

Venus Liquidator

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Liquidation	Documentation quality	Medium	<div><div></div></div>
Timeline	2023-07-12 through 2023-07-17	Test quality	Medium	<div><div></div></div>
Language	Solidity	Total Findings	7	<div><div></div><div>Fixed: 2</div><div>Acknowledged: 5</div></div>
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review	High severity findings ⓘ	0	
Specification	None	Medium severity findings ⓘ	0	
Source Code	<ul style="list-style-type: none">VenusProtocol/venus-protocol #457c8cd 	Low severity findings ⓘ	5	<div><div></div><div>Fixed: 1</div><div>Acknowledged: 4</div></div>
Auditors	<ul style="list-style-type: none">Julio Aguliar Auditing EngineerNikita Belenkov Auditing EngineerPavel Shabarkin Auditing EngineerMustafa Hasan Senior Auditing Engineer	Undetermined severity findings ⓘ	0	
		Informational findings ⓘ	2	<div><div></div><div>Fixed: 1</div><div>Acknowledged: 1</div></div>

Summary of Findings

The Venus Protocol is a decentralized, algorithmic-based lending and credit system on the Binance Smart Chain. It enables users to utilize their cryptocurrencies by supplying collateral to the protocol that may be borrowed by pledging over-collateralized cryptocurrencies.

The Venus protocol operates on a granular architecture, where the Liquidator smart contract acts as an interface to initiate the liquidation process of borrowed positions. The core mechanisms of lending/borrowing such as health factor checks, evaluation of borrow positions, and more are implemented within the Comptroller contract, which is out-of-scope for this security audit and may have its own vulnerabilities.

The security audit revealed several low- and info-severity issues related to input validation and protocol-specific business logic. Overall, the code of the Liquidator contract is well written following best industry coding practices. We recommend fixing the identified security issues and improving the test suite to increase the code coverage and the SuMo score.

Fix Review Update

The majority of the issues were acknowledged by the Venus team since they want to address them in a future update.

ID	DESCRIPTION	SEVERITY	STATUS
VEN-1	The Protocol Might Get More Collateral than Desired	<ul style="list-style-type: none">Low ⓘ	Acknowledged
VEN-2	Missing Input Validation	<ul style="list-style-type: none">Low ⓘ	Fixed
VEN-3	Using Vulnerable Solidity Version	<ul style="list-style-type: none">Low ⓘ	Acknowledged
VEN-4	Reserve Reduction Logic May Not Be Executed	<ul style="list-style-type: none">Low ⓘ	Acknowledged
VEN-5	Uninitialized <code>forceVAILiquidate</code> Allows Bypassing the <code>VAI</code> Debt Validation	<ul style="list-style-type: none">Low ⓘ	Acknowledged
VEN-6	Paused Liquidation of the <code>VAI</code> Token May Be Bypassed	<ul style="list-style-type: none">Informational ⓘ	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
VEN-7	Ownership Can Be Renounced	• Informational ⓘ	Fixed

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

*i***Disclaimer**

Only features that are specified in the scope and at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. The smart contract in question in this audit interacts with other smart contracts from the Venus team that are NOT part of the scope but perform very important and relevant tasks for the correct functioning of the Liquidator. Additionally, all features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
 1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
 2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
 1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

As mentioned before the focus of this audit is the Liquidator smart contract, which interacts with other smart contracts from the Venus team that perform important tasks necessary for its correct functioning. However, the `Comptroller` , `VAIController` , and `VToken` are out of scope and may have vulnerabilities of their own.

Files Included

- contracts/Liquidator/Interfaces.sol
- contracts/Liquidator/Liquidator.sol
- contracts/Liquidator/LiquidatorStorage.sol

Files Excluded

- contracts/Comptroller/*
- contracts/DelegateBorrowers/*
- contracts/Governance/*
- contracts/InterestRateModels/*
- contracts/Lens/*
- contracts/Oracle/*
- contracts/Swap/*

- contracts/Tokens/*
- contracts/Utils/*
- contracts/Vault/*
- contracts/VRTVault/*
- contracts/XVSVault/*

Findings

VEN-1 The Protocol Might Get More Collateral than Desired

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

We could add the extra check in `Comptroller._setLiquidationIncentive`, but we prefer to not add it to reduce the number of changes, and avoid an upgrade of the `Comptroller` contract. The `liquidationIncentiveMantissa` attribute can be modified only via VIP, with a delayed period when anyone can review the new value and warn about incoherences. So, the described issue should be identified then.

File(s) affected: `Liquidator.sol`

Description: The `comptroller.liquidationIncentiveMantissa()` is used to determine the amount of collateral to be seized based on the amount of debt that the liquidator wants to pay, and to limit the maximum percentage value that the `treasuryPercentMantissa` can have.

If the `liquidationIncentiveMantissa` gets reduced under the value of the `treasuryPercentMantissa` without updating the latter, it would lead to the protocol getting more collateral than what should be.

Recommendation: Make sure these variables are updated in the correct order and in the same transaction.

VEN-2 Missing Input Validation

• Low ⓘ Fixed

Update

Marked as "Fixed" by the client. Addressed in: `5f39ba0ee58a3c430075044c0f1ca919bb5759d5`, `b2267aacdd60e9b5e6f8823baaed64c7029e9846`. The client provided the following explanation:

Non Zero Address check already present.

Related Issue(s): [SWC-123](#)

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The function `setPendingRedeemChunkLength()` should check that `newLength_` is not 0.

Recommendation: We recommend adding the relevant check.

VEN-3 Using Vulnerable Solidity Version

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

We are using `AccessControlledV8`, from the `@venusprotocol/governance-contracts` package, that is using solidity 0.8.13. We will upgrade to solidity 0.8.20 in a coordinated way in the future.

File(s) affected: `Interfaces.sol`, `Liquidator.sol`, `LiquidatorStorage.sol`

Description: As security standards develop, so does the Solidity language. In order to stay up to date with current practices, it's important to use a recent version of Solidity and recent conventions. The version used in all the contracts in scope, `0.8.13`, is known to have vulnerabilities, for more information [see here](#).

Recommendation: Consider using Solidity version `0.8.18` instead and refer to the list of [recommended versions](#) for up-to-date suggestions.

VEN-4 Reserve Reduction Logic May Not Be Executed

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. It was added as an improvement proposal in:
<https://github.com/VenusProtocol/vips/commit/91e25d3f1ab38bd75b0f2946f34c82dee32570d3>.

File(s) affected: `Liquidator.sol`

Description: The `_reduceReservesInternal()` function may not carry out its logic in a liquidation event in case `pendingRedeemChunkLength` remains unset (defaulted to zero).

It does not matter how many items are in `pendingRedeem`, as long as `pendingRedeemChunkLength` remains unset, the `range` and, therefore, the `index` variables will be set to zero and the loop will be skipped.

Recommendation: We recommend setting `pendingRedeemChunkLength` at construction time or calling `setPendingRedeemChunkLength()` in the deployment script.

VEN-5

Uninitialized `forceVAILiquidate` Allows Bypassing the `VAI` Debt Validation

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. It was added as an improvement proposal in:
<https://github.com/VenusProtocol/vips/commit/91e25d3f1ab38bd75b0f2946f34c82dee32570d3>.

File(s) affected: `Liquidator.sol`

Description: The `forceVAILiquidate` variable enforces the protocol to check the debt amount of `VAI` token. If it is greater than the minimum threshold (`minLiquidatableVAI`), then it should revert the liquidation procedure unless the liquidated token is the `VAI` token. Additionally, the documentation states that the protocol will enforce the liquidation of `VAI` first over other tokens. However, the `forceVAILiquidate` variable is not set during the initialization step and will default to a paused state, which would essentially bypass the need to validate the `VAI` debt.

Recommendation: If this is not the desired behavior, make sure to initialize the `forceVAILiquidate` variable during the contract deployment.

VEN-6

Paused Liquidation of the `VAI` Token May Be Bypassed

• Informational ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

We wouldn't change anything here, to avoid duplication of checks.

File(s) affected: `Liquidator.sol`

Description: The function `_checkForceVAILiquidate()` initiates by verifying whether the `VAI` liquidation is currently paused. If a pause is detected, it bypasses the subsequent checks and continues with the rest of the execution. However, when the `vToken` is identified as the `VAI` token, the `liquidateBorrow()` function neglects to confirm the pausing status of the `vaiController` prior to attempting the liquidation. This oversight may potentially cause a revert within the `vaiController` contract, as confirmed by the Venus team. However, the `Liquidator` contract has the capability to manage this situation before invoking the `vaiController` contract.

Recommendation: To bolster the robustness of the system and more clearly outline the business logic, we recommend conducting a check of `_isVAILiquidationPaused` in the `liquidateBorrow()` function within the `Liquidator` contract prior to initiating the external call.

VEN-7 Ownership Can Be Renounced

• Informational ⓘ Fixed

Update

Marked as "Fixed" by the client. Addressed in: `f99196754ceea6e28ac69be3ad55cdce8d261339`. The client provided the following explanation:

Empty implementation of `renounceOwnership` function provided.

Description: The `Liquidator` contract inherits `Ownable2StepUpgradeable`, which allows having an owner of the contract. It also features a function allowing to renounce owner role via `renounceOwnership()`. If the owner renounces their ownership, any function guarded by the

`onlyOwner` modifier will no longer be able to be executed.

Recommendation: Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Adherence to Best Practices

1. **Fixed** `ProtocolLiquidationIncentiveTransferred` event casts `wBNB` to `address(wBNB)`, which is unnecessary as `wBNB` is already of type `address`.
2. **Acknowledged** `vBnb` is stored as an interface, while `wBNB` is stored as an address. It is not a good practice to use both ways in the same contract. Pick one and use it consistently.
3. **Fixed** The `Interfaces` contract is missing the SPDX license identifier.
4. For readability and maintainability avoid using magic numbers. Instead, assign them to a constant variable.
 1. **Fixed** The function `validateTreasuryPercentMantissa()` uses `1e18`.

Adherence to Specification

1. **Fixed** According to the documentation, the integration of the `AccessControlManager` is to replace the check of the owner in every function. However, the function `setProtocolShareReserve()` still has the `onlyOwner` modifier. Make sure to either update the documentation or add the relevant check.
2. **Fixed** According to the documentation, the protocol would like to force users to liquidate `VAI` positions first. However, there is a mechanism to activate and deactivate this feature. Make sure to document this mechanism as well in the documentation or remove it from the implementation if necessary.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

- `021...b82 ./Liquidator/Liquidator.sol`
- `cd3...608 ./Liquidator/LiquidatorStorage.sol`
- `530...6fa ./Liquidator/Interfaces.sol`

Tests



- `13a...259 ./Liquidator/liquidatorHarnessTest.ts`
- `d74...6d8 ./Liquidator/restrictedLiquidations.ts`
- `1fe...e9b ./Liquidator/liquidatorTest.ts`

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- [Slither](#)  v0.8.3
- [SuMo](#)  bb1c918

Steps taken to run the tools:

Slither

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

SuMo

1. `npm install @morenabarboni/sumo`
2. `npx/yarn sumo lookup`
3. `npx/yarn sumo mutate`

Automated Analysis

Slither

274 results have been reported by the tool, and relevant issues have been included in the report.

SuMo

Mutation testing is a test adequacy assessment technique - it aims to evaluate and improve the quality of a given test suite:

- It purposefully injects artificial faults into the Code Under Test
- Each faulty version - MUTANT - of the original program contains a single fault
- It evaluates the ability of the test suite to detect these faults.

Results:

- Generated mutants: 450
- Tested mutants: 443
- Live mutants: 195
- Killed mutants: 248
- Equivalent mutants: 0
- Redundant mutants: 0
- Stillborn mutants: 7
- Timed-Out mutants: 0

MUTATION SCORE = 55.98

The higher the score the better. The live mutants represent parts of the code that were not tested well: i.e. negative tests are missing.

