CERTIK

Venus - ERC4626 wrapper

Security Assessment

CertiK Assessed on May 14th, 2025

CertiK Assessed on May 14th, 2025

Venus - ERC4626 wrapper

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEX	Binance Smart Chain (BSC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 05/14/2025	N/A

CODEBASE	COMMITS
PR-497 Base	0e4e03b9ded158397de0a9508487fe153c6020ac
PR-137 Base	46dc80e664d4e6c96af3a0a794bb45c8078e1d60
PR-497 Update1	c090a20ca3a17ee9953c5d21db01a35d941d226b
View All in Codebase Page	View All in Codebase Page

Highlighted Centralization Risks

Contract upgradeability

Vulnerability Summary

14	9	2	0	3	0
Total Findings	Resolved	Timelock	Partially Resolved	Acknowledged	Declined

2	Centralization	2 Timelock	Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
0	Major		Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
6	Medium	6 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

2	Minor	1 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
4	Informational	2 Resolved, 2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | VENUS - ERC4626 WRAPPER

Disclaimer

CODEBASE | VENUS - ERC4626 WRAPPER

Repository

PR-497 Base

PR-137 Base

PR-497 Update1

PR-497 Update2

PR-497 Update5

Commit

0e4e03b9ded158397de0a9508487fe153c6020ac 46dc80e664d4e6c96af3a0a794bb45c8078e1d60
c090a20ca3a17ee9953c5d21db01a35d941d226b 4614d0b851c0cd54715e925a6a16566ec9965f6b
597cb93d1c7149ed8e47780068d6cee03c8fb10b

AUDIT SCOPE | VENUS - ERC4626 WRAPPER

6 files audited ● 1 file with Acknowledged findings ● 2 files with Resolved findings ● 3 files without findings

ID	Repo	File	SHA256 Checksum
VER	VenusProtocol/isolated-pools	VenusERC4626.sol	ef93b7cd5a81ee9ca0c14ab78792c1113f95b34ebc1bce61c290ce071f4bd91e
IPS	VenusProtocol/isolated-pools	Interfaces/IProtocolShareReserve.sol	0051821fbad71559b29a47d0d73b3675348f76fdf908883c2a2729024398522f
VEC	VenusProtocol/isolated-pools	VenusERC4626Factory.sol	bcf5cc025b83586d60a986e002bd5014947dbbd095bf72b4f7fea24bfd3695b9
ICI	VenusProtocol/isolated-pools	Interfaces/IComptroller.sol	36fc4c6c70586e3a6f721bd2ca707af084eb18420fb96ab0e270efe2668cc38c
IRD	VenusProtocol/isolated-pools	Interfaces/IRewardsDistributor.sol	7a7e2d025d3902ce0bf016fbd83bb5d0ef651684a2836c22f2d4ae02986d05f3
IPR	VenusProtocol/protocol-reserve	IProtocolShareReserve.sol	90752b6dadd5701f41ce99bc701c1c250d9b7a4c88d457f1708171d2776efda8

APPROACH & METHODS | VENUS - ERC4626 WRAPPER

This report has been prepared for Venus to discover issues and vulnerabilities in the source code of the Venus - ERC4626 wrapper project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

OVERVIEW | VENUS - ERC4626 WRAPPER

This audit concerns the changes made in files outlined in the following PRs:

- PR-497
- PR-137

Note that any centralization risks present in the existing codebase before these PRs were not considered in this audit and only those added in these PRs are addressed in the audit. We recommend all users carefully review the centralization risks, much of which can be found in our previous audits, which can be found here: https://skynet.certik.com/projects/venus.

PR-497

Adds the `VenusERC4626Factory` and `VenusERC4626` contracts to the isolated pools codebase. In particular the factory contract allows for supported `vTokens` to have a `VenusERC4626` vault created for them utilizing the beacon proxy pattern. The `VenusERC4626` contract allows for users to interact with the associated `vToken` via the `ERC4626` interface. Any rewards accumulated by the funds deposited in the vault can be claimed via the `claimRewards()` function, which sends them to the `rewardRecipient` to be handled and are not necessarily given to depositors of the vault.

PR-137

Adds the income types `ERC4626_WRAPPER_REWARDS` and `FLASHLOAN` to the protocol share reserve allowing these new income types to be handled.

