Quantstamp

RiskOracle (Venus)

Executive Summary

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

Type	Risk Oracle
Timeline	2025-02-11 through 2025-02-13
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• VenusProtocol/governance-contracts #02d8986 • VenusProtocol/venus-protocol #baa689e
Auditors	• Julio Aguilar Auditing Engineer • Jennifer Wu Auditing Engineer • Roman Rohleder Senior Auditing Engineer

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

Summary of Findings

The Venus team has integrated Chaos Labs' `RiskOracle` into the Venus pipeline to apply on-chain recommendations. This audit focuses on the first phase of this integration, where the recommendations are limited to updating the borrow and supply caps for Venus pools. Additionally, the team extended the Core Pool Comptroller interface to align with the Isolated Pools Comptroller, which was also reviewed in this audit.

The code is well-documented, structured, and follows best practices. Testing is extensive, with the Risk Steward contracts achieving over 90% coverage, though testing diamond facet contracts in the comptroller could be improved. While the audit did not identify any critical vulnerabilities, we recommend implementing the suggested improvements.

Fix-Review Update: the Venus team fixed all the issues.

ID	DESCRIPTION	SEVERITY	STATUS
RIS-1	**Markets with Zero Caps Cannot Be Updated by the Risk Steward**	● 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 code expands over 2 repositories. The `governance-contracts` repository includes the new Risk Steward feature, while the `venus-protocol` repository involves the files that were refactored to include function aliases of existing functions.

Files Included

- https://github.com/VenusProtocol/venus-protocol
  - Commit Hash: `baa689ebfc05447d2461c6a5966a971708c85fbc`
  - contracts/Comptroller/Diamond/
    - facets/
      - MarketFacet.sol
      - PolicyFacet.sol
      - SetterFacet.sol
    - Interfaces
      - IMarketFacet.sol
      - IPolicyFacet.sol
      - ISetterFacet.sol
- https://github.com/VenusProtocol/governance-contracts
  - Commit Hash: `02d89861ecddb947bfe1165ba8ddb0485f7c5cd9`
  - contracts/
    - RiskSteward/
      - RiskStewardReceiver.sol
      - MarketCapsRiskSteward.sol
      - IRiskSteward.sol
      - IRiskStewardReceiver.sol
    - interfaces/
      - ICorePoolComptroller.sol
      - IIsolatedPoolsComptroller.sol
      - IRiskOracle.sol
      - IVToken.sol

Files Excluded

Everything else.

Operational Considerations

The system is upgradeable, allowing for bug fixes and new feature additions. However, this also introduces a security risk if the admin is compromised and deploys malicious or unaudited code. We assume the admin and other privileged addresses are multi-sigs following the best practices regarding key management to mitigate such cases.

Key Actors And Their Capabilities

In the `governance-contracts` repository, the contracts in scope use `AccessControlledV8`, which is an extension of Open Zeppelin's `AccessControl` developed by Venus, to validate the caller is allowed to call certain functions.

- In the `RiskStewardReceiver`, the allowed addresses can pause and unpause the contract. They can also call `setRiskParameterConfig()` to set a new config for a given `updateType`, and invoke `toggleConfigActive()` to activate or deactivate updating a given `updateType`.
- In the `MarketCapsRiskSteward`, the allowed addresses can set the max allowed update delta possible for the borrow and supply caps. Additionally, only after verifying that the `RiskOracle` has a valid update can the `RiskStewardReceiver` call `MarketCapsRiskSteward` which is then able to set the caps in the Venus comptroller.

In the `venus-protocol` repository, the diamond facets in scope use the admin address and the `AccessControlManagerV5` contract which has a similar functionality as `AccessControlledV8`. In the contracts in scope, the different admins can list and unlist markets, pause specific actions on given markets, and also update the price oracle address, the close factor, the collateral factor, the liquidation incentives, and the borrow and supply caps.

Findings

RIS-1
Markets with Zero Caps Cannot Be Updated by the Risk Steward

● Informational ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `7cf64a20a17edf84215a955f8d681e2917d41298`.

