Zellic

Spectral Token

Smart Contract Security Assessment

October 13, 2023

Prepared for:

Spectral Finance

Prepared by:

Aaron Esau

Zellic Inc.

Contents

About Zellic

Zellic was founded in 2020 by a team of blockchain specialists with more than a decade of combined industry experience. We are leading experts in smart contracts and Web3 development, cryptography, web security, and reverse engineering. Before Zellic, we founded perfect blue, the top competitive hacking team in the world. Since then, our team has won countless cybersecurity contests and blockchain security events.

Zellic aims to treat clients on a case-by-case basis and to consider their individual, unique concerns and business needs. Our goal is to see the long-term success of our partners rather than simply provide a list of present security issues. Similarly, we strive to adapt to our partners' timelines and to be as available as possible. To keep up with our latest endeavors and research, check out our website zellic.io or follow @zellic_io on Twitter. If you are interested in partnering with Zellic, please contact us at hello@zellic.io.

1 Executive Summary

Zellic conducted a security assessment for Spectral Finance on October 13th, 2023. During this engagement, Zellic reviewed Spectral Token's code for security vulnerabilities, design issues, and general weaknesses in security posture.

1.1 Goals of the Assessment

In a security assessment, goals are framed in terms of questions that we wish to answer. These questions are agreed upon through close communication between Zellic and the client. In this assessment, we sought to answer the following questions:

- Can a whitelisted address transfer tokens early (i.e. in the 12 month cliff period)?
- Is it possible to bypass the allocation tracking mechanism?
- Are ERC20 functions properly overridden?

1.2 Non-goals and Limitations

We did not assess the following areas that were outside the scope of this engagement:

- Governance contracts
- The veSpec contract
- Front-end components
- Infrastructure relating to the project
- Key custody

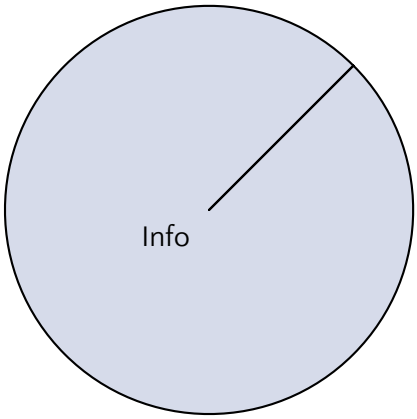
1.3 Results

During our assessment on the scoped Spectral Token contracts, we discovered two findings, all of which were informational in nature.

Additionally, Zellic recorded its notes and observations from the assessment for Spectral Finance's benefit in the Discussion section (4) at the end of the document.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	0
Informational	2



2 Introduction

2.1 About Spectral Token

Spectral Finance is a decentralized solver network leveraging zero knowledge machine learning to guarantee integrity and quality of machine learning models that solve real-world problems.

2.2 Methodology

During a security assessment, Zellic works through standard phases of security auditing including both automated testing and manual review. These processes can vary significantly per engagement, but the majority of the time is spent on a thorough manual review of the entire scope.

Alongside a variety of tools and analyzers used on an as-needed basis, Zellic focuses primarily on the following classes of security and reliability issues:

Basic coding mistakes. Many critical vulnerabilities in the past have been caused by simple, surface-level mistakes that could have easily been caught ahead of time by code review. Depending on the engagement, we may also employ sophisticated analyzers such as model checkers, theorem provers, fuzzers, and so on as necessary. We also perform a cursory review of the code to familiarize ourselves with the contracts.

Business logic errors. Business logic is the heart of any smart contract application. We examine the specifications and designs for inconsistencies, flaws, and weaknesses that create opportunities for abuse. For example, these include problems like unrealistic tokenomics or dangerous arbitrage opportunities. To the best of our abilities, time permitting, we also review the contract logic to ensure that the code implements the expected functionality as specified in the platform's design documents.

Integration risks. Several well-known exploits have not been the result of any bug within the contract itself; rather, they are an unintended consequence of the contract's interaction with the broader DeFi ecosystem. Time permitting, we review external interactions and summarize the associated risks: for example, flash loan attacks, oracle price manipulation, MEV/sandwich attacks, and so on.

Code maturity. We look for potential improvements in the codebase in general. We look for violations of industry best practices and guidelines and code quality

standards. We also provide suggestions for possible optimizations, such as gas optimization, upgradeability weaknesses, centralization risks, and so on.

For each finding, Zellic assigns it an impact rating based on its severity and likelihood. There is no hard-and-fast formula for calculating a finding's impact. Instead, we assign it on a case-by-case basis based on our judgment and experience. Both the severity and likelihood of an issue affect its impact. For instance, a highly severe issue's impact may be attenuated by a low likelihood. We assign the following impact ratings (ordered by importance): Critical, High, Medium, Low, and Informational.

Zellic organizes its reports such that the most important findings come first in the document, rather than being strictly ordered on impact alone. Thus, we may sometimes emphasize an "Informational" finding higher than a "Low" finding. The key distinction is that although certain findings may have the same impact rating, their *importance* may differ. This varies based on various soft factors, like our clients' threat models, their business needs, and so on. We aim to provide useful and actionable advice to our partners considering their long-term goals, rather than a simple list of security issues at present.

