing as a library, that could be deployed once and then used via delegate calls.

Client Comment *Creating a library from this contract would be very complicated as we heavily rely on inheritance in our implementation, which is not supported in libraries.*

```
35 abstract contract RedstoneConsumerBytesBase is RedstoneConsumerBase
      ↪ {
```

CVF-38. FIXED

- **Category** Suboptimal
- **Source** RedstoneConsumerBytesBase.sol

Description This line implicitly checks that the “calldataPointersForValues” array is not empty.

Recommendation Consider making this check explicit by adding a “require” statement.

```
52 bytes calldata firstValue = getCalldataBytesFromCalldataPointer(
      ↪ calldataPointersForValues[0]);
```

CVF-39. FIXED

- **Category** Flaw
- **Source** RedstoneConsumerBytesBase.sol

Description There are no validity checks for these values.

Recommendation Consider checking that calldataOffset + valueByteSize <= the calldata length.

```
83 uint256 calldataOfffset = _getNumberFromFirst128Bits(
      ↪ byteValueCalldataPtr);
   uint256 valueByteSize = _getNumberFromLast128Bits(
      ↪ byteValueCalldataPtr);
```

CVF-40. FIXED

- **Category** Unclear behavior
- **Source** RedstoneConsumerBytesBase.sol

Recommendation Here “or” would be more appropriate than “add”.

```
187 calldataPtr := add(shl(BITS_COUNT_IN_16_BYTES, calldataOffsetArg),
        ↪ valueByteSize)
```

CVF-41. FIXED

- **Category** Suboptimal
- **Source** RedstoneConsumerBytesBase.sol

Recommendation Conversions to “uint256” are redundant.

```
194 return uint256(number >> 128);
```

```
199 return uint256((number << 128) >> 128);
```

CVF-42. FIXED

- **Category** Readability
- **Source** RedstoneConsumerBytesBase.sol

Recommendation This could be simplified to: return uint128 (number);

```
199 return uint256((number << 128) >> 128);
```

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