File(s) affected: `MarketCapsRiskSteward.sol`

Description: The `MarketCapsRiskSteward` calculates the allowed range for cap updates based on a percentage of the current cap (`previousValue`). If `previousValue` is zero, any non-zero `newValue` will exceed the maximum permissible difference. When the cap is zero, any new non-zero value fails this check, blocking updates through the Risk Steward.

Recommendation: Document this limitation so developers and users understand that a zero cap must be manually set to a non-zero value by an admin before subsequent range-based updates can occur.

Auditor Suggestions

S1 Code NatSpec Improvements

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `fa12cc7c45f660f6ea8ee825dc34c8eb2b4b07ff`.

File(s) affected: `RiskStewardReceiver.sol`, `MarketCapsRiskSteward.sol`

Description: The following is a list of missing or incorrect NatSpec:

1. `RiskStewardReceiver.setRiskParameterConfig()`: Missing NatSpec comment for emitted event `RiskParameterConfigSet()`.
2. `MarketCapsRiskSteward.initialize()`: `is 0` → `is 0 or greater than MAX_BPS`.
3. `MarketCapsRiskSteward.setMaxDeltaBps()`: `is 0` → `is 0 or greater than MAX_BPS`.
4. `MarketCapsRiskSteward._processSupplyCapUpdate()`: Missing `@custom:error UpdateNotInRange if the update is not within the allowed range`.
5. `MarketCapsRiskSteward._processBorrowCapUpdate()`: Missing `@custom:error UpdateNotInRange if the update is not within the allowed range`.

Recommendation: We recommend applying the suggestion improvements.

S2 Upgradeable Contract Storage Gaps

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `9886fe6c7740e3dec70e8d27781f2173e7eebcc7`.

File(s) affected: `MarketCapsRiskSteward.sol`

Description: For future update considerations, add storage gaps in upgradeable contracts to avoid storage collisions when introducing new variables. This pattern is used in OpenZeppelin's upgradeable contracts.

Recommendation: Add a storage gap (e.g., `uint256[50] private __gap;`) near the end of the contract.

S3 Gas Optimization in `_decodeBytesToUint256()`

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `605efdfb5561aaf62bfe0afa8b5b946883a58f96`.

File(s) affected: `MarketCapsRiskSteward.sol`

Description: The function `_decodeBytesToUint256()` is called twice during the update process: in `validateXXXCapUpdate()` and `_updateXXXCaps()`, leading to redundant computation and unnecessary gas consumption.

Recommendation: Instead of calling `_decodeBytesToUint256()` twice, consider passing the already decoded value directly to `_updateSupplyCaps()` and `_updateBorrowCaps()`.

S4 Ownership Can Be Renounced

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `a8ea183304ce28bd2afa6078b843edf98b27e831`.

File(s) affected: `MarketCapsRiskSteward.sol`, `RiskStewardReceiver.sol`

Description: If the owner renounces ownership, all ownable contracts will become ownerless, preventing the execution of any function restricted by the `onlyOwner` modifier. While `MarketCapsRiskSteward` and `RiskStewardReceiver` do not directly use the `onlyOwner` modifier, they inherit from `AccessControlledV8`, which relies on it to update the access control manager's address.

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

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.

Files

- `b07...2d0 ./IMarketFacet.sol`
- `4f7...30a ./IPolicyFacet.sol`
- `f85...949 ./ISetterFacet.sol`
- `fea...b65 ./MarketFacet.sol`
- `210...5f2 ./PolicyFacet.sol`
- `2f2...ca5 ./SetterFacet.sol`
- `082...cd9 ./IVToken.sol`
- `e2e...f8f ./IIsolatedPoolsComptroller.sol`
- `655...947 ./IRiskOracle.sol`
- `2ef...a6a ./ICorePoolComptroller.sol`
- `975...9df ./RiskSteward/RiskStewardReceiver.sol`
- `0c3...0b9 ./RiskSteward/IRiskSteward.sol`
- `0ce...fa6 ./RiskSteward/MarketCapsRiskSteward.sol`
- `671...039 ./RiskSteward/IRiskStewardReceiver.sol`

Tests

- `04d...6e3 ./RiskStewardReceiver.ts`
- `263...24d ./Diamond/liquidateCalculateAmoutSeizeTest.ts`
- `b68...4a9 ./Diamond/pauseTest.ts`
- `622...a2f ./Diamond/XVSSpeeds.ts`
- `5f3...9e6 ./Diamond/assetListTest.ts`
- `f25...232 ./Diamond/diamond.ts`
- `4f0...b88 ./Diamond/comptrollerTest.ts`
- `1fe...335 ./Diamond/accessControl.ts`
- `919...32c ./Diamond/scripts/deploy.ts`

Test Suite Results

Venus has an extensive test suite of over 700 tests in the `venus-protocol` repository and 140 tests in the `governance-contracts` repository.

To run the tests, execute the following steps in both repositories:
`yarn install`
`npx hardhat test`

```
venus-protocol:

Compiled 228 Solidity files successfully (evm targets: istanbul, paris).


  VBNBAdmin
    ✔ set VBNBAdmin as vBNB admin
    harvest income
      ✔ reduce BNB reserves
    set interest rate model
      ✔ setInterestRateModel

  Comptroller
    _initializeMarket
      ✔ Supply and borrow state after initializing the market in the pool
    _setVenusSpeeds
      ✔ Revert on invalid supplySpeeds input
      ✔ Revert on invalid borrowSpeeds input
      ✔ Revert for unlisted market
      ✔ Revert on invalid borrowSpeeds input
      ✔ Updating non-zero speeds after setting it zero

  Comptroller
    _setAccessControlManager
      ✔ Reverts if called by non-admin
      ✔ Reverts if ACM is zero address
      ✔ Sets ACM address in storage
      ✔ should revert on same value
    Access Control
      setCollateralFactor
        ✔ Should have AccessControl
        ✔ Should revert for same values
      setLiquidationIncentive
        ✔ Should have AccessControl
      setMarketBorrowCaps
        ✔ Should have AccessControl
      setMarketSupplyCaps
        ✔ Should have AccessControl
      setProtocolPaused
        ✔ Should have AccessControl
      setActionsPaused
        ✔ Should have AccessControl
      _supportMarket
        ✔ Should have AccessControl
      supportMarket
        ✔ Should have AccessControl
      seizeVenus
        ✔ Should have AccessControl

  Comptroller: assetListTest
    enterMarkets
      ✔ properly emits events
      ✔ adds to the asset list only once (71ms)
      ✔ the market must be listed for add to succeed (49ms)
      ✔ returns a list of codes mapping to user's ultimate membership in given addresses (41ms)
    exitMarket
      ✔ doesn't let you exit if you have a borrow balance (58ms)
      ✔ rejects unless redeem allowed (105ms)
      ✔ accepts when you're not in the market already (62ms)
      ✔ properly removes when there's only one asset (95ms)
      ✔ properly removes when there's only two assets, removing the first (136ms)
      ✔ properly removes when there's only two assets, removing the second (116ms)
      ✔ properly removes when there's only three assets, removing the first (131ms)
      ✔ properly removes when there's only three assets, removing the second (135ms)
      ✔ properly removes when there's only three assets, removing the third (130ms)
    entering from borrowAllowed
      ✔ enters when called by a vtoken (43ms)
      ✔ reverts when called by not a vtoken
      ✔ adds to the asset list only once (54ms)
    unlistMarkets
      ✔ properly emits events and unlist market (83ms)
      ✔ reverts when unlisting not a listed market (69ms)

  Comptroller
    constructor
      ✔ on success it sets admin to creator and pendingAdmin is unset (768ms)
    _setLiquidationIncentive
      ✔ fails if incentive is less than 1e18
      ✔ accepts a valid incentive and emits a NewLiquidationIncentive event
      ✔ should revert on same values
    _setVenusVAIVaultRate
      ✔ should revert on same values
    _setVAIVaultInfo
      ✔ should revert on same values
    _setVAIController
      ✔ should revert on same values
    _setVAIMintRate
      ✔ should revert on same values
    _setLiquidatorContract
      ✔ should revert on same values
      ✔ should revert on zero address
    _setPauseGuardian
      ✔ should revert on same values
    _setVenusSpeeds
      ✔ ensure non zero address for venus speeds
    _setPriceOracle
      ✔ fails if called by non-admin
      ✔ accepts a valid price oracle and emits a NewPriceOracle event
      ✔ setPriceOracle is alias for _setPriceOracle
      ✔ Should revert on same values
    _setComptrollerLens
      ✔ fails if not called by admin
      ✔ should fire an event
      ✔ should revert on same value
    _setCloseFactor
      ✔ fails if not called by admin
      ✔ should revert on same values
      ✔ fails if factor is set out of range
    _setCollateralFactor
      ✔ fails if asset is not listed
      ✔ fails if factor is set without an underlying price
      ✔ succeeds and sets market
      ✔ succeeds and sets market using alias
      ✔ should revert on same values
    _setForcedLiquidation
      ✔ fails if asset is not listed
      ✔ fails if ACM does not allow the call
      ✔ sets forced liquidation
      ✔ should alias setForcedLiquidation to _setForcedLiquidation
      ✔ sets forced liquidation for VAI, even though it is not a listed market (43ms)
      ✔ emits IsForcedLiquidationEnabledUpdated event
    _setForcedLiquidationForUser
      ✔ fails if asset is not listed
      ✔ fails if ACM does not allow the call
      ✔ sets forced liquidation for user
      ✔ sets forced liquidation for VAI, even though it is not a listed market (43ms)
      ✔ emits IsForcedLiquidationEnabledForUserUpdated event
    _supportMarket
      ✔ fails if asset is not a VToken
      ✔ succeeds and sets market
      ✔ cannot list a market a second time
      ✔ can list two different markets (94ms)
    updateDelegate
      ✔ should revert when zero address is passed
      ✔ should revert when approval status is already set to the requested value
      ✔ should emit event on success
    Hooks
      mintAllowed
        ✔ allows minting if cap is not reached
        ✔ reverts if supply cap reached (38ms)
        ✔ reverts if market is not listed
      redeemVerify
        ✔ should allow you to redeem 0 underlying for 0 tokens (476ms)
        ✔ should allow you to redeem 5 underlyig for 5 tokens (257ms)
        ✔ should not allow you to redeem 5 underlying for 0 tokens
      liquidateBorrowAllowed
        Forced liquidations enabled for user
          ✔ enables forced liquidation for user
          ✔ reverts if borrowed market is not listed
          ✔ reverts if collateral market is not listed
          ✔ does not revert if borrowed vToken is VAIController
          ✔ allows liquidations without shortfall
          ✔ allows to repay 100% of the borrow
          ✔ fails with TOO_MUCH_REPAY if trying to repay > borrowed amount
          ✔ checks the shortfall if isForcedLiquidationEnabledForUser is set back to false
        Forced liquidations enabled for entire market
          ✔ reverts if borrowed market is not listed
          ✔ reverts if collateral market is not listed
          ✔ does not revert if borrowed vToken is VAIController
          ✔ allows liquidations without shortfall
          ✔ allows to repay 100% of the borrow
          ✔ fails with TOO_MUCH_REPAY if trying to repay > borrowed amount
```

```
            ✔ checks the shortfall if isForcedLiquidationEnabled is set back to false
          Forced liquidations disabled
            ✔ reverts if borrowed market is not listed
            ✔ reverts if collateral market is not listed
            ✔ does not revert if borrowed vToken is VAIController
            ✔ fails if borrower has 0 shortfall
            ✔ succeeds if borrower has nonzero shortfall
        borrow
          ✔ allows borrowing if cap is not reached (51ms)
          ✔ reverts borrowing if borrow cap is reached (42ms)
          ✔ reverts borrowing if borrow cap is 0 (39ms)
0xe0F209238AaA159EE72EA30be280b6744606ceB4
          ✔ getBorrowingPower is an alias for getAccountLiquidity (51ms)

  Comptroller
    ✔ Revert on check for the function selector
    ✔ Add Facet and function selectors to proxy (45ms)
    ✔ Get all facet function selectors by facet address
    ✔ Get facet position by facet address
    ✔ Get all facet addresses
    ✔ Get all facets address and their selectors
    ✔ Get facet address and position by function selector
    ✔ Remove function selector from facet mapping
    ✔ Replace the function from facet mapping (59ms)
    ✔ Remove all functions (39ms)

  Comptroller
    liquidateCalculateAmountSeize
      ✔ fails if borrowed asset price is 0
      ✔ fails if collateral asset price is 0
      ✔ fails if the repayAmount causes overflow
      ✔ fails if the borrowed asset price causes overflow
      ✔ reverts if it fails to calculate the exchange rate
      ✔ returns the correct value for
1000000000000000000,1000000000000000000,1000000000000000000,1000000000000000000,1000000000000000000
      ✔ returns the correct value for
2000000000000000000,1000000000000000000,1000000000000000000,1000000000000000000,1000000000000000000
      ✔ returns the correct value for
2000000000000000000,2000000000000000000,1420000000000000000,1300000000000000000,2450000000000000000
      ✔ returns the correct value for
2789000000000000000,5230480842000000000,771320000000000000000,1300000000000000000,1.000245e+22
      ✔ returns the correct value for
7.009232529961056e+24,2.5278726317240445e+24,2.6177112093242585e+23,1179713989619784000,7.790468414639561
e+24
      ✔ returns the correct value for
2.0409756365493427e+24,3.3475361108677775e+24,2.4507276676185885e+24,1434793812588402700,6.15997968973083
5e+24

  ComptrollerMock
    _setActionsPaused
      ✔ reverts if the market is not listed
      ✔ does nothing if the actions list is empty
      ✔ does nothing if the markets list is empty
      ✔ can pause one action on several markets
      ✔ can pause several actions on one market
      ✔ can pause and unpause several actions on several markets (85ms)

  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 (78ms)
      ✔ fails if comptrollers don't match (43ms)
      ✔ fails if repayBorrowBehalf returns a non-zero error code
      ✔ fails if borrowBehalf returns a non-zero error code (59ms)
      ✔ transfers repayAmount of vTokenToRepay.underlying() from the sender (70ms)
      ✔ approves vToken to transfer money from the contract (72ms)
      ✔ calls repayBorrowBehalf after transferring the underlying to self (72ms)
      ✔ converts the amounts using the oracle exchange rates (66ms)
      ✔ uses the actually repaid amount rather than specified amount (67ms)
      ✔ transfers the actually borrowed amount to the owner (68ms)
    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 (62ms)
      ✔ fails if repayBorrowBehalf returns a non-zero error code (49ms)
      ✔ fails if borrowBehalf returns a non-zero error code (89ms)
      ✔ transfers repayAmount of underlying from the sender (108ms)
      ✔ approves vToken to transfer money from the contract (103ms)
      ✔ calls repayBorrowBehalf after transferring the underlying to self (106ms)
      ✔ converts the amounts using the oracle exchange rates (110ms)
      ✔ uses the actually repaid amount rather than specified amount (118ms)
      ✔ transfers the actually borrowed amount to the owner (126ms)
    sweepTokens
      ✔ fails if called by a non-owner
      ✔ transfers the full balance to the owner

  Evil Token test
Duplicate definition of Log (Log(string,address), Log(string,uint256))
Duplicate definition of Log (Log(string,address), Log(string,uint256))
Duplicate definition of Log (Log(string,address), Log(string,uint256))
    ✔ Check the updated vToken states after transfer out (598ms)

  BUSDLiquidator
    setLiquidatorShare
      ✔ should set liquidator share (44ms)
      ✔ should emit NewLiquidatorShare event
      ✔ should revert if caller is not owner
      ✔ should revert if new liquidator share is < 1
      ✔ should revert if new liquidator share is > (liquidation incentive - treasury percent)
      ✔ should succeed if new liquidator share is = (liquidation incentive - treasury percent) (42ms)
    liquidateEntireBorrow
      ✔ should repay entire borrow (672ms)
Bal Prev BigNumber { _hex: '0x00', _isBigNumber: true }
Bal After BigNumber { _hex: '0x00', _isBigNumber: true }
      ✔ should seize collateral (697ms)
    liquidateBorrow
      ✔ should repay a part of the borrow (547ms)
      ✔ should seize collateral (582ms)

  TokenRedeemer
    redeemAndTransfer
      ✔ should fail if called by a non-owner
      ✔ should fail if redeem fails (39ms)
      ✔ should succeed with zero amount (76ms)
      ✔ should redeem all vTokens (130ms)
      ✔ should transfer all underlying to the receiver (129ms)
    redeemUnderlyingAndTransfer
      ✔ should fail if called by a non-owner
      ✔ should revert if redeemer does not have vToken balance (63ms)
      ✔ should redeem and transfer successfully (191ms)
    redeemUnderlyingAndRepayBorrowBehalf
      ✔ should revert if redeemer does not have vToken balance (49ms)
      ✔ should redeem and repay successfully (474ms)
    redeemAndBatchRepay
      Generic
        ✔ fails if called by a non-owner
      Full repayment
        Native asset
          ✔ redeems just the required amount of vTokens (290ms)
          ✔ repays all borrows in full (314ms)
          ✔ transfers the excess vTokens to the receiver (194ms)
          ✔ transfers the excess BNB to the receiver (248ms)
        Tokens
          ✔ redeems just the required amount of vTokens (324ms)
          ✔ repays up to specified caps (363ms)
          ✔ repays all borrows in full (377ms)
          ✔ transfers the excess vTokens to the receiver (325ms)
          ✔ transfers the excess underlying to the receiver (406ms)
      Partial repayment
        Native asset
          ✔ redeems all available vTokens, up to 1 vToken wei (206ms)
          ✔ repays the three borrows: [in full, partially, no repayment] (237ms)
          ✔ uses the excess BNB to repay the debt in full (281ms)
          ✔ does not keep any vBNB or BNB balance (220ms)
        Tokens
          ✔ redeems all available vTokens, up to 1 vToken wei (261ms)
          ✔ repays the three borrows: [in full, partially, no repayment] (281ms)
          ✔ uses the excess underlying to repay the debt in full (318ms)
          ✔ does not keep any vToken or underlying balance (311ms)
    batchRepayVAI
      ✔ fails if called by a non-owner
      ✔ repays one borrow successfully (253ms)
      ✔ repays multiple borrows successfully and transfers refund to treasury (653ms)
      ✔ repays up to caps (572ms)
      ✔ partially repays borrows if insufficient VAI (500ms)
      ✔ can repay small amounts without failure (705ms)
    sweepTokens
      ✔ fails if called by a non-owner
      ✔ sweeps tokens to destination if called by owner (43ms)
      ✔ sweeps native asset to destination

  Two Kinks Interest Rate Model Tests
    ✔ Utilization rate: borrows is zero
    ✔ Utilization rate
    ✔ Borrow Rate: below kink1 utilization
    ✔ Borrow Rate: above kink1 and below kink2 utilization
    ✔ Borrow Rate: above kink2 utilization (39ms)
    ✔ Borrow Rate: above kink2 utilization and negative multipliers (66ms)
    ✔ Supply Rate

  VenusLens: Rewards Summary
    ✔ Should get summary for all markets (188ms)

  Liquidator
    splitLiquidationIncentive
network block skew detected; skipping block events (emitted=2631 blockNumber3632)
network block skew detected; skipping block events (emitted=2631 blockNumber3632)
network block skew detected; skipping block events (emitted=2631 blockNumber3632)
      ✔ splits liquidationIncentive between Treasury and Liquidator with correct amounts
    distributeLiquidationIncentive
      ✔ distributes the liquidationIncentive between Treasury and Liquidator with correct amounts (52ms)
      ✔ reverts if transfer to liquidator fails
      ✔ reverts if underlying transfer to protocol share reserves fails (47ms)

  Liquidator
    liquidateBorrow
      liquidating BEP-20 debt
network block skew detected; skipping block events (emitted=2631 blockNumber3639)
        ✔ fails if borrower is zero address
        ✔ fails if some BNB is sent along with the transaction (47ms)
        ✔ transfers the seized collateral to liquidator and protocolShareReserve (157ms)
        ✔ transfers tokens from the liquidator (185ms)
        ✔ approves the borrowed VToken to spend underlying (155ms)
        ✔ calls liquidateBorrow on borrowed VToken (156ms)
        ✔ emits LiquidateBorrowedTokens event (154ms)
      liquidating VAI debt
        ✔ transfers VAI from the liquidator (167ms)
        ✔ approves VAIController to spend VAI (132ms)
        ✔ calls liquidateVAI on VAIController (135ms)
    liquidating BNB debt
      ✔ fails if msg.value is not equal to repayment amount (103ms)
      ✔ transfers BNB from the liquidator (100ms)
      ✔ calls liquidateBorrow on VBNB (104ms)
      ✔ forwards BNB to VBNB contract (100ms)
    setTreasuryPercent
      ✔ updates treasury percent in storage (42ms)
      ✔ fails when permission is not granted
      ✔ fails when the percentage is too high
      ✔ uses the new treasury percent during distributions (187ms)
    Force VAI Liquidation
      ✔ Should able to liquidate any token when VAI debt is lower than minLiquidatableVAI (100ms)
      ✔ Should not able to liquidate any token when VAI debt is greater than minLiquidatableVAI
      ✔ Should able to liquidate any token when VAI debt is greater than minLiquidatableVAI but forced
liquidation is enabled
      ✔ Should able to liquidate VAI token when VAI debt is greater than minLiquidatableVAI (117ms)
      ✔ Should able to liquidate any token and VAI token when force Liquidation is off (151ms)

  Liquidator
    Restricted liquidations
      addToAllowlist
        ✔ fails if not allowed to call
        ✔ adds address to allowlist (39ms)
        ✔ fails if already in the allowlist (45ms)
        ✔ emits LiquidationPermissionGranted event
      removeFromAllowlist
        ✔ fails if not allowed to call
        ✔ fails if not in the allowlist
        ✔ removes address from allowlist (100ms)
        ✔ emits LiquidationPermissionRevoked event (47ms)
      restrictLiquidation
        ✔ fails if not allowed to call
        ✔ restricts liquidations for the borrower
        ✔ fails if already restricted (55ms)
        ✔ emits LiquidationRestricted event
      unrestrictLiquidation
        ✔ fails if not allowed to call
        ✔ removes the restrictions for the borrower (68ms)
        ✔ fails if not restricted
        ✔ emits LiquidationRestricted event (43ms)
      liquidateBorrow
        ✔ fails if the liquidation is restricted (43ms)
        ✔ proceeds with the liquidation if the guy is allowed to (68ms)

  PrimeScenario Token
    setMaxLoopsLimit()
      ✔ Revert when maxLoopsLimit setter is called by non-owner
      ✔ Revert when new loops limit is less than old limit
      ✔ maxLoopsLimit setter success
    protocol setup
      ✔ markets added
      ✔ borrow balance
      ✔ get markets in prime
    mint and burn
      ✔ stake and mint (341ms)
      ✔ stake and unstake (219ms)
      ✔ stake manually (226ms)
      ✔ burn revocable token (603ms)
      ✔ cannot burn irrevocable token (571ms)
      ✔ manually burn irrevocable token (448ms)
      ✔ issue (592ms)
      ✔ upgrade (468ms)
      ✔ stake, issue and unstake (821ms)
      ✔ issue, stake and burn (759ms)
    boosted yield
      ✔ calculate score (125ms)
network block skew detected; skipping block events (emitted=3699 blockNumber7779709)
      ✔ accrue interest - prime token minted after market is added (370ms)
      ✔ claim interest (256ms)
      update score
        ✔ add existing market after issuing prime tokens - update score gradually (784ms)
```

```
network block skew detected; skipping block events (emitted=3703 blockNumber7779713)
network block skew detected; skipping block events (emitted=3703 blockNumber7779713)
network block skew detected; skipping block events (emitted=3703 blockNumber7779713)
        ✔ add existing market after issuing prime tokens - update score manually (1325ms)
    PLP integration
      ✔ claim interest (344ms)
      ✔ APR Estimation (79ms)
      ✔ Hypothetical APR Estimation (211ms)

  PrimeLiquidityProvