Report

v. 2.0

Customer

Spectral

Smart Contract Audit

Special Staking

28th April 2025

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	27.02.25	A. Zveryanskaya	Initial Draft
0.2	27.02.25	A. Zveryanskaya	Minor revision
1.0	27.02.25	A. Zveryanskaya	Release
1.1	28.04.25	A. Zveryanskaya	The fix link has been updated.
2.0	28.04.25	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.

Spectral Labs is building the Onchain Agent Economy – a bold new paradigm where anyone can create, own, and govern intelligent agents that autonomously navigate the crypto landscape, execute complex strategies, and seize opportunities 24/7.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

```
/
        SpecialStaking.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.
- **Recommendations** contain code style, best practices and other suggestions.

5 Our findings

We found 2 critical, 2 major, and a few less important issues. All identified Critical and Major issues have been fixed.

Fixed 4 out of 4 issues

6 Critical Issues

CVF-1 FIXED

- **Category** Flaw
- **Source** SpectralStaking.sol

Description In case all stakes for the distribution were made exactly at the end of distribution period, or in case the distribution period length is zero, this check will fail.

Recommendation Change to ">=".

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/ce01ff19df70142ed59321bc053333704e6c6fc3*

```
402 // Ensure result is positive before uint256 conversion
    require(totalWeightSigned > 0, "Total␣weight␣must␣be␣positive");
```

CVF-2 FIXED

- **Category** Flaw
- **Source** SpectralStaking.sol

Description In case a user had no stake during the distribution period, but staked some token in the very second when next distribution was created (but before it was created), then this condition will fail, and such user won't be able to proceed with further distributions.

Recommendation Change to ">=".

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/ce01ff19df70142ed59321bc053333704e6c6fc3*

```
411 // Ensure result is positive before uint256 conversion
    require(totalUserWeightSigned > 0, "Total␣user␣weight␣must␣be␣
       ↪ positive");
```

7 Major Issues

CVF-3 FIXED

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

Description The returned value is ignored.

Recommendation Ensure true is returned or use the “safeTransferFrom” function.

Client Comment *Solved in Commit https://github.com/Spectral-Finance/spectral-staking/commit/5343aeea8ac32e5cf23a83e8716f46492f84cb9f*

```
269 IERC20Upgradeable(rewardTokenA).transferFrom(
```

```
278     IERC20Upgradeable(rewardTokenB).transferFrom(
```

CVF-4 FIXED

- **Category** Procedural
- **Source** SpectralStaking.sol

Description Transferring two different tokens in the same transaction is a bad practice, as if there are some problems with one of the tokens, the user won’t get the other tokens as well.

Recommendation Instead of transferring tokens, update a mapping of how much tokens the user is eligible to withdraw, and let the user to withdraw in a separate call.

Client Comment *Added safeTransfer in Commit https://github.com/Spectral-Finance/spectral-staking/commit/a5812b291f1f47891224c571da92712dbdf3b615 Note: All the tokens being distributed have the same code and implementation via diamond proxy pattern. There shouldn’t be an issue with one token only.*

```
426 IERC20Upgradeable(rewardTokenA).transfer(
```

```
433     IERC20Upgradeable(rewardTokenB).transfer(
```

8 Moderate Issues

CVF-5 FIXED

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

Description In case “rewardTokenB” is zero, the “totalRewardsB” value is silently ignored, which could hide problems.

Recommendation Refactor like this: if (totalRewardsB > 0) { require (rewardTokenB != address (0));

Client Comment *Solved in Commit https://github.com/Spectral-Finance/spectral-staking/commit/9f0a1e217c8923d0b97d70a54fa30a409df510bc*

275
```
if(rewardTokenB != address(0))
{
    require(totalRewardsB > 0, "Reward must be greater than 0");
```

CVF-6 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description Consider a situation when a user had some stakes in the past, but then withdrawn all staked tokens and didn’t have any tokens for a long period of time. If such a user will make a stake again, he will need to claim rewards for all the distributions of the period when he didn’t have any stakes. This could be waste of gas.

Recommendation In case a user with no stakes and with no debt weight in the upcoming distribution, claims reward for the last distribution, reset lastDeposit to zero for such a user, so he will be able to skip further distributions.

Client Comment *Solved by resetting the lastDeposit in the withdraw and the transferCalibration in case user runs out of staking tokens: https://github.com/Spectral-Finance/spectral-staking/commit/5c0b2b9c7b7818d7c739decdb621b8e6ed523d85*

311
```
//Let the user skip the sequential claiming for distributions before
   ↪  their first deposit
if(lastDeposit[msg.sender] == 0)
{
    userDistributions[distributionCount - 1][msg.sender].claimed =
       ↪ true;
    lastClaimedIndex[msg.sender] = distributionCount - 1;
}
```

CVF-7 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description This check may fail only when the protocol is in inconsistent state due to some coding errors and cannot proceed with distributions. Usually, "assert" is used for checks that should never fail.

Recommendation Use "assert" instead of "require" here.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/ce01ff19df70142ed59321bc053333704e6c6fc3*

402
```
// Ensure result is positive before uint256 conversion
require(totalWeightSigned > 0, "Total␣weight␣must␣be␣positive");
```

CVF-8 FIXED

- **Category** Flaw
- **Source** SpectralStaking.sol

Description Here both, the numerator and the denominator are multiplied by the same value, which doesn't have sense.

Recommendation Don't multiply anything by "PRECISION".

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/6252ebf46fd0eadb5a836008e03d715af17efdf0*

415
```
uint256 totalUserWeight = uint256(totalUserWeightSigned).mul(
   ↪ PRECISION);
```

418
```
userRewardA = Math.mulDiv(totalUserWeight, distributions[
   ↪ distributionIndex].totalRewardsA, totalDistributionWeight.mul(
   ↪ PRECISION));
```

432
```
    userRewardB = Math.mulDiv(totalUserWeight, distributions[
       ↪ distributionIndex].totalRewardsB, totalDistributionWeight.
       ↪ mul(PRECISION));
```

CVF-9 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description The "_claim" function performs several checks that doesn't make sense to be performed on every loop iteration. This applies to distributionIndex range checks at lines 360 and 363–366, sequentiality check at lines 369–370, withdraw delay check at lines 373–376, double claiming check at line 384.

Recommendation Refactor the code to performs these checks only once before or after the loop, but not in every loop iteration.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/0e8304466fd33315227cac7d25526424f11b5bf0*