We recommend adding tests that cover all possible scenarios.

Test Suite Results

Liquidator

`splitLiquidationIncentive`

- ✓ splits liquidationIncentive between Treasury and Liquidator with correct amounts

`distributeLiquidationIncentive`

- ✓ distributes the liquidationIncentive between Treasury and Liquidator with correct amounts (57ms)
- ✓ reverts if transfer to liquidator fails
- ✓ reverts if underlying transfer to protocol share reserves fails (47ms)

Liquidator

`liquidateBorrow`

`liquidating BEP-20 debt`

- ✓ fails if borrower is zero address
 - ✓ fails if some BNB is sent along with the transaction (42ms)
 - ✓ transfers the seized collateral to liquidator and protocolShareReserve (132ms)
 - ✓ transfers tokens from the liquidator (158ms)
 - ✓ approves the borrowed VToken to spend underlying (126ms)
 - ✓ calls liquidateBorrow on borrowed VToken (129ms)
 - ✓ emits LiquidateBorrowedTokens event (134ms)
- `liquidating VAI debt`

```

    ✓ transfers VAI from the liquidator (135ms)
    ✓ approves VAIController to spend VAI (117ms)
    ✓ calls liquidateVAI on VAIController (115ms)
liquidating BNB debt
    ✓ fails if msg.value is not equal to repayment amount (76ms)
    ✓ transfers BNB from the liquidator (94ms)
    ✓ calls liquidateBorrow on VBNB (86ms)
    ✓ forwards BNB to VBNB contract (89ms)
setTreasuryPercent
    ✓ updates treasury percent in storage (44ms)
    ✓ fails when permission is not granted
    ✓ fails when the percentage is too high
    ✓ uses the new treasury percent during distributions (151ms)
Force VAI Liquidation
    ✓ Should able to liquidate any token when VAI debt is lower than minLiquidatableVAI (79ms)
    ✓ Should not able to liquidate any token when VAI debt is greater than minLiquidatableVAI
    ✓ Should able to liquidate VAI token when VAI debt is greater than minLiquidatableVAI
    ✓ Should able to liquidate any token and VAI token when force Liquidation is off


Liquidator
  Restricted liquidations
    addToAllowlist
      ✓ fails if not allowed to call
      ✓ adds address to allowlist (41ms)
      ✓ fails if already in the allowlist (41ms)
      ✓ emits LiquidationPermissionGranted event
    removeFromAllowlist
      ✓ fails if not allowed to call
      ✓ fails if not in the allowlist
      ✓ removes address from allowlist (67ms)
      ✓ emits LiquidationPermissionRevoked event (47ms)
    restrictLiquidation
      ✓ fails if not allowed to call
      ✓ restricts liquidations for the borrower
      ✓ fails if already restricted (58ms)
      ✓ emits LiquidationRestricted event
    unrestrictLiquidation
      ✓ fails if not allowed to call
      ✓ removes the restrictions for the borrower (67ms)
      ✓ fails if not restricted
      ✓ emits LiquidationRestricted event (45ms)
  liquidateBorrow
    ✓ fails if the liquidation is restricted (44ms)
    ✓ proceeds with the liquidation if the guy is allowed to (63ms)
```

Code Coverage

We recommend adding more tests to improve the code coverage since it currently ranges between 68% and 84%, and we normally advise at least 90%.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Liquidator/	78.42	68	83.87	76.43	
Interfaces.sol	100	100	100	100	
Liquidator.sol	78.42	68	83.87	76.43	... 514,515,516
LiquidatorStorage.sol	100	100	100	100	

Fix Review Update

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Liquidator/	77.86	65.38	81.25	75.95	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Interfaces.sol	100	100	100	100	
Liquidator.sol	77.86	65.38	81.25	75.95	... 518,519,520
LiquidatorStorage.sol	100	100	100	100	

Changelog

- 2023-07-18 - Initial report
- 2023-07-26 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp’s mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp’s team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over \$200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp’s collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a

limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

