VENUS × PESSIMISTIC

SECURITY ANALYSIS

by Pessimistic

This report is public

May 2, 2025

ABSTRACT

In this report, we consider the security of smart contracts of Venus ERC4626 Wrapper project. Our task is to find and describe security issues in the smart contracts of the platform.

DISCLAIMER

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, a security audit is not investment advice.

SUMMARY

In this report, we considered the security of Venus ERC4626 Wrapper smart contracts. We described the audit process in the section below.

The audit showed several issues of medium severity:
Overpowered access to sweep function, Possible mint of shares using stale state, and Ownership is not transferred to the real owner. Also, several low-severity issues were found.

The overall quality of the codebase is good.

After the initial audit, the codebase was updated. The developers increased the number of tests and code coverage. All issues were fixed or commented.

GENERAL RECOMMENDATIONS

We do not have any additional recommendations.

PROJECT OVERVIEW

Project description

For the audit, we were provided with two public pull requests: VEN-2887 and VEN-2985 on commits c090a20ca3a17ee9953c5d21db01a35d941d226b and 46dc80e664d4e6c96af3a0a794bb45c8078e1d60 respectively.

The scope of the audit included:

- **contracts/ERC4626/Interfaces/IComptroller.sol**;
- **contracts/ERC4626/Interfaces/IRewardsDistributor.sol**;
- **contracts/ERC4626/VenusERC4626.sol**;
- **contracts/ERC4626/VenusERC4626Factory.sol**;
- **contracts/MaxLoopsLimitHelper.sol**;
- **governance-contracts/contracts/Governance/AccessControlledV8.sol**;
- **contracts/Interfaces/IProtocolShareReserve.sol**.

The documentation for the project included private documentation for the audited scope and public technical documentation.

All 38 tests for the provided scope pass successfully. The code coverage for the scope is 74.83%.

The total LOC of audited sources is 375.

Codebase update #1

After the initial audit, the codebase was updated. For the recheck, we were provided with the new pull request on commit faea7132f6d75aa48c38f7d79f62f5ee81caf7cb.

This update included fixes or comments for all issues. The number of tests increased. All 39 tests passed. The code coverage increased to 77.25%.

AUDIT PROCESS

We started the audit on April 21 and finished on April 24, 2025.

We inspected the materials provided for the audit. Then, we contacted the developers for an introduction to the project. After a discussion, we performed preliminary research and specified those parts of the code and logic that require additional attention during an audit:

- ERC4626 standard violations;
- Accurate share distribution among vault participants;
- Defined permissions for the contract owner;
- Compatibility with chains of deployment.

We manually analyzed all the contracts within the scope of the audit and checked their logic.

We scanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules;
- Semgrep rules for smart contracts;
- Audit Agent AI contracts analyzer.

We ran tests and calculated the code coverage.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

After the initial audit, we discussed the results with the developers. On April 28 the developers provided us with an updated version of the code. In this update, most of the issues were resolved, and the remaining ones were addressed with comments.

We reviewed the updated codebase and rescanned the project with the following tools:

- Static analyzer Slither;
- Our plugin Slitherin with an extended set of rules.

Afterwards, we updated the statuses of the issues, and inserted developer comments in the respective sections.

MANUAL ANALYSIS

The contracts were manually analyzed, and their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

The audit showed no critical issues.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Overpowered access to sweep function (addressed)

In the current implementation, `owner` is allowed to sweep any type of token including `vToken` stored in the vault. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig for the `owner` of **VenusERC4626** contracts.

> `Comment from the developers:`
>
> The owner of the VenusERC4626 contracts created by the VenusERC4626Factory will be the owner of the factory.
>
> The owner of the factory deployed by Venus will be the Normal Timelocks listed in the official documentation: Venus Governance Docs
>
> **Timelock Addresses by Network:**
>
> BNB Chain: 0x939bD8d64c0A9583A7Dcea9933f7b21697ab6396
>
> Ethereum: 0xd969E79406c35E80750aAae061D402Aab9325714
>
> opBNB: 0x10f504e939b912569Dca611851fDAC9E3Ef86819
>
> Arbitrum One: 0x4b94589Cc23F618687790036726f744D602c4017
>
> ZKSync Era: 0x093565Bc20AA326F4209eBaF3a26089272627613
>
> Optimism: 0x0C6f1E6B4fDa846f63A0d5a8a73EB811E0e0C04b
>
> Base: 0x21c12f2946a1a66cBFf7eb997022a37167eCf517
>
> Unichain: 0x918532A78d22419Da4091930d472bDdf532BE89a

M02. Possible mint of shares using stale state (fixed)

The `deposit` and `mint` functions rely on stale state by using `exchangeRateStored` within the `totalAssets` function before the exchange rate is updated. This allows fresh depositors to capture a portion of the supplying rewards belonging to previous users, as their share allocation is calculated based on an outdated exchange rate.

To address this, consider calling `accrueInterest()` or `exchangeRateCurrent()` to update the stored exchange rate before performing any calculations or checks such as `maxDeposit`, `maxMint`, `maxWithdraw`, and `maxRedeem` in **VenusERC4626** contract.

> The issues have been fixed and are not present in the latest version of the code.

M03. Ownership is not transferred to the real owner (fixed)

Currently, the **VenusERC4626Factory** contract becomes the `owner` of each **VenusERC4626** instance deployed via the `createERC4626` function. In its current implementation, the factory cannot transfer ownership to another account, nor can it call vault-restricted functions such as `sweepToken`. Consider updating the logic to transfer ownership to a designated managing account immediately after deployment via the factory.

> The issue has been fixed and is not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. Ensure non-zero arguments (commented)

Consider adding `ensureNonzeroAddress` check for `rewardRecipient_` and additional zero check for `loopsLimit_` to revert on unexpected zero values for specified arguments at line 175 in `initialize2` function of **VenusERC4626** contract.

> `Comment from the developers:`
>
> We are already checking this in other calls in nested manner.

L02. Gas optimization (fixed)

Multiple storage variables (`vToken`, `rewardRecipient`, `comptroller`, `rewardRecipient`) are read inside the for-loop. Consider caching them outside of the loop and using cached variables in order to save gas in `claimRewards` function of **VenusERC4626** contract.

> The issues have been fixed and are not present in the latest version of the code.

L03. Indexed event (fixed)

Consider marking `rewardToken` field on `ClaimRewards` event as indexed at line 41 in **VenusERC4626** contract.

> The issue has been fixed and is not present in the latest version of the code.

L04. TODO comment (fixed)

Consider resolving all the todo comments before deployment, as it will improve the quality of the deployed codebase. The comment can be found in line 142 in `createERC4626` function of **VenusERC4626Factory** contract.

> The issue has been fixed and is not present in the latest version of the code.

Notes

N01. Fee on transfer tokens not supported (fixed)

Consider capturing the actual transferred amounts using the difference between pre/post transfer amounts to correctly support fee-on-transfer tokens during `deposit`, `mint`, `withdraw`, and `redeem` operations.

> The issues have been fixed and are not present in the latest version of the code.

N02. Inflation attack

There is a potential for an inflation attack, where an attacker could attempt to skew share distribution by making a minimal deposit before a large asset influx when the vault is empty. However, this vector is mitigated by protections in the OpenZeppelin library, making the attack possible in theory but not profitable in practice.

This analysis was performed by Pessimistic:

Yhtyyar Sahatov, Security Engineer
Oleg Bobrov, Security Engineer
Evgeny Marchenko , Senior Security Engineer
Irina Vikhareva, Project Manager

May 2, 2025