```
452 for (uint256 i = _fromDistributionIndex; i <= _toDistributionIndex;
    ↪ i++) {
     _claim(i);
```

CVF-10 FIXED

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

Description Overflow is possible when converting types.

Recommendation Use safe conversion or explicitly require "amount" to fit into "int256".

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/68105fc40e9f9dbc77c4bd7a2554c41b43776353*

```
465 int userDebtWeight = int(amount) *
```

```
476 userDistributionFrom.potentialStaked -= int256(amount);
    userDistributionTo.potentialStaked += int256(amount);
```

9 Minor Issues

CVF-11 FIXED

- **Category** Procedural
- **Source** SpectralStaking.sol

Recommendation Consider specifying as "^0.8.0" unless there is something special regarding this particular version.

Client Comment *Acknowledged, we are sticking with ^0.8.20 in all our protocol*

```
2 pragma solidity ^0.8.20;
```

CVF-12 FIXED

- **Category** Procedural
- **Source** SpectralStaking.sol

Description We didn't review these files.

Client Comment *Acknowledged*

```
11 import "../interfaces/ISpectralStakingToken.sol";
   import "./SpectralStakingToken.sol";
```

CVF-13 FIXED

- **Category** Procedural
- **Source** SpectralStaking.sol

Description No access level specified for this variable, so internal access will be used by default.

Recommendation Explicitly specify an access level.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/84fc25391bb015e5caa2c46967c5769cb7623bff*

```
28 uint8 version;
```

CVF-14 FIXED

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

Recommendation The type for these fields should be “IERC20”.

Client Comment *Acknowledged, no change needed as no security or efficiency advantage*

```
42 address rewardTokenA;
   address rewardTokenB;
```

```
52 address rewardTokenA;
   address rewardTokenB;
```

CVF-15 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description It seems redundant to have both, the “claimed” field in the “UserDistributionInfo” structure and the “lastClaimedIndex” mapping, as one can be derived from the other.

Recommendation Refactor the code to leave only one of these two things.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/d594dcbccf48676b71330e0990df706eb6fa8250*

```
64     bool claimed; // Whether the user has claimed rewards for this
          ↪ distribution
```

```
70 mapping(address => uint256) public lastClaimedIndex;
```

CVF-16 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Recommendation It would be more efficient to merge these two mappings into a single mapping whose keys are users and values are structs encapsulating the values of the original mappings.

Client Comment *Acknowledged, no change needed as we intend to remove the lastDeposit checks in the future.*

```
68 mapping(address => uint256) public lastDeposit;
```

```
70 mapping(address => uint256) public lastClaimedIndex;
```

CVF-17 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Recommendation It would be more efficient to merge these two mappings into a single mapping whose keys are distribution IDs and values are structs encapsulating the values of the original mappings.

Client Comment *Acknowledged, no change needed as we sometimes access one of the mappings without the other*

```
72 mapping(uint256 => DistributionInfo) public distributions;
```

```
74 mapping(uint256 => mapping(address => UserDistributionInfo)) public
   ↪ userDistributions;
```

CVF-18 FIXED

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

Recommendation The type for keys in this mapping should be “IERC20”.

Client Comment *Acknowledged, no change needed as no security or efficiency advantage*

```
76 mapping(address => mapping (address => DistributionBufferInfo))
   ↪ public distributionBuffer;
```

CVF-19 FIXED

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

Recommendation The type for these parameters should be “IERC20”.

Client Comment *Acknowledged, no change needed as no security or efficiency advantage*

```
88  address indexed rewardTokenA,
    address indexed rewardTokenB,
```

```
97  address rewardTokenA,
    address rewardTokenB,
```

```
103 address rewardTokenA,
    address rewardTokenB,
```

CVF-20 FIXED

- **Category** Procedural
- **Source** SpectralStaking.sol

Recommendation These parameters should be indexed.

Client Comment *Added index to all of them except line 98 due to the 3 index limit Commit https://github.com/Spectral-Finance/spectral-staking/commit/06559bd13c2870ebf8d299f47cda82ae9c1a44ff*

```
97  address rewardTokenA,
    address rewardTokenB,
```

```
103 address rewardTokenA,
    address rewardTokenB,
```

CVF-21 FIXED

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

Recommendation The argument type should be more specific.

Client Comment *Acknowledged, no change needed as no security or efficiency advantage*

```
125 function initialize(address _spectralToken) public initializer {
```

CVF-22 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description This check is redundant, as it is anyway possible to pass a dead spectral token address.

Recommendation Remove this check.

Client Comment *Acknowledged, no change needed*

```
126 require(_spectralToken != address(0), "Invalid␣token␣address");
```

CVF-23 FIXED

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

Description This should emit the “UpdateAdmin” event.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/c3301f59b2f3e364b516cfa014bbdfd2d69bffd2*

```
130 admin = msg.sender;
```

CVF-24 FIXED

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

Recommendation This default value should be a named constant.

Client Comment *Acknowledged, there is a setter function for it and it can’t be const.*

```
132 distributionBufferBundlingPeriod = 2 days;
```

CVF-25 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description This check is redundant, as it is anyway possible to pass a dead admin address.

Recommendation Remove this check.

Client Comment *Acknowledged, no change needed*

```
154 require(_admin != address(0), "Invalid␣admin␣address");
```

CVF-26 FIXED

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

Description This event is emitted even if nothing actually changed.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/288e58e1089e3484d374454a5957fc5a54e13c62*

```
156 emit UpdateAdmin(_admin);
```

CVF-27 FIXED

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

Recommendation The type for these arguments should be “IERC20”.

Client Comment *Acknowledged, no change needed as no security or efficiency advantage*

```
161 address rewardTokenA,
    address rewardTokenB,
```

```
175 address rewardTokenA,
    address rewardTokenB,
```

```
205 address rewardTokenA,
    address rewardTokenB) external onlyAdmin {
```

```
227 address rewardTokenA,
    address rewardTokenB,
```

```
260 address rewardTokenA,
    address rewardTokenB,
```

CVF-28 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description The storage addresses of fields are calculated twice here.

Recommendation Use the “+=” operator.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/6ec7af97e67c8d566e466e7bf1d1f5eefe3e98c4*

```
189 oldDistributionBufferInfo.totalRewardsA = totalRewardsA +
       ↪ oldDistributionBufferInfo.totalRewardsA;
190 oldDistributionBufferInfo.totalRewardsB = totalRewardsB +
       ↪ oldDistributionBufferInfo.totalRewardsB;
```

CVF-29 FIXED

- **Category** Procedural
- **Source** SpectralStaking.sol

Recommendation This should be done in the conditional operator above.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/b7f9d4303b346b1dcb1ba542a77c848a148b67e4*

```
191 oldDistributionBufferInfo.creationTimestamp = oldCreationTimestamp;
```

CVF-30 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description These assignments have no effect.

Recommendation Remove them.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/f5a2edb5f1ce21fbf2c50634a06ba43e74e4a2f4*

```
219 distributionBufferInfo.rewardTokenA = distributionBufferInfo.
    ↪ rewardTokenA;
220 distributionBufferInfo.rewardTokenB = distributionBufferInfo.
    ↪ rewardTokenB;
```

CVF-31 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description This check is redundant, as it is anyway possible to pass a dead reward token address.

Recommendation Remove this check.

Client Comment *Acknowledged, no change needed.*

```
265 require(rewardTokenA != address(0), "Invalid␣reward␣token␣address");
```

CVF-32 FIXED

- **Category** Suboptimal
- **Source** SpectralStaking.sol

Description This check is redundant, as it will anyway be performed again when burning staking tokens below.

Recommendation Remove this check.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/f5a2edb5f1ce21fbf2c50634a06ba43e74e4a2f4*

```
329 require(
330     stakingToken.balanceOf(msg.sender) >= amount,
    "Insufficient␣staked␣amount"
);
```

CVF-33 FIXED

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

Description This function should emit some event.

Client Comment *Solved in commit https://github.com/Spectral-Finance/spectral-staking/commit/efaab7e245d16078287bcce3a880906a95ee80c9*

```
492 function setBufferBundlingPeriod(uint256 _bufferBundlingPeriod)
    ↪ external onlyAdmin {
```

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