DEPENDENCIES | VENUS - ERC4626 WRAPPER

Third Party Dependencies

The protocol is serving as the underlying entity to interact with third party protocols. The third parties that the contracts interact with are:

- ERC20 Tokens
- Oracles

The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. Moreover, updates to the state of a project contract that are dependent on the read of the state of external third party contracts may make the project vulnerable to read-only reentrancy. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

Recommendations

We recommend constantly monitoring the third parties involved to mitigate any side effects that may occur when unexpected changes are introduced, as well as vetting any third party contracts used to ensure no external calls can be made before updates to its state.

FINDINGS | VENUS - ERC4626 WRAPPER

14	0	2	0	6	2	4
Total Findings	Critical	Centralization	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Venus - ERC4626 wrapper. Through this audit, we have uncovered 14 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
VEW-01	**Centralized Control Of Contract Upgrade**	**Centralization**	**Centralization**	**48h Timelock**
VEW-02	**Centralization Related Risks**	**Centralization**	**Centralization**	**48h Timelock**
VEW-03	Upgradeable Beacon Cannot Be Upgraded Without Upgrading Factory Contract	Logical Issue	Medium	Resolved
VEW-04	Rounding Off Amount When Minting `vToken` Allows Users To Receive Shares Worth More Than Value Received By Vault	Logical Issue	Medium	Resolved
VEW-05	Excess Underlying When Withdrawing Is Not Handled	Logical Issue	Medium	Resolved
VEW-06	Max Withdraw And Redeem Do Not Account For Reserves	Logical Issue	Medium	Resolved
VEW-07	Changing `rewardRecipient` Can Cause Multiple Vaults For Same `vToken`	Logical Issue	Medium	Resolved
VEW-17	`_roundAssets()` Doesn't Round The Amount Of Assets To A Multiple Of The Exchange Rate	Incorrect Calculation	Medium	Resolved
VEW-08	Potential Inflation Attacks	Logical Issue	Minor	Acknowledged
VEW-11	Usage Of Low Level Call	Design Issue	Minor	Resolved

ID	Title	Category	Severity	Status
VEW-09	Not Compatible With Fee On Transfer Tokens	Logical Issue	Informational	Acknowledged
VEW-10	Typos And Inconsistencies	Inconsistency	Informational	Resolved
VEW-12	Missing/Incomplete Natspec Comments	Inconsistency	Informational	Resolved
VEW-15	Initialize Function Comments And Grouping	Coding Style	Informational	Acknowledged

VEW-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	Centralization	VenusERC4626.sol (PR-497 Base): 22; VenusERC4626Factory.sol (PR-497 Base): 17	48h Timelock

Description

In the contract `VenusERC4626Factory` , the role `admin` has the authority to update the implementation contract behind the proxy contract.

In addition, in the `UpgradeableBeacon` the `owner` has the authority to update the implementation contract that the beacon points too.

Any compromise to the `admin` or `owner` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed to by the proxy and therefore execute potential malicious functionality.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.

- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
  AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

- Renounce the ownership and never claim back the privileged role;
  OR
- Remove the risky functionality.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Venus, 04/30/2025]: The DefaultProxyAdmin on each network will be the admins of the Proxy contract of the VenusERC4626Factory. See the DefaultProxyAdmin contracts on each network here: https://docs-v4.venus.io/deployed-contracts/markets. The owner of DefaultProxyAdmin on each network is the Normal Timelock (listed here: https://docs-v4.venus.io/deployed-contracts/governance)

The owner of the VenusERC4626Factory will be the Normal Timelock on each network. The owner of the beacon will be the same wallet (see https://github.com/VenusProtocol/isolated-pools/commit/6bc471622f23bf51dd85aa2f219b7d2bf4af7d07).

VEW-02 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	Centralization	VenusERC4626.sol (PR-497 Update1): 105~116, 122, 134; VenusERC4626Factory.sol (PR-497 Update1): 92~109	48h Timelock

Description

In the contract `VenusERC4626Factory` the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the functions:

- `setRewardRecipient()`
- `setMaxLoopsLimit()`

Any compromise to the `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` may allow a hacker to take advantage of this authority and do the following:

- Set the `rewardRecipient` to an address they control to steal rewards for themselves.
- Set the max loops limit to a large value allowing a denial of service.

In the contract `VenusERC4626`, the function `claimRewards()` claims rewards accumulated by the vaults deposited funds and send them to the `rewardRecipient`. The `rewardRecipient` then handles further distribution of the rewards, which are not necessarily given to the depositors of the vault.

