Quantstamp

Venus Protocol: Native Token Gateway

Executive Summary

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

Type	DeFi (Lending Market)
Timeline	2024-02-28 through 2024-03-01
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md (venus-protocol) README.md (isolated-pools) Scope Doc
Source Code	• VenusProtocol/isolated-pools #9fe398b • VenusProtocol/venus-protocol #dbd4edc
Auditors	• Hytham Farah Auditing Engineer • Julio Aguilar Auditing Engineer • Jennifer Wu Auditing Engineer

Documentation quality	Medium
Test quality	High
Total Findings	5 Fixed: 1 Acknowledged: 4
High severity findings	0
Medium severity findings	0
Low severity findings	1 Acknowledged: 1
Undetermined severity findings	0
Informational findings	4 Fixed: 1 Acknowledged: 3

Summary of Findings

The main purpose of this audit is to verify two PRs, VEN-2375 and VEN-2356, adding a gateway allowing users to interact with the venus protocol directly with native currency. Currently, users must wrap native currency before interacting with the protocol.

The audit involved two seperate repos, the more recent, `isolated-pools` repo and the legacy `venus-protocol` repo contain the initial version of the protocol and which is still being maintained due to its TVL. In both cases, the features added were to enable direct interaction with the native currency.

In the `venus-protocol` repo, the main change observed is the separation of the redeemer and a reciever in many of the functions associated with the VToken contracts. This allows a redeemer to approve a receiver as a valid delegate who may then call functions that execute the operation of redeeming vTokens back to the original underlying token.

The `isolated-pools` repository featured more extensive changes. Along with the separation outlined above, the new `NativeTokenGateway` contract, which handles the wrapping and unwrapping of native currencies directly for users.

No major issues were found and only a single low-severity issue when about centralized access to user funds accidentally sent to the contract.

Update: All issues in the report were either acknowledged or fixed by the clients.

ID	DESCRIPTION	SEVERITY	STATUS
VEN-GATE-1	Centralization Risk	Low	Acknowledged
VEN-GATE-2	Already Approved Borrow Delegates Can Now Redeem as Well	Informational	Acknowledged
VEN-GATE-3	Different Solidity Versions Used in the Same Codebase	Informational	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
VEN-GATE-4	Unlocked Pragma	Informational	Acknowledged
VEN-GATE-5	Event Emission for Unchanged Delegate Status	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.

Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. 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

The scope consisted of the contracts changed in the PRs VEN-2375 for the `venus-protocol` repo and VEN-2356 for the `isolated-pools` repo.

Files Included

- Repo 1 - Isolated Pools
    - contracts/Comptroller.sol
    - contracts/ComptrollerStorage.sol
    - contracts/Gateway/Interfaces/IVtoken.sol
    - contracts/Gateway/Interfaces/IWrappedNative.sol
    - contracts/Gateway/NativeTokenGateway.sol
    - contracts/VToken.sol
    - contracts/VTokenInterfaces.sol
- Repo 2 - Venus Protocol:
    - contracts/Tokens/VTokens/VBep20.sol
    - contracts/Tokens/VTokens/VToken.sol

- contracts/Comptroller/Diamond/facets/MarketFacet.sol

Files Excluded

All files in the repository not explicitly mentioned above.

Findings

VEN-GATE-1 Centralization Risk

Low Acknowledged

Update

The client acknowledged the issue providing the following explanation:

> The owner of the contract will be the Normal Timelock contract (https://bscscan.com/address/0×939bD8d64c0A9583A7Dcea9933f7b21697ab6396 on BNB chain, for example) used by Governance to execute normal VIP's (Venus Improvement Proposals). So, only the Venus community with a vote will be able to execute the privilege functions in the NativeTokenGateway contract

File(s) affected: `NativeTokenGateway`

Description: The `NativeTokenGateway` allows the owner to transfer all native and ERC20 tokens in the contract to themselves through the functions `sweepNative()` and `sweepToken()`. However, it is worth noting that this is only related to stuck native and ERC20s tokens since users should only send native tokens and receive its wrapped version through specific functions defined by the contract.

Recommendation: We recommend adding this to user-facing documentation. Additionally, in order to reduce the centralization risk, the contract could add the constraint to the `receive()` and `fallback()` functions to allow only the wrapped native token contract to send native tokens. This could then allow the team to remove the `sweepNative()` function altogether.

VEN-GATE-2
Already Approved Borrow Delegates Can Now Redeem as Well

Informational Acknowledged

Update

The client acknowledged the issue providing the following explanation:

> Mentioned in the natspec that now the user will be able to redeem as well on behalf of the approver. Moreover, only one wallet has approved borrow on behalf until now in the Core pool, and it doesn't have any activity. Finally, this info will be shared in the official Venus channels and added to the VIP description.

File(s) affected: `VToken (venus-protocol)`

Description: The protocol already allows users the ability to approve delegates to borrow and pay on their behalf. The protocol's upgrade extends the functionalities available to delegates, now includes the ability to redeem underlying collateral on behalf of the vToken holder. This extension is built upon the pre-existing delegate feature, which already permitted delegates to mint, repay, or borrow.

The issue is that users approved those borrow delegates under the assumption that they would only be able to borrow on their behalf. However, with this new update, those same borrow delegates are automatically redeem delegates as well.

Recommendation: Ensure that all existing vToken holders are informed about the new delegate permission to redeem collateral. It is crucial to communicate this change effectively, allowing vToken holders to reassess their delegate relationships. Those who do not wish to extend this additional permission to their delegates should consider revoking delegation before the upgrade takes effect.

VEN-GATE-3
Different Solidity Versions Used in the Same Codebase

Informational Acknowledged

Update

The client acknowledged the issue providing the following explanation:

> We have developed a completely new functionality which will be working independently so, using close to the latest solidity version.

File(s) affected: `NativeTokenGateway`

Description: The isolated pools repository uses Solidity version `0.8.13` in most of the contracts in scope except for the `NativeTokenGateway` contract which uses `0.8.20`. Using different compiler versions brings risks of inconsistent contract behavior. Additionally, some chains do not support the latter version, and deployment might not be possible, or the contracts usage might be hindered by this.

Recommendation: We recommend using only one version of Solidity throughout the whole codebase. Additionally, if version `0.8.20` is used, make sure to set the `evmVersion` parameter in the hardhat configuration file to `paris` for those chains that still do not support it.

VEN-GATE-4 Unlocked Pragma

• Informational Acknowledged

Update

The client acknowledged the issue providing the following explanation:

> Carrying on with the same practice as before.

File(s) affected: `VBep20`, `VToken (venus-protocol)`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, we recommend to remove the caret to lock the file onto a specific Solidity version.

VEN-GATE-5 Event Emission for Unchanged Delegate Status

• Informational Fixed

Update

This fix was implemented for both repositories. The `updateDelegate()` function now reverts if the delegate status of the provided address remains unchanged.

Update

The client fixed the issue. Addressed in: `f8c58046a47d28c1b599e8aeec1096e42a699b8d`, `0d4864063f2cf629220d58739e8b81a53a733731`.

File(s) affected: `Comptroller`, `MarketFacet`

Description: The `updateDelegate` function emits a `DelegateUpdated` event for every call, even if the delegate's approval status remains unchanged.

Recommendation: Modify the `updateDelegate` function to include a condition that checks for an actual change in the delegate's approval status before emitting the `DelegateUpdated` event and updating the contract's state.

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).

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

- `e76...525 ./isolated-pools/contracts/Comptroller.so`
- `d29...3b7 ./isolated-pools/contracts/ComptrollerStorage.sol`
- `4dd...e39 ./isolated-pools/contracts/VTokenInterfaces.sol`
- `615...023 ./venus-protocol/contracts/Comptroller/Diamond/facets/MarketFacet.sol`
- `e36...196 ./venus-protocol/contracts/Tokens/VTokens/VToken.sol`
- `a39...9e3 ./isolated-pools/contracts/ComptrollerStorage.sol`
- `dcd...303 ./venus-protocol/contracts/Tokens/VTokens/VBep20.sol`
- `b4f...df1 ./isolated-pools/contracts/Gateway/NativeTokenGateway.sol`
- `357...376 ./isolated-pools/contracts/Gateway/NativeTokenGateway.sol`
- `33e...66e ./isolated-pools/contracts/Gateway/Interfaces/IVToken.sol`
- `3e0...2b1 ./isolated-pools/contracts/Gateway/Interfaces/IWrappedNative.sol`

Tests

- `7d4...dc5 ./tests/tests/VRTConverterProxyTest.js`
- `9c3...174 ./tests/tests/TimelockTest.js`
- `b04...2a4 ./tests/tests/Jest.js`
- `cbe...4a8 ./tests/tests/VRTConverterTest.js`
- `2d3...279 ./tests/tests/CompilerTest.js`
- `36b...130 ./tests/tests/SpinaramaTest.js`
- `081...538 ./tests/tests/XVSVestingTest.js`
- `657...265 ./tests/tests/Matchers.js`
- `bc7...533 ./tests/tests/Errors.js`
- `dc7...5da ./tests/tests/MaximillionTest.js`
- `478...f6d ./tests/tests/XVSVestingProxyTest.js`
- `cea...474 ./tests/tests/gasProfiler.js`
- `7b2...24b ./tests/tests/VenusChainlinkOracleTest.js`
- `ef8...2b3 ./tests/tests/Scenario.js`
- `670...51c ./tests/tests/treasuryTest.js`
- `724...fe6 ./tests/tests/Tokens/setComptrollerTest.js`
- `a2b...97d ./tests/tests/Tokens/adminTest.js`
- `9a2...c1a ./tests/tests/Tokens/accrueInterestTest.js`
- `736...62b ./tests/tests/Tokens/borrowAndRepayVBNBTest.js`
- `cb7...7d4 ./tests/tests/Tokens/xvsLikeTest.js`
- `7f3...4c7 ./tests/tests/Tokens/setInterestRateModelTest.js`
- `00b...ffa ./tests/tests/Tokens/borrowAndRepayTest.js`
- `f0c...8da ./tests/tests/Tokens/vTokenTest.js`
- `34d...eed ./tests/tests/Tokens/mintAndRedeemFeeTest.js`
- `c44...639 ./tests/tests/Tokens/liquidateTest.js`
- `b6f...126 ./tests/tests/Tokens/safeTokenTest.js`
- `fb2...bf8 ./tests/tests/Tokens/mintAndRedeemTest.js`
- `b10...245 ./tests/tests/Tokens/transferTest.js`
- `bd0...d20 ./tests/tests/Tokens/reservesTest.js`

- 09a...cc4 ./tests/tests/Tokens/mintAndRedeemVBNBTest.js
- 05e...0e5 ./tests/tests/Tokens/mintAndRedeemVBNBFeeTest.js
- db0...aaf ./tests/tests/Fuzz/VenusWheelFuzzTest.js
- 79f...098 ./tests/tests/Comptroller/protocolPauseTest.js
- 785...c57 ./tests/tests/Comptroller/accountLiquidityTest.js
- 883...2fa ./tests/tests/Comptroller/releaseToVaultTest.js
- 195...dd1 ./tests/tests/Comptroller/proxiedComptrollerTest.js
- 28e...569 ./tests/tests/hardhat/EvilXToken.ts
- 0dc...fa4 ./tests/tests/hardhat/WhitePaperInterestRateModel.ts
- 423...510 ./tests/tests/hardhat/MaxLoopsLimitHelper.ts
- 911...487 ./tests/tests/hardhat/Rewards.ts
- e19...c6b ./tests/tests/hardhat/JumpRateModelV2.ts
- fef...55c ./tests/tests/hardhat/PoolRegistry.ts
- 1dd...509 ./tests/tests/hardhat/Shortfall.ts
- 46b...932 ./tests/tests/hardhat/UpgradedVToken.ts
- 04c...fd7 ./tests/tests/hardhat/Prime.ts
- d4e...ae1 ./tests/tests/hardhat/AccessControl.ts
- 3f0...81b ./tests/tests/hardhat/Gateway/NativeTokenGateway.ts
- baa...eef ./tests/tests/hardhat/Unitroller/unitrollerTest.ts
- 896...d73 ./tests/tests/hardhat/Unitroller/adminTest.ts
- 472...cd2 ./tests/tests/hardhat/Prime/PrimeLiquidityProvider.ts
- ecd...545 ./tests/tests/hardhat/Prime/Prime.ts
- c8b...24c ./tests/tests/hardhat/Liquidator/liquidatorHarnessTest.ts
- 66d...fc8 ./tests/tests/hardhat/Liquidator/restrictedLiquidations.ts
- 64f...d61 ./tests/tests/hardhat/Liquidator/liquidatorTest.ts
- b7e...843 ./tests/tests/hardhat/Tokens/mintAndRedeemTest.ts
- 071...f8d ./tests/tests/hardhat/Tokens/liquidateTest.ts
- 77e...5be ./tests/tests/hardhat/Tokens/transferTest.ts
- 474...2c1 ./tests/tests/hardhat/Tokens/accrueInterestTest.ts
- ead...1b2 ./tests/tests/hardhat/Tokens/setters.ts
- 20b...196 ./tests/tests/hardhat/Tokens/borrowAndRepayTest.ts
- 371...ce7 ./tests/tests/hardhat/XVS/XVSVaultFix.ts
- 591...bb4 ./tests/tests/hardhat/XVS/XVSVault.ts
- ac5...62d ./tests/tests/hardhat/DelegateBorrowers/SwapDebtDelegate.ts
- 8ef...459 ./tests/tests/hardhat/DelegateBorrowers/MoveDebtDelegate.ts
- b68...082 ./tests/tests/hardhat/fixtures/ComptrollerWithMarkets.ts
- 108...bb1 ./tests/tests/hardhat/lib/TokenDebtTracker.ts
- bf6...cc4 ./tests/tests/hardhat/lib/ApproveOrRevert.ts
- 795...7cd ./tests/tests/hardhat/Comptroller/pauseTest.ts
- b7b...256 ./tests/tests/hardhat/Comptroller/liquidateAccountTest.ts
- 4ad...c19 ./tests/tests/hardhat/Comptroller/assetsListTest.ts
- 6f4...120 ./tests/tests/hardhat/Comptroller/liquidateCalculateAmountSeizeTest.ts
- 347...169 ./tests/tests/hardhat/Comptroller/healAccountTest.ts
- 3ea...bc7 ./tests/tests/hardhat/Comptroller/setters.ts
- e6e...cbe ./tests/tests/hardhat/Comptroller/hooks.ts
- 08c...41b ./tests/tests/hardhat/Comptroller/accountLiquidityTest.ts
- 263...24d ./tests/tests/hardhat/Comptroller/Diamond/liquidateCalculateAmoutSeizeTest.ts
- b68...4a9 ./tests/tests/hardhat/Comptroller/Diamond/pauseTest.ts
- 622...a2f ./tests/tests/hardhat/Comptroller/Diamond/XVSSpeeds.ts
- e9b...ed9 ./tests/tests/hardhat/Comptroller/Diamond/assetListTest.ts
- f25...232 ./tests/tests/hardhat/Comptroller/Diamond/diamond.ts
- fc8...533 ./tests/tests/hardhat/Comptroller/Diamond/comptrollerTest.ts
- b50...a78 ./tests/tests/hardhat/Comptroller/Diamond/accessControl.ts
- 919...32c ./tests/tests/hardhat/Comptroller/Diamond/scripts/deploy.ts
- d9f...3ed ./tests/tests/hardhat/VRT/VRTVault.ts

- a6f...87a ./tests/tests/hardhat/Lens/RewardsSummary.ts
- 42b...320 ./tests/tests/hardhat/Lens/Rewards.ts
- 0ed...c4e ./tests/tests/hardhat/Lens/PoolLens.ts
- bc4...5d9 ./tests/tests/hardhat/VAI/VAIController.ts
- cac...dd5 ./tests/tests/hardhat/VAI/VAIVault.ts
- ba3...e7e ./tests/tests/hardhat/VAI/PegStability.ts
- a62...a90 ./tests/tests/hardhat/Admin/VBNBAdmin.ts
- 126...0ec ./tests/tests/hardhat/Swap/swapTest.ts
- ee8...111 ./tests/tests/hardhat/integration/index.ts
- cd3...7c7 ./tests/tests/hardhat/util/AddressOrContract.ts
- 716...ec2 ./tests/tests/hardhat/util/Proposals.ts
- aec...cc7 ./tests/tests/hardhat/util/Errors.ts
- 95f...f9d ./tests/tests/hardhat/util/types.ts
- e3b...855 ./tests/tests/hardhat/util/ComptrollerTestHelpers.ts
- 754...e67 ./tests/tests/hardhat/util/TokenTestHelpers.ts
- f41...b1a ./tests/tests/hardhat/Fork/reduceResevesTest.ts
- e80...821 ./tests/tests/hardhat/Fork/vrtStopRewards.ts
- 716...ccc ./tests/tests/hardhat/Fork/VTokensUpgrade.ts
- 9f0...adc ./tests/tests/hardhat/Fork/liquidation.ts
- e1d...32f ./tests/tests/hardhat/Fork/VRTVaultUpgrade.ts
- 6ab...f21 ./tests/tests/hardhat/Fork/vBNBAdmin.ts
- 71d...3f4 ./tests/tests/hardhat/Fork/NativeTokenGateway.ts
- 533...2fc ./tests/tests/hardhat/Fork/BUSDLiquidator.ts
- ba8...122 ./tests/tests/hardhat/Fork/ForceVAIDebtFirstTest.ts
- 3ed...a5f ./tests/tests/hardhat/Fork/liquidatorForkTestsTestnet.ts
- 095...421 ./tests/tests/hardhat/Fork/utils.ts
- c05...50e ./tests/tests/hardhat/Fork/vTokenRedeemUpgrade.ts
- 4fc...df9 ./tests/tests/hardhat/Fork/supply.ts
- b14...c8c ./tests/tests/hardhat/Fork/Shortfall.ts
- 033...848 ./tests/tests/hardhat/Fork/pegStabilityTest.ts
- 9a0...35c ./tests/tests/hardhat/Fork/diamondTest.ts
- 17d...6da ./tests/tests/hardhat/Fork/XVSVaultUpgrade.ts
- 3a6...2ed ./tests/tests/hardhat/Fork/vTokenUpgradeHelper.ts
- 623...73d ./tests/tests/hardhat/Fork/borrowAndRepayTest.ts
- 63c...aa0 ./tests/tests/hardhat/Fork/reduceReservesTest.ts
- bd8...600 ./tests/tests/hardhat/Fork/RewardsForkTest.ts
- 359...623 ./tests/tests/hardhat/Fork/VAIVaultUpgrade.ts
- 312...0b3 ./tests/tests/hardhat/Fork/swapTest.ts
- 5fd...115 ./tests/tests/hardhat/Fork/RiskFund.ts
- 965...ec4 ./tests/tests/hardhat/Fork/vTokenACMUpdates.ts
- 886...d78 ./tests/tests/hardhat/Fork/TokenRedeemer.ts
- 1f4...621 ./tests/tests/hardhat/Fork/RiskFundSwap.ts
- a61...33e ./tests/tests/Flywheel/FlywheelTest.js
- 6e5...83f ./tests/tests/Flywheel/GasTest.js
- 7b6...a80 ./tests/tests/Lens/SnapshotLensTest.js
- 833...0c1 ./tests/tests/Lens/VenusLensTest.js
- a97...dcc ./tests/tests/Lens/InterestRateModelLensTest.js
- 95d...542 ./tests/tests/XVSVault/xvsVaultTest.js
- b09...1ca ./tests/tests/XVSVault/xvsVaultProxyTest.js
- 3c9...7ed ./tests/tests/Models/InterestRateModelTest.js
- 2b3...1db ./tests/tests/Models/DAIInterestRateModelTest.js
- bcf...e6a ./tests/tests/Utils/JS.js
- 6c4...dd1 ./tests/tests/Utils/Venus.js
- 266...19d ./tests/tests/Utils/BSC.js
- 7d5...485 ./tests/tests/Utils/EIP712.js

- `028...91c ./tests/tests/VAI/liquidateVAITest.js`
- `138...e5d ./tests/tests/VAI/liquidateVAICalculateAmountSeizeTest.js`
- `934...a91 ./tests/tests/integration/index.ts`
- `e02...286 ./tests/tests/VRTVault/vrtVaultProxyTest.js`
- `12a...a20 ./tests/tests/VRTVault/vrtVaultTest.js`

Toolset

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

Setup

Tool Setup:

- Slither v0.10.1

Steps taken to run the tools:

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

Automated Analysis

Slither

All results were either included in the report or dismissed as false positives.

Test Suite Results

Note that the full output of the test suite was pruned to include only the relevant files in scope. In general, the test suite is extensive and includes many thorough test cases, particularly, in the `isolated-pools` repository.

```
ISOLATED POOLS:

   Comptroller
    liquidateCalculateAmountSeize
      ✔ fails if borrowed asset price is 0
      ✔ fails if collateral asset price is 0
      ✔ fails if the repayAmount causes overflow  (62ms)
      ✔ fails if the borrowed asset price causes overflow
      ✔ reverts if it fails to calculate the exchange rate
      ✔ returns the correct value for
100000000000000000,100000000000000000,100000000000000000,100000000000000000,100000000000000000
(880ms)
      ✔ returns the correct value for
200000000000000000,100000000000000000,100000000000000000,100000000000000000,100000000000000000
(844ms)
      ✔ returns the correct value for
200000000000000000,200000000000000000,142000000000000000,130000000000000000,245000000000000000
(88ms)
      ✔ returns the correct value for
278900000000000000,523048084200000000,771320000000000000000,130000000000000000,1.000245e+22 (53ms)
      ✔ returns the correct value for
7.009232529961056e+24,2.5278726317240445e+24,2.6177112093242585e+23,1179713989619784000,7.790468414639561
e+24 (62ms)
      ✔ returns the correct value for
5.45844215431389e+24,4.907457615737792e+24,6.261192962207383e+24,1034863877154841900,5.199917799591388e+2
4 (53ms)

Comptroller
    setActionsPaused
      ✔ reverts if AccessControlManager does not allow it
```

```
      ✔ reverts if the market is not listed
      ✔ does nothing if the actions list is empty (57ms)
      ✔ does nothing if the markets list is empty (54ms)
      ✔ can pause one action on several markets (55ms)
      ✔ can pause several actions on one market (64ms)
      ✔ can pause and unpause several actions on several markets (181ms)
      ✔ reverts if the market is paused (39ms)
      ✔ reverts if market is not listed
      ✔ reverts if maxloops limit is crossed
  NativeTokenGateway
    wrapAndSupply
      ✔ should revert when zero amount is provided to mint (38ms)
      ✔ should wrap and supply eth (322ms)
    redeemUnderlyingAndUnwrap
      ✔ should revert when sender is not approved to redeem on behalf (73ms)
      ✔ should redeem underlying tokens and unwrap and sent it to the user (394ms)
    borrowAndUnwrap
      ✔ should revert when sender is not approved to borrow on behalf (71ms)
      ✔ should borrow and unwrap weth and sent it to borrower (753ms)
    wrapAndRepay
      ✔ should wrap and repay (693ms)
    sweepNative
      ✔ should revert when called by non owener
      ✔ should execute successfully (73ms)
    SweepToken
      ✔ should revert when called by non owner
      ✔ should sweep all tokens (96ms)
  VToken
    accrueInterest
      ✔ reverts if the interest rate is absurdly high (165ms)
      ✔ fails if new borrow rate calculation fails (135ms)
      ✔ fails if simple interest factor calculation fails (127ms)
      ✔ fails if new borrow index calculation fails (141ms)
      ✔ fails if new borrow interest index calculation fails (167ms)
      ✔ fails if interest accumulated calculation fails (162ms)
      ✔ fails if new total borrows calculation fails (162ms)
      ✔ fails if interest accumulated for reserves calculation fails (194ms)
      ✔ fails if new total reserves calculation fails (261ms)
      ✔ succeeds and saves updated values in storage on success (295ms)

  VToken
    borrowFresh
      ✔ fails if comptroller tells it to (38ms)
      ✔ proceeds if comptroller tells it to (68ms)
      ✔ fails if market not fresh (64ms)
      ✔ continues if fresh (96ms)
      ✔ fails if error if protocol has less than borrowAmount of underlying = cash - reserves  (71ms)
      ✔ fails if borrowBalanceStored fails (due to non-zero stored principal with zero account index)
(192ms)
      ✔ fails if calculating account new total borrow balance overflows (192ms)
      ✔ fails if calculation of new total borrow balance overflows (76ms)
      ✔ reverts if transfer out fails (74ms)
      ✔ transfers the underlying cash, tokens, and emits Transfer, Borrow events (166ms)
      ✔ stores new borrow principal and interest index (289ms)
    borrow
      ✔ emits a borrow failure if interest accrual fails (91ms)
      ✔ returns error from borrowFresh without emitting any extra logs (97ms)
      ✔ returns success from borrowFresh and transfers the correct amount (204ms)
    borrowBehalf
      ✔ reverts when caller is not approved by borrower to borrow on his behalf (65ms)
      ✔ returns success from borrowBehalf and transfers the correct amount (128ms)
    repayBorrowFresh
      benefactor paying
```

```
    ✔ fails if repay is not allowed (39ms)
    ✔ fails if block number ≠ current block number (68ms)
    ✔ fails if insufficient approval (79ms)
    ✔ fails if insufficient balance (71ms)
    ✔ returns an error if calculation of new total borrow balance fails (101ms)
    ✔ reverts if doTransferIn fails (73ms)
    ✔ transfers the underlying cash, and emits Transfer, RepayBorrow events (121ms)
    ✔ stores new borrow principal and interest index (194ms)
  borrower paying
    ✔ fails if repay is not allowed (38ms)
    ✔ fails if block number ≠ current block number (70ms)
    ✔ fails if insufficient approval (76ms)
    ✔ fails if insufficient balance (76ms)
    ✔ returns an error if calculation of new total borrow balance fails (277ms)
    ✔ reverts if doTransferIn fails (78ms)
    ✔ transfers the underlying cash, and emits Transfer, RepayBorrow events (123ms)
    ✔ stores new borrow principal and interest index (211ms)
repayBorrow
  ✔ emits a repay borrow failure if interest accrual fails (81ms)
  ✔ returns error from repayBorrowFresh without emitting any extra logs (141ms)
  ✔ returns success from repayBorrowFresh and repays the right amount (227ms)
    ✔ repays the full amount owed if payer has enough (189ms)
    ✔ fails gracefully if payer does not have enough (149ms)
  repayBorrowBehalf
    ✔ emits a repay borrow failure if interest accrual fails (79ms)
    ✔ returns error from repayBorrowFresh without emitting any extra logs (123ms)
    ✔ returns success from repayBorrowFresh and repays the right amount (216ms)

VToken
  liquidateBorrowFresh
    ✔ fails if comptroller tells it to (40ms)
    ✔ proceeds if comptroller tells it to (257ms)
    ✔ fails if market not fresh (72ms)
    ✔ fails if collateral market not fresh (179ms)
    ✔ fails if borrower is equal to liquidator (68ms)
    ✔ fails if repayAmount = 0 (69ms)
    ✔ fails if calculating seize tokens fails and does not adjust balances (1251ms)
    ✔ fails if repay fails (77ms)
    ✔ reverts if seize fails (190ms)
    ✔ transfers the cash, borrows, tokens, and emits Transfer, LiquidateBorrow events (1341ms)
  liquidateBorrow
    ✔ emits a liquidation failure if borrowed asset interest accrual fails (115ms)
    ✔ emits a liquidation failure if collateral asset interest accrual fails (176ms)
    ✔ returns error from liquidateBorrowFresh without emitting any extra logs (198ms)
    ✔ returns success from liquidateBorrowFresh and transfers the correct amounts (1467ms)
  seize
    ✔ fails if seize is not allowed (39ms)
    ✔ fails if vTokenBalances[borrower] < amount (82ms)
    ✔ fails if vTokenBalances[liquidator] overflows (78ms)
    ✔ succeeds, updates balances, adds to reserves, and emits Transfer and ReservesAdded events (624ms)

VToken
  mintFresh
    ✔ fails if comptroller tells it to (39ms)
    ✔ proceeds if comptroller tells it to (82ms)
    ✔ fails if not fresh (66ms)
    ✔ continues if fresh (107ms)
    ✔ fails if insufficient approval (72ms)
    ✔ fails if insufficient balance (71ms)
    ✔ proceeds if sufficient approval and balance (81ms)
    ✔ fails if exchange calculation fails (98ms)
    ✔ fails if transferring in fails (70ms)
    ✔ transfers the underlying cash, tokens, and emits Mint, Transfer events (494ms)
```

```
mint
  ✔ emits a mint failure if interest accrual fails (109ms)
  ✔ returns error from mintFresh without emitting any extra logs (73ms)
  ✔ returns success from mintFresh and mints the correct number of tokens (196ms)
  ✔ emits an AccrueInterest event (333ms)
redeemFreshTokens
  ✔ fails if comptroller tells it to (40ms)
  ✔ fails if not fresh (58ms)
  ✔ continues if fresh (108ms)
  ✔ fails if insufficient protocol cash to transfer out (61ms)
  ✔ fails if exchange calculation fails (60ms)
  ✔ fails if transferring out fails (74ms)
  ✔ fails if total supply < redemption amount (80ms)
  ✔ reverts if new account balance underflows (79ms)
  ✔ transfers the underlying cash, tokens, and emits Redeem, Transfer events (480ms)
redeemFreshAmount
  ✔ fails if comptroller tells it to (41ms)
  ✔ fails if not fresh (59ms)
  ✔ continues if fresh (104ms)
  ✔ fails if insufficient protocol cash to transfer out (58ms)
  ✔ fails if exchange calculation fails (56ms)
  ✔ fails if transferring out fails (76ms)
  ✔ fails if total supply < redemption amount (78ms)
  ✔ reverts if new account balance underflows (78ms)
  ✔ transfers the underlying cash, tokens, and emits Redeem, Transfer events (478ms)
redeem
  ✔ emits a redeem failure if interest accrual fails (142ms)
  ✔ returns error from redeemFresh without emitting any extra logs (201ms)
  ✔ returns success from redeemFresh and redeems the right amount (240ms)
  ✔ revert if exchange rate is high and amount is not enough for a token (127ms)
  ✔ returns success from redeemFresh and redeems the right amount of underlying (180ms)
  ✔ emits an AccrueInterest event (217ms)
redeemBehalf
  ✔ reverts when caller is not the user's approved delegate (70ms)
  ✔ returns success from redeemBehalf and transfers the correct amount (84ms)
redeemUnderlyingBehalf
  ✔ reverts when caller is not the user's approved delegate (69ms)
  ✔ returns success from redeemUnderlyingBehalf and transfers the correct amount (85ms)

VToken
  setProtocolSeizeShare
    ✔ reverts if access control manager does not allow the call (72ms)
    ✔ reverts if the provided seize share is larger than the liquidation incentive minus one (85ms)
    ✔ updates protocolSeizeShare and emits an event on success (83ms)
  set access control manager
    ✔ reverts if not an owner set access control manager (38ms)
    ✔ success by admin (38ms)
  set interestRateModel
    ✔ reverts if rejected by access control manager (48ms)
    ✔ success if allowed to set interest rate model (58ms)
  set reserve factor
    ✔ reverts if rejected by access control manager (47ms)
    ✔ success if allowed to set setReserveFactor (48ms)
  setProtocolShareReserve
    ✔ reverts if called by a non-owner (40ms)
    ✔ reverts if zero address (40ms)
    ✔ sets protocol share reserve if called by admin (39ms)
  setShortfallContract
    ✔ reverts if called by a non-owner (39ms)
    ✔ reverts if zero address (40ms)
    ✔ sets shortfall contract if called by admin (38ms)

VToken
```

```
    transfer
      ✔ cannot transfer from a zero balance (81ms)
      ✔ transfers 50 tokens (179ms)
      ✔ doesn't transfer when src == dst (110ms)
      ✔ approve and transfer (460ms)
      ✔ rejects transfer when not allowed (106ms)

  UpgradedVToken: Tests
    ✔ Upgrade the vToken contract (154ms)

VENUS PROTOCOL:

 MoveDebtDelegate
    setBorrowAllowed
      ✔ fails if called by a non-owner
      ✔ fails if called with zero address for vTokenToBorrow
      ✔ sets borrowAllowed to the specified value
      ✔ emits an event
      ✔ does not emit an event if no-op
    setRepaymentAllowed
      ✔ fails if called by a non-owner
      ✔ fails if called with zero address for vTokenToRepay
      ✔ sets borrowAllowed to the specified value
      ✔ emits an event
      ✔ does not emit an event if no-op
  moveDebt
      ✔ fails if called with a token that is not allowed to be borrowed
      ✔ fails if called with a token that is not allowed to be repaid
      ✔ fails if called with a borrower who is not in the repayment allowlist
      ✔ succeeds if repayments are allowed for ANY_USER (121ms)
      ✔ fails if comptrollers don't match (62ms)
      ✔ fails if repayBorrowBehalf returns a non-zero error code (46ms)
      ✔ fails if borrowBehalf returns a non-zero error code (91ms)
      ✔ transfers repayAmount of vTokenToRepay.underlying() from the sender (104ms)
      ✔ approves vToken to transfer money from the contract (109ms)
      ✔ calls repayBorrowBehalf after transferring the underlying to self (115ms)
      ✔ converts the amounts using the oracle exchange rates (114ms)
      ✔ uses the actually repaid amount rather than specified amount (115ms)
      ✔ transfers the actually borrowed amount to the owner (119ms)
  sweepTokens
      ✔ fails if called by a non-owner
      ✔ transfers the full balance to the owner

  assetListTest
    swapDebt
      ✔ fails if called by a non-owner
      ✔ fails if comptrollers don't match (85ms)
      ✔ fails if repayBorrowBehalf returns a non-zero error code (69ms)
      ✔ fails if borrowBehalf returns a non-zero error code (139ms)
      ✔ transfers repayAmount of underlying from the sender (163ms)
      ✔ approves vToken to transfer money from the contract (181ms)
      ✔ calls repayBorrowBehalf after transferring the underlying to self (176ms)
      ✔ converts the amounts using the oracle exchange rates (171ms)
      ✔ uses the actually repaid amount rather than specified amount (176ms)
      ✔ transfers the actually borrowed amount to the owner (186ms)
    sweepTokens
      ✔ fails if called by a non-owner
      ✔ transfers the full balance to the owner (40ms)


  TokenRedeemer
    redeemAndTransfer
      ✔ should fail if called by a non-owner
```

```
    ✔ should fail if redeem fails (103ms)
    ✔ should succeed with zero amount (186ms)
    ✔ should redeem all vTokens (266ms)
    ✔ should transfer all underlying to the receiver (315ms)

  liquidating BNB debt
    ✔ fails if msg.value is not equal to repayment amount (276ms)
    ✔ transfers BNB from the liquidator (303ms)
    ✔ calls liquidateBorrow on VBNB (261ms)
    ✔ forwards BNB to VBNB contract (286ms)
```

Code Coverage

Note that the full output of the coverage data was pruned to include only the relevant files in scope. Coverage in the `isolated-pools` repository is exceptionally high. We recommend increasing the coverage for all the relevant contracts in the `venus-protocol` repository to a greater degree.

ISOLATED POOLS:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	97.41	77.01	95.29	95.37	
Comptroller.sol	97.2	85.63	100	95.8	... 80,988,1400
ComptrollerInterface.sol	100	100	100	100	
ComptrollerStorage.sol	100	100	100	100	
VToken.sol	99.66	69.75	100	96.62	... 5,1370,1407
VTokenInterfaces.sol	100	100	100	100	
contracts/Gateway/	100	62.5	90.91	96.43	
INativeTokenGateway.sol	100	100	100	100	
NativeTokenGateway.sol	100	62.5	90.91	96.43	180,191
contracts/Gateway/Interfaces/	100	100	100	100	
IVToken.sol	100	100	100	100	
IWrappedNative.sol	100	100	100	100	
All files	97.07	73.77	94.35	95.54	

VENUS PROTOCOL:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/Comptroller/Diamond/facets/	78.17	66.06	82.93	78.85	
MarketFacet.sol	94.23	77.78	76.92	93.75	45,212,216,217

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/Comptroller/Diamond/interfaces/	100	100	100	100	
IMarketFacet.sol	100	100	100	100	
contracts/Tokens/VTokens/	56.13	38.98	51.33	60.91	
VBep20.sol	57.58	25	50	61.11	... 144,145,173
VToken.sol	62.09	39.91	67.27	66.02	... 9,1660,1665
All files	67.09	47.99	64.76	68.51	

Changelog

- 2024-03-01 - Initial Report
- 2024-03-05 - 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

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.