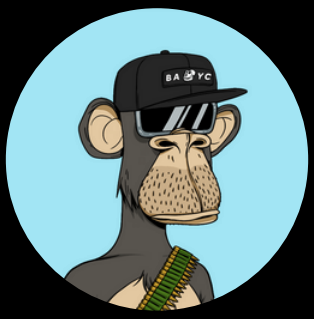
WEBTHREE CONSULTING &
AUDITING

Saffron Finance Fixed Income Audit

MEET OUR TEAM

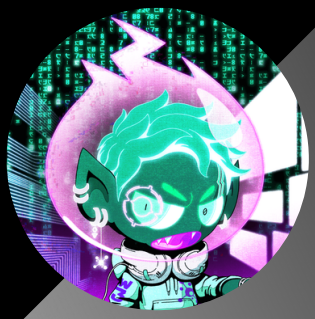
Our mission is to help ensure the integrity of the Web3 space continues to thrive by providing clients with a full suite of services to assist them with the launch of their projects.



Mr_Hmmm
CEO &
Founder

ZooperDooper
Dev & Partner

Ziggy
Dev & Partner



Cristhian
Dev & Partner

Saffron Finance Fixed Income - Security Audit Report

Audited by: ZooperDooper, Stiquit

Date Provided: 4th November 2025

Date Delivered: 18th November 2025

Scope of Work

The scope of this audit is based on the material provided by the client. The goal of this audit is to ensure the following:

- Find potential exploits for the contract
- Find issues in the contract that will affect the integrity of the mint
- Ensure the contract adheres to the business logics provided by the client

Business Requirements

- There are two main parties, fixed income investors and variable income investors
- Fixed income investors provide the liquidity for the pool and receives a % of the total variable income invested in the vault
- Variable Income investors receive a % of the fees generated from the pool, proportional to the amount invested relative to the total
- Withdrawals from either parties can be made prior to the pool starting, and fees withdrawals are distributed once the vault has finished.

Contracts Reviewed

- Github: edaacd47bfd812a1350e7af0952569db41a5025b

Disclaimer

This is a limited report based on our analysis of the Smart Contract audit and performed in accordance with best practices as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the smart contract source code analyzed. The details are set out in this report. In order to get a full view of our findings and the scope of our analysis, it is crucial for you to read the full report.

While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions.

Summary

For the most part the contract is well written and has thorough documentation and unit tests. We found only low severity issues and minor optimizations. All issues have been addressed/acknowledged by the team.

Findings Overview

Severity	Count
Critical	0
Medium	0
Low	0
Nits	0

All issues have been addressed.

Low Issues

Low - Redundant Fixed Bearer Token

Status: Addressed

With the current implementation, the fixed depositor calls the process in the following order: deposit → claim → withdraw. During the deposit process, they are

minted 1 claim token, when the vault has started, they can claim which burns the claim token and mints 1 fixed bearer token. This fixed bearer token when the vault has ended is then burnt to claim the % of the variable deposit. This middle step seems redundant and unnecesary as we are burning 1 to mint 1, and the claim token can be burnt when withdrawing early or withdrawing after the correct time has elapsed. The prerequisite to claim before withdraw introduces unneeded friction for the fixed depositor. The implications are pre-emptive start or never triggering the start.

Low - Discrepancies between variable token and variable bearer token decimals, can lead to misconfiguration and cause potential issues

Status: Addressed

Setting the variable side capacity to a value that is not in the same decimals as the variable token can cause the capacity to be misconfigured and can lead to potential issues. For example, if the variable token is USDC with 6 decimals, and the variable side capacity is set with 18 decimals, then it can result into overcapacity. This is however deemed low as it is a user input and under assumption that the platform sets it correctly. Under the assumption that configurations are correctly set, there will be no issues overall it could be confusing for external third parties without using the saffron finance GUI. Consider normalizing the decimals to the variable asset and using similar names and symbols for pairing tokens.

Nits / Informational

Nits - Replace String Revert Messages with Custom Errors

Status: Addressed

Locations: Vault.sol: 12 requires, VaultFactory.sol: 20 requires, UniV3LimitedRangeAdapter.ols: 19 requires, VaultBearerToken.sol: 4 requires.

```
// Current implementation:
require(initialized, "NI");
require(!initialized, "AI");
```

```
require(msg.sender == factory, "NF");

// Optimized:
error NotInitialized();
error AlreadyInitialized();
error NotFactory();

if (!initialized) revert NotInitialized();
if (initialized) revert AlreadyInitialized();
if (msg.sender != factory) revert NotFactory();
```

Nits - Consider using named imports

Status: Addressed

The codebase contains multiple instances of full file imports, which although functional, are generally not regarded as best practices.

```
import { AdapterBase } from "./adapters/AdapterBase.sol";
```

Nits - Cache Storage Variables In Memory

Status: Addressed

A micro-optimization that can be applied throughout the code base to reduce gas costs.

```
deposit() {
  uint256 variableBearerTokenTotalSupply = variableBearerToken.totalSupply
();
  uint256 claimTokenTotalSupply = claimToken.totalSupply();

  // replace instances of variableBearerToken.totalSupply() with variableBearer
TokenTotalSupply
  // replace instances of claimToken.totalSupply() with claimTokenTotalSupply
}
```

```
claim() {
  uint256 claimTokenTotalSupply = claimToken.totalSupply();

  // replace instances of claimToken.totalSupply() with claimTokenTotalSupply
}
```

Nits - Use Unchecked Math Where Overflow is Impossible

Status: Addressed

A micro-optimization that can be applied to reduce gas costs.

```
// CURRENT
feeDivisor = 10_000 - feeBps;

// OPTIMIZED (feeBps is validated < 10_000)
unchecked {
   feeDivisor = 10_000 - feeBps;
}
```

Report generated by WebThree Consulting