Finally, Zellic provides a list of miscellaneous observations that do not have security impact or are not directly related to the scoped contracts itself. These observations — found in the Discussion (4) section of the document — may include suggestions for improving the codebase, or general recommendations, but do not necessarily convey that we suggest a code change.

2.3 Scope

The engagement involved a review of the following targets:

Spectral Token Contracts

Repository	https://github.com/Spectral-Finance/spectral-token
Version	spectral-token: `34972edda4ce10a698d89e3ccde0e2e1379d708d`
Program	SpecToken
Type	Solidity
Platform	EVM-compatible

2.4 Project Overview

Zellic was contracted to perform a security assessment with one consultant for a total of one person-day. The assessment was conducted over the course of one calendar day.

Contact Information

The following project manager was associated with the engagement:

Chad McDonald, Engagement Manager
chad@zellic.io

The following consultant was engaged to conduct the assessment:

Aaron Esau, Engineer
aaron@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

October 13, 2023	Start of primary review period
October 13, 2023	End of primary review period

3 Detailed Findings

3.1 Potential integer underflow in `calculateAllocation`

- **Target**: SpecToken
- **Category**: Coding Mistakes
- **Likelihood**: N/A
- **Severity**: Informational
- **Impact**: Informational

Description

The `addToWhitelist` function allows the owner to update the allocation amount at any time during the `whitelistAddEnd` period. The `totalAllocation` variable is calculated as follows:

```
uint256 totalAllocation = currentMonth
    * allocations[_account].monthlyAllocation
    + allocations[_account].initialAllocation;
```

If the value of `totalAllocation` is less than the `totalSpent[_account]` total amount (i.e., the amount of token that has been transferred out, other than that transferred to the `veTokenMigrator` address), the following calculation will underflow, causing a reversion:

```
return totalAllocation - totalSpent[_account];
```

This may happen if the owner calls `addToWhitelist` and decreases the `initialAllocation` or `monthlyAllocation` amounts.

Impact

The `calculateAllocation` function provides less configurability than likely intended as the owner cannot always decrease the allocation configuration.

Regardless, we recommend preventing underflows to improve correctness (enabling formal verification in the future) and make errors more easily debuggable.

Recommendations

Use the maximum value between `0` and `totalAllocation - totalSpent[_account]`:

```
function calculateAllocation(address _account)
    public view returns (uint256) {
    uint256 currentMonth = calculateCurrentMonth();
    if(currentMonth < 12){
        return 0; //12 month cliff
    }
    uint256 totalAllocation = currentMonth
    * allocations[_account].monthlyAllocation
    + allocations[_account].initialAllocation;
    if (totalAllocation < totalSpent[_account]) return 0;
    return totalAllocation - totalSpent[_account];
}
```

Remediation

This issue has been acknowledged by Spectral Finance, and a fix was implemented in commit d39c89a3.

3.2 Centralization risks of the owner

- **Target**: SpecToken
- **Category**: Protocol Risks
- **Likelihood**: N/A
- **Severity**: Informational
- **Impact**: Informational

Description

Note that the owner of the `SpecToken` contract has the ability to (including, but not limited to):

- pause ERC-20 transfers at any time
- upgrade the contract to potentially mint coins
- block specific users by setting a large allocation configuration on specific users such that `calculateAllocation` reverts with an integer overflow error

Recommendations

Ensure the contract owner is a trustworthy governance contract, or otherwise prominently document and accept the centralization risks.

Remediation

This issue has been acknowledged by Spectral Finance.

4 Discussion

The purpose of this section is to document miscellaneous observations that we made during the assessment. These discussion notes are not necessarily security-related, and do not convey that we are suggesting a code change.

4.1 Misleading `initialAllocation` name

Note that the allocation configuration variable `initialAllocation` in the `TotalAllocation` struct is not actually the initial allocation; rather, the initial allocation after 12 months is `12 * allocations[_account].monthlyAllocation + allocations[_account].initialAllocation`.

This is because the allocation begins after a 12 month cliff period:

```
if(currentMonth < 12){
    return 0; //12 month cliff
}
uint256 totalAllocation = currentMonth
    * allocations[_account].monthlyAllocation
    + allocations[_account].initialAllocation;
```

Note that Spectral Finance renamed this variable to `firstAllocation` in commit d39c89a3.

5 Assessment Results

At the time of our assessment, the reviewed code was not deployed to the Ethereum Mainnet.

During our assessment on the scoped Spectral Token contracts, we discovered two findings, all of which were informational in nature. Spectral Finance acknowledged all findings and implemented fixes.

5.1 Disclaimer

This assessment does not provide any warranties about finding all possible issues within its scope; in other words, the evaluation results do not guarantee the absence of any subsequent issues. Zellic, of course, also cannot make guarantees about any code added to the project after the version reviewed during our assessment. Furthermore, because a single assessment can never be considered comprehensive, we always recommend multiple independent assessments paired with a bug bounty program.

For each finding, Zellic provides a recommended solution. All code samples in these recommendations are intended to convey how an issue may be resolved (i.e., the idea), but they may not be tested or functional code. These recommendations are not exhaustive, and we encourage our partners to consider them as a starting point for further discussion. We are happy to provide additional guidance and advice as needed.

Finally, the contents of this assessment report are for informational purposes only; do not construe any information in this report as legal, tax, investment, or financial advice. Nothing contained in this report constitutes a solicitation or endorsement of a project by Zellic.