Furthermore, in the contract `VenusERC4626Factory` the role `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` can grant addresses the privilege to call the functions:

- `setMaxLoopsLimit()`
- `setRewardRecipient()`

Any compromise to the `DEFAULT_ADMIN_ROLE` of the `AccessControlManager` may allow a hacker to take advantage of this authority and do the following:

- Set the `rewardRecipient` to an address they control to steal rewards for themselves.
- Set the max loops limit to a large value allowing a denial of service.

Lastly, the role `owner` has privilege over the function `sweepToken()`. Any compromise of the `owner` role may allow a hacker to take advantage of this authority and steal tokens the contract holds for themselves.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Venus, 04/30/2025]: Only the Normal Timelock on each network has the DEFAULT_ADMIN_ROLE on the AccessControlManager contract. See https://docs-v4.venus.io/deployed-contracts/governance for the list of Normal Timelock and AccessControlManager contracts. So, only via a Normal VIP (involving the Venus Community) will be possible to:

- grant permissions to execute the following functions on the VenusERC4626Factory contract: setRewardRecipient, setMaxLoopsLimit
- grant permissions to execute the following functions on the VenusERC4626 contract: setRewardRecipient, setMaxLoopsLimit

Only the Normal Timelocks on each network will have permissions to invoke the setRewardRecipient functions. The Normal, Fast-Track and Critical Timelocks on each network will have permissions to invoke the setMaxLoopsLimit functions.

The rewardRecipient attribute will be defined by the Venus Community via Governance. They will decide what to do with the claimed rewards. It's the expected design.

The owner of the VenusERC4626Factory (and therefore the owner of the VenusERC4626 contracts) will be the Normal Timelock on each network. So, only via Governance will be possible to sweep the tokens.

VEW-03 UPGRADEABLE BEACON CANNOT BE UPGRADED WITHOUT UPGRADING FACTORY CONTRACT

Category	Severity	Location	Status
Logical Issue	Medium	VenusERC4626Factory.sol (PR-497 Base): 80	Resolved

Description

According to the `package.json` file the repository uses version 4.9 of OpenZeppelin contracts. In this version the `UpgradeableBeacon` contract inherits `Ownable`, which sets the `msg.sender` as the owner (**source**). As such it will set the `VenusERC4626Factory` as the owner, which is the only privileged entity allowed to upgrade the implementation the beacon points to via the function `upgradeTo()` (**source**).

The `VenusERC4626Factory` does not have any functionality allowing it to call `upgradeTo()` so that the beacon will not be upgradeable without first upgrading the `VenusERC4626Factory` implementation so that it can call `upgradeTo()`.

Recommendation

We recommend ensuring that the proper account is given the privilege to upgrade the implementation the beacon points to.

Alleviation

`[CertiK, 04/22/2025]`: The client refactored the code so that the owner of the factory is the owner of the beacon in commit 6bc471622f23bf51dd85aa2f219b7d2bf4af7d07.

VEW-04 ROUNDING OFF AMOUNT WHEN MINTING `vToken` ALLOWS USERS TO RECEIVE SHARES WORTH MORE THAN VALUE RECEIVED BY VAULT

Category	Severity	Location	Status
Logical Issue	Medium	VenusERC4626.sol (PR-497 Base): 144~145, 163~164	Resolved

Description

The asset of the `VenusERC4626` vault is the underlying token of a `vToken` , however, the contract holds `vToken` . A user is minted shares according to the amount of underlying token they supply and the amount of `vToken` received by the contract is determined by the `vToken` exchange rate which rounds down. The contract may receive an amount of `vToken` whose value is less than the value of the supplied underlying token by the rounded off amount. As the users shares are calculated based on the underlying amount they supply and not the amount of `vToken` received by the contract, a user may receive an amount of shares whose value is greater than the value the vault receives.

Recommendation

We recommend determining the amount of shares to be minted based on the value received by the vault.

Alleviation

[Venus, 05/14/2025]: The team heeded the advice and resolved the issue by revamping the interaction with vToken in commit 597cb93d1c7149ed8e47780068d6cee03c8fb10b.

