Zellic

SpiceFiNFT4626

Smart Contract Security Assessment

January 26, 2023

Prepared for:

Niyant Narang

Spice Finance Inc.

Prepared by:

Ayaz Mammadov and Filippo Cremonese

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 Spice Finance Inc. from January 4th to January 9th, 2022. During this engagement, Zellic reviewed SpiceFiNFT4626's code for security vulnerabilities, design issues, and general weaknesses in security posture.

Please note that at Spice Finance Inc.'s request, due to an upcoming launch, this report covers only the SpiceFiNFT4626 contract. There will be a separate report issued covering the two other contracts in the audit scope - SpiceLending and Note.

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:

- Is it possible for a borrower to steal or obtain an unfair loan from a lender?
- Is it possible for a lender to steal from a borrower or force them to undertake a loan?
- Are funds posted as collateral safe?
- Is the liquidation mechanism sound?
- Is the loan renegotiation feature safe?

1.2 Non-goals and Limitations

Due to the time-boxed nature of security assessments in general, there are limitations in the coverage an assessment can provide. As such, the following items are out of scope for this assessment:

- Attacks involving a compromise or malicious action of SpiceFi and issues involving human error, for example due to improper key custody
- Problems relating to the front-end components and infrastructure of the project, to which we did not have access
- Issues stemming from code or infrastructure outside of the assessment scope

1.3 Results

During our assessment on the scoped SpiceFiNFT4626 contract, we discovered one issue of medium impact.

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	1
Low	0
Informational	0

2 Introduction

2.1 About SpiceFiNFT4626

Spice Lending is a set of smart contracts facilitating yield generation through aggregating lender side liquidity for NFT-backed loans. Users who deposit funds to earn yield also receive NFT receipt tokens that can be further collateralized to earn additional yield through leverage and more general types of loans.

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 the contracts' 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 code base 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 optimiza-

tion, 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.

2.3 Scope

The engagement involved a review of the following targets:

SpiceFiNFT4626 Contracts

Repository	https://github.com/teamspice/vault
Versions	`547ee77c7bae558e8593710b3dfc57ab5bb5d0fc`
Contracts	• SpiceFiNFT4626
Type	Solidity
Platform	EVM-compatible

2.4 Project Overview

Zellic was contracted to perform a security assessment with two consultants for a total of eight person-days. The assessment was conducted over the course of two calendar weeks.

Contact Information

The following project managers were associated with the engagement:

Chad McDonald, Engagement Manager
chad@zellic.io

The following consultants were engaged to conduct the assessment:

Ayaz Mammadov, Engineer
ayaz@zellic.io

Filippo Cremonese, Engineer
fcremo@zellic.io

2.5 Project Timeline

The key dates of the engagement are detailed below.

January 4, 2023	Start of primary review period
January 9, 2023	End of primary review period

3 Detailed Findings

3.1 Missing access control revocation

- **Target**: `SpiceFiNFT4626`
- **Category**: Coding Mistakes
- **Severity**: Medium
- **Likelihood**: High
- **Impact**: Medium

Description

The `initialize` function assigns the `DEFAULT_ADMIN_ROLE` to the `multisig_` account passed as a parameter to the function. The address is also stored in the aptly named `multisig` member variable.

The `setMultisig` function can be used by authorized callers to update the `multisig` variable; however, the function does not revoke the role assigned to the former multisig address, nor does it grant the role to the new address.

Impact

Rotated multisig addresses might retain unintended roles, and new multisig addresses may not be assigned the correct role.

Recommendations

Ensure the proper roles are assigned and revoked by `setMultisig`.

Remediation

The issue is fixed as of commit 51fdc6e1; the `DEFAULT_ADMIN_ROLE` is revoked from the previous `multisig` account.

4 Threat Model

This provides a full threat model description for various functions. As time permitted, we analyzed each function in the smart contracts and created a written threat model for some critical functions. A threat model documents a given function's externally controllable inputs and how an attacker could leverage each input to cause harm. Not all functions in the audit scope may have been modeled. The absence of a threat model in this section does not necessarily suggest that a function is safe.

4.1 File: SpiceFiNFT4626

Function: `previewDeposit (same as OZ ERC4626)`

Function: `previewMint (same as OZ ERC4626)`

Function: `previewWithdraw (same as OZ ERC4626)`

Function: `previewRedeem (same as OZ ERC4626)`

Function: `deposit (same as OZ ERC4626)`

Function: `mint`

Intended behavior

Accept the user asset and mint the shares to a tokenId of their choosing or mint a new NFT.

Preconditions

If the tokenId is an existing NFT, the caller has to be the owner of the NFT.

Inputs

- `tokenId`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: The NFT token Id
- `assets`
  - **Control**: Full control

- **Authorization**: None
- **Impact**: Amount of shares to receive

Function call analysis

- `previewMint`
  - **What is controllable?**: Everything
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: The amount of asset to take
- `weth.transferFrom`
  - **What is controllable?**: Amount
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `_deposit`
  - **What is controllable?**: tokenId, assets
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `redeem`

Intended behavior

Redeem shares from a given tokenId NFT.

Preconditions

Must not be a revealed NFT, must be withdrawable, and caller has to own the NFT.

Inputs

- `tokenId`
  - **Control**: Full control

  - **Authorization**: None
  - **Impact**: The NFT token Id
- `shares`
  - **Control**: Full Control
  - **Authorization**: None
  - **Impact**: Amount of shares to redeem
- `receiver`
  - **Control**: Full Control
  - **Authorization**: None
  - **Impact**: The person who receives the asset

Function call analysis

- `previewRedeem`
  - **What is controllable?**: Everything
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: The amount of asset to take
- `_convertToAssets`
  - **What is controllable?**: Shares
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Calculate assets with fees
- `_withdraw`
  - **What is controllable?**: tokenId, shares, receiver
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `withdraw`

Intended behavior

Withdraw asset from a given tokenId NFT.

Preconditions

mMst not be a revealed NFT, must be withdrawable, and caller has to own the NFT.

Inputs

- `tokenId`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: The NFT token Id
- `asset`
  - **Control**: Full Control
  - **Authorization**: None
  - **Impact**: Amount of asset to withdraw
- `receiver`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: The person who receives the asset

Function call analysis

- `previewWithdraw`
  - **What is controllable?**: Everything
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: The amount of asset to take
- `_convertToAssets`
  - **What is controllable?**: Shares
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Calculate assets with fees
- `_convertToShares`
  - **What is controllable?**: Assets
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**:

Calculate assets with fees

- `_withdraw`
  - **What is controllable?**: tokenId, asset, receiver
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `deposit`

Intended behavior

Accept the user asset and mint the shares to a tokenId of their choosing or mint a new NFT.

Preconditions

If the tokenId is an existing NFT, the caller has to be the owner of the NFT.

Inputs

- `tokenId`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: The NFT token Id
- `assets`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Amount of asset to invest

Function call analysis

- `previewDeposit`
  - **What is controllable?**: Everything
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: The amount of shares to mint

- `weth.transferFrom`
  - **What is controllable?**: amount
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `_deposit`
  - **What is controllable?**: tokenId, assets
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `deposit` (strategist)

Intended behavior

Invest the assets of this vault into another vault.

Preconditions

`msg.sender` has to be a strategist.

Inputs

- `vault`
  - **Control**: Full control
  - **Authorization**: Checks if the vault is approved, through VAULT_ROLE
  - **Impact**: Vault to invest in
- `assets`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Amount of asset to invest
- `minShares`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Slippage check

Function call analysis

- `_checkRole`
  - **What is controllable?**: Control the address, but nothing else
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Means that the address was not approved
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `asset.safeIncreaseAllowance`
  - **What is controllable?**: vault, amount
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `vault.deposit`
  - **What is controllable?**: assets and receiver
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Reverts if the vault doesn't have enough asset
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `mint` (strategist)

Intended behavior

invest the assets of this vault into another vault.

Preconditions

`msg.sender` has to be a strategist.

Inputs

- `vault`
  - **Control**: Full control
  - **Authorization**: Checks if the vault is approved, through VAULT_ROLE
  - **Impact**: Vault to invest in
- `shares`
  - **Control**: Full control

  - **Authorization**: None
  - **Impact**: Amount of asset to invest
- `maxAsset`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Slippage check

Function call analysis

- `_checkRole`
  - **What is controllable?**: Control the address, but nothing else
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Means that the address was not approved
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `asset.safeIncreaseAllowance`
  - **What is controllable?**: vault, amount
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Nothing
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `vault.mint`
  - **What is controllable?**: shares and receiver
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Reverts if the vault doesn't have enough asset
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `withdraw` (strategist)

Intended behavior

Withdraw capital from invested vault.

Preconditions

`msg.sender` has to be a strategist.

Inputs

- `vault`
  - **Control**: Full control
  - **Authorization**: Checks if the vault is approved, through VAULT_ROLE
  - **Impact**: Vault to invest in
- `assets`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Amount of asset to invest
- `maxAsset`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Slippage check

Function call analysis

- `_checkRole`
  - **What is controllable?**: Control the address, but nothing else
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Means that the address was not approved
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `vault.withdraw`
  - **What is controllable?**: assets, receiver, owner
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Ok
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

Function: `redeem` (strategist)

Intended behavior

Redeem capital from invested vault.

Preconditions

`msg.sender` has to be a strategist.

Inputs

- `vault`
  - **Control**: Full control
  - **Authorization**: Checks if the vault is approved, through VAULT_ROLE
  - **Impact**: Vault to invest in
- `assets`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Amount of asset to invest
- `minAssets`
  - **Control**: Full control
  - **Authorization**: None
  - **Impact**: Slippage check

Function call analysis

- `_checkRole`
  - **What is controllable?**: Control the address, but nothing else
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Means that the address was not approved
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded
- `vault.redeem`
  - **What is controllable?**: assets, receiver, owner
  - **What happens if it reverts, reenters, or does other unusual control flow?**: Ok
  - **If return value is controllable, how is it used and how can it go wrong**: Discarded

5 Audit Results

At the time of our audit, the code was not deployed to mainnet Ethereum.

During our assessment on the scoped SpiceFiNFT4626 contracts, we discovered one issue of medium impact. Spice Finance Inc. acknowledged all findings and implemented fixes.

5.1 Disclaimers

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 additional code added to the assessed project after the audit version of 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 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.

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.