VEW-05 EXCESS UNDERLYING WHEN WITHDRAWING IS NOT HANDLED

Category	Severity	Location	Status
Logical Issue	Medium	VenusERC4626.sol (PR-497 Base): 208~210, 263~271	Resolved

Description

When withdrawing or redeeming from the vault it computes an amount of underlying asset to withdraw from the `vToken`. This amount is then withdrawn via the call `vToken.redeemUnderlying(assets)`. If the amount of assets is not a multiple of the exchange rate, then `redeemUnderlying()` will round the amount of `vToken` up, which can cause the amount of underlying asset received to be greater that the input amount. (**source**)

In this case the vault will receive more underlying asset than the input amount, however, only the input amount is transfered and the `totalAsset()` value is based solely on the `vToken` balance of the contract. As such the amount of underlying asset due to the rounding will not be handled and locked in the contract, requiring the contract to be upgraded to be recovered.

Recommendation

We recommend ensuring that any excess amount of underlying token redeemed is properly handled.

Alleviation

[CertiK, 04/22/2025]: The client added a function to sweep tokens in commit d5e37a60930ea301465489ce932936c90db3ddee.

The addition of the sweep function allows for the rounded off amount to be collected. However, the rounded off amount is value taken from the share holders of the vault. If the exchange rate is high enough that the rounded off amount is significant, then this can result in a significant amount of value being taken from the shareholders. We recommend refactoring the code to ensure that this rounded off amount is provided as value to the share holders.

[Venus, 05/06/2025]: The issue was addressed in the commit https://github.com/VenusProtocol/isolated-pools/commit/775f1a7f36c9dd90ab4038188b3ce1086e248588.

VEW-06 MAX WITHDRAW AND REDEEM DO NOT ACCOUNT FOR RESERVES

Category	Severity	Location	Status
Logical Issue	Medium	VenusERC4626.sol (PR-497 Base): 243~245, 257~258	Resolved

Description

The maximum amount of shares that can be redeemed is bounded by the amount of underlying asset currently held by the `vToken` minus the amount of reserves of the `vToken` (**source**). However, the logic of `maxWithdraw()` and `maxRedeem()` only bound the amount by the return value of `getCash()`, which does not account for the reserves of the `vToken` and can return a value that exceeds the actual amount that can be withdrawn or redeemed.

Recommendation

We recommend accounting for the reserves when calculating the maximum amounts that can be withdrawn or redeemed.

Alleviation

`[CertiK, 04/22/2025]`: The client made the recommended changes in commit 4984037e4ff64775d79528a63709e8d853d7294d.

VEW-07 CHANGING `rewardRecipient` CAN CAUSE MULTIPLE VAULTS FOR SAME `vToken`

Category	Severity	Location	Status
Logical Issue	Medium	VenusERC4626Factory.sol (PR-497 Base): 93, 143~144	Resolved

Description

It the `rewardRecipient` is changed, then this changes a parameter that affects the deployment address generated by create2. This allows for multiple vaults to be created for the same `vToken` with the only difference being the vaults `rewardRecipient`.

If this is intended behavior, then the potential issue is that `computeVaultAddress()` will only return the address of the vault for the `vToken` with the currently set `rewardRecipient`.

Note that if the `loopsLimit` is changed during a contract upgrade, then this can also cause multiple vaults for the same `vToken`.

Recommendation

We recommend determining if this is intended behavior, in which case we recommend ensuring that `computeVaultAddress()` can determine the address of all vaults associated with a `vToken` including those with a different `rewardRecipient`.

Furthermore, we recommend considering adding a mapping to track the vaults that have been deployed by the contract, as it may be useful to validate if a vault was generated via the factory contract.

Alleviation

`[CertiK, 04/22/2025]`: The client added a mapping and refactored the code resolving this finding in commit f2e0ece49aa29b504d646ead4fa48bab8758ba75.

VEW-17 `_roundAssets()` DOESN'T ROUND THE AMOUNT OF ASSETS TO A MULTIPLE OF THE EXCHANGE RATE

Category	Severity	Location	Status
Incorrect Calculation	Medium	VenusERC4626.sol (PR-497 Update2): 448~458	Resolved

Description

```
448     function _roundAssets(uint256 assets, Rounding rounding) internal view
returns (uint256) {
449         uint256 exchangeRate = vToken.exchangeRateStored();
450         uint256 redeemTokens = (assets * EXP_SCALE) / exchangeRate;
451
452         uint256 _redeemAmount = (redeemTokens * exchangeRate) / EXP_SCALE;
453
454         if (_redeemAmount != 0 && _redeemAmount != assets && rounding ==
 Rounding.Up) redeemTokens++; // round up
455
456         // redeemAmount = exchangeRate * redeemTokens
457         return (exchangeRate * redeemTokens) / EXP_SCALE;
458     }
```

The code is supposed to "Round the amount of assets, up or down, to a multiple of the exchange rate", however, it doesn't.

The code also breaks the encapsulation principle and assumes the `redeem()` behavior of vToken.

Scenario

1. Let's assume for simplicity that `decimals = 3` and `EXP_SCALE = 1000`.
2. Let's assume that vToken holds 31000 assets and 17000 vTokens. That gives `exchangeRate = 31000 / 17000 = 1823` scaled.
3. Let's assume the user wants to withdraw 7000 assets and calls `previewWithdraw(7000)`. It is supposed to return the rounded-up amount of shares to be burned.
4. Then `redeemTokens = (assets * EXP_SCALE) / exchangeRate = 7000 * 1000 / 1823 = 3839.8 = 3839`.
5. Then `_redeemAmount = (redeemTokens * exchangeRate) / EXP_SCALE = 3839 * 1823 / 1000 = 6998.5 = 6998`.
6. Since 6998 != 7000, the `redeemTokens` is incremented to 3840.
7. The function returns `(exchangeRate * redeemTokens) / EXP_SCALE = 1823 * 3840 / 1000 = 7000.3 = 7000`.
8. As a result, due to the `EXP_SCALE` division on the last step, the resulting number is not a multiple of `exchangeRate`. Furthermore, **it was not rounded up**.

The expected logic of `ERC4626.withdraw()`:

1. The user calls `withdraw(7000)`.
2. `super.previewWithdraw(7000)` returns `(totalSupply + 1) / (totalAssets + 1)` rounded up. Let's say 6789 ERC4626 shares.
3. `beforeWithdraw(7000)` performs `vToken.redeemUnderlying(7000)`. 3840 vTokens of ERC4626 burned. 6995 (for some reason) assets sent back to ERC4626.
4. `actualAssets` is now 6995.
5. `_withdraw()` burns 6789 of user ERC4626 shares. It transfers 6995 assets to the `receiver`.

Recommendation

We recommend removing of `_roundAssets()`.

Alleviation

[Venus, 05/09/2025]: The team heeded the advice and resolved the issue by removing `_roundAssets()` in commit da042048645b3ddeebe6bc5e1409cf9fdd7fc72b.

VEW-08 | POTENTIAL INFLATION ATTACKS

Category	Severity	Location	Status
Logical Issue	Minor	VenusERC4626.sol (PR-497 Base): 22	Acknowledged

Description

Openzeppelin's v4.9 ERC4646 implementation implements virtual assets and shares to mitigate risks of empty or nearly empty vaults. However their implementation does not provide full protection from inflation attacks. In particular, if there a large number of initial deposits that can be included in the same block, then an attacker can still manipulate the rate to profitably steal portions of deposits. In addition, an attacker could manipulate the rate at an initial loss to cause rounding errors that will have them profit more in the long run or prevent users with small amounts of funds from using the contract without losing their funds.

Recommendation

We recommend depositing and locking a non-trivial amount when deploying new ERC4626 vaults to add further protection from inflation attacks.

VEW-11 | USAGE OF LOW LEVEL CALL

Category	Severity	Location	Status
Design Issue	Minor	VenusERC4626.sol (PR-497 Base): 115~124	Resolved

Description

The return value of the low-level call is purposefully not checked in case the recipient is not the protocol share reserve. However, an alternative would be to utilize the `try-catch` pattern in order to avoid the use of a low-level call.

Recommendation

We recommend replacing of low-level call with a try-catch.

Alleviation

[Venus, 05/07/2025]: Changes have been reflected in the commit: https://github.com/VenusProtocol/isolated-pools/commit/9a7a16c274fde157cf9b655494d122b23771c818

VEW-09 | NOT COMPATIBLE WITH FEE ON TRANSFER TOKENS

Category	Severity	Location	Status
Logical Issue	Informational	VenusERC4626.sol (PR-497 Base): 144	Acknowledged

Description

Openzeppelin's ERC4626 implementation expects strict adherence to the ERC20 standard and thus assumes that the amount input when `transfer()` is called is the amount that will be received by the contract (**source**).

In addition, `afterDeposit()` assumes that the input amount of underlying asset will be received by the contract.

Recommendation

We recommend ensuring that no `vToken` has an underlying token where the amount received by the contract after transfer may be different than the input amount.

Alleviation

[Venus, 05/06/2025]: Issue acknowledged. I won't make any changes for the current version.

Venus doesn't support underlying assets with fees on transfer, so the "official" VenusERC4626Factory won't be able to create VenusERC4626 wrappers for a fee on transfer assets.

VEW-10 | TYPOS AND INCONSISTENCIES

Category	Severity	Location	Status
Inconsistency	Informational	Interfaces/IProtocolShareReserve.sol (PR-497 Base): 6~10; VenusERC4626.sol (PR-497 Base): 189; VenusERC4626.sol (PR-497 Update2): 174, 429; IProtocolShareReserve.sol (PR-137 Base): 10	Resolved

Description

In the contract `VenusERC4626` :

- In the function `redeem()` the naming of the last parameter `redeemer` may be misleading. The `redeemer` in this case is the owner of the shares who must have given approval to the caller to redeem the shares on their behalf. The caller is redeeming the shares and is only the `redeemer` if they are the owner of the shares themselves. We recommend considering naming the parameter `owner` .

The interface `IProtocolShareReserve` is not consistent across the codebases. In particular, in the Isolated Pools codebase it does not include the `FLASHLOAN` income type.

Recommendation

We recommend clarifying or fixing the typos and inconsistencies above.

Alleviation

[Venus, 05/13/2025]: Issue acknowledged. Changes have been reflected in the commit hash: https://github.com/VenusProtocol/isolated-pools/commit/ffbe519cda4251f797dcd56b39af67d4d2cfc3fd

VEW-12 | MISSING/INCOMPLETE NATSPEC COMMENTS

Category	Severity	Location	Status
Inconsistency	Informational	VenusERC4626.sol (PR-497 Base): 208~210, 284~296; VenusERC4626Factory.sol (PR-497 Base): 98	Resolved

Description

In the contract `VenusERC4626` :

- The function `totalAssets()` does not have any NatSpec comments.
- The function `_decimalsOffset()` does not have complete NatSpec comments.
- The function `_generateVaultName()` does not have complete NatSpec comments.
- The function `_generateVaultSymbol()` does not have complete NatSpec comments.

Furthermore, the NatSpec comments do not include the events or errors that are emitted, if this is desired, then they must also be added.

In the contract `VenusERC4626Factory` :

- The NatSpec comments for the function `createERC4626()` do not include the events and errors that may be emitted.

Recommendation

We recommend adding in the missing comments or modifying the incomplete comments mentioned above.

Alleviation

`[CertiK, 04/22/2025]` : The client made the recommended changes in commit d955b5efc804194dc9b99feb36e3f908314c1b21.

VEW-15 | INITIALIZE FUNCTION COMMENTS AND GROUPING

Category	Severity	Location	Status
Coding Style	Informational	VenusERC4626.sol (PR-497 Update1): 166~180	Acknowledged

Description

The initialize functions are intended to be called in the same transaction when initializing a new ERC4626 proxy.

Recommendation

We recommend grouping the initialize functions together and adding comments that they must be called in the same transaction when initializing the proxy. This is because if only initialize is called, then any other user could call `initialize2()` to set the access control manager, reward recipient, and loops limit to malicious values.

Alleviation

[Venus, 05/06/2025]: Issue acknowledged. I won't make any changes for the current version.

We have broken down the initialize functions into 2 steps as a mitigation for the VEW-07. There is already a comment in the `initialize` function mentioning the function `initialize2` .

Regarding grouping the "initialize" functions, it would break the solhint rules. We are using solhint 3.3.7, where only the function with the "initializer" modifier can exceptionally be before the external functions.

OPTIMIZATIONS | VENUS - ERC4626 WRAPPER

ID	Title	Category	Severity	Status
VEW-13	Redundant Check	Code Optimization	Optimization	Resolved
VEW-16	Inefficient Use Of Local Variables In Code	Gas Optimization	Optimization	Resolved
VEW-18	`actualShares` Calculation In `_deposit()` Can Be Simplified	Coding Style, Gas Optimization	Optimization	Acknowledged

VEW-13 | REDUNDANT CHECK

Category	Severity	Location	Status
Code Optimization	Optimization	VenusERC4626.sol (PR-497 Update1): 175	Resolved

Description

The check that `accessControlManager_` is not the zero address is unnecessary as it calls `AccessControlledV8.__AccessControlled_init`, which calls `AccessControlledV8._setAccessControlManager()` that has a check to ensure it is nonzero.

Recommendation

We recommend removing the redundant check.

Alleviation

[Venus, 05/06/2025]: Changes have been reflected in the commit: https://github.com/VenusProtocol/isolated-pools/commit/8af3e1a793a28bf48b152fb16f6ebf724a1f4016

VEW-16 | INEFFICIENT USE OF LOCAL VARIABLES IN CODE

Category	Severity	Location	Status
Gas Optimization	Optimization	VenusERC4626.sol (PR-497 Update2): 140~145	Resolved

Description

```
IComptroller _comptroller = comptroller;
VToken _vToken = vToken;
address _rewardRecipient = rewardRecipient;

RewardsDistributor[] memory rewardDistributors =
comptroller.getRewardDistributors();
```

The `IComptroller _comptroller`, `VToken _vToken`, and `address _rewardRecipient` variables are created and assigned but not utilized effectively. The `_comptroller` should be used instead of `comptroller` and `_vToken` is only used once so it can be eliminated for code optimization.

Recommendation

We recommend using `_comptroller` instead of `comptroller` where applicable. Also, evaluate the necessity of `_vToken` since it is used only once. Removing such unnecessary local variables can help to optimize the code.

Alleviation

[Venus, 05/08/2025]: The mitigation has been applied in this commit: https://github.com/VenusProtocol/isolated-pools/commit/98115e020a40af1607c75de1173a7296fdd18f30

VEW-18 `actualShares` CALCULATION IN `_deposit()` CAN BE SIMPLIFIED

Category	Severity	Location	Status
Coding Style, Gas Optimization	Optimization	VenusERC4626.sol (PR-497 Update4): 400 ~412	Acknowledged

Description

The following code calculates the number of shares (`actualShares`) to be issued for a given amount of assets deposited:

```
406    uint256 actualShares = MathUpgradeable.mulDiv(
407        actualAssetsValue,
408        totalSupply() + 10 ** _decimalsOffset(),
409        totalAssets() + 1 - actualAssetsValue,
// remove the new assets deposited to the VToken in this operation
410        MathUpgradeable.Rounding.Down
411    );
```

The given code snippet depicts a mathematical calculation that is unnecessarily complex and can be highly simplified. The current calculation of `actualAssetsValue` and `totalAssets` involves the calculation of `vToken.exchangeRateStored()` for two times. However, this formula is **unnecessarily complex** and involves redundant calculations, particularly multiple invocations of `vToken.exchangeRateStored()` . Upon simplification, we observe:

$$\text{actualShares} = \frac{\text{actualAssetsValue} \cdot (\text{totalSupply} + 10^{\text{decimalsOffset}})}{\text{totalAssets} + 1 - \text{actualAssetsValue}}$$

Given that:

- `actualAssetsValue = vTokensReceived * R'`
- `totalAssets = vTokenBalanceAfter * R' = (vTokenBalanceBefore + vTokensReceived) * R'`

We can factor out the exchange rate `R'` and cancel it, leading to a simplified and equivalent expression:

$$\text{actualShares} = \frac{vTokensReceived \cdot (\text{totalSupply} + 10^{\text{decimalsOffset}})}{\text{vTokenBalanceBefore} + 1}$$

Benefits of the Simplified Formula

1. **No Redundant Exchange Rate Calculations**: Eliminates the need to call `vToken.exchangeRateStored()` multiple times, which saves gas and simplifies reasoning. Note that the Rate has changed after the `mintVTokens()` call.

2. **Improved Precision**: Avoids potential rounding inconsistencies due to repeated multiplications and divisions involving the exchange rate.
3. **Clarifies the `+1` Offset**: The `+1` in the denominator now clearly serves as a safety margin (e.g., to prevent division by zero), and is easier to justify in the simplified form.

Recommendation

Use the simplified formula instead:

```
uint256 actualShares = MathUpgradeable.mulDiv(
    vTokensReceived,
    totalSupply() + 10 ** _decimalsOffset(),
    vTokenBalanceBefore + 1,
    MathUpgradeable.Rounding.Down
);
```

Alleviation

[Venus, 05/13/2025]: Issue acknowledged. I won't make any changes for the current version.

APPENDIX | VENUS - ERC4626 WRAPPER

Finding Categories

Categories	Description
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Incorrect Calculation	Incorrect Calculation findings are about issues in numeric computation such as rounding errors, overflows, out-of-bounds and any computation that is not intended.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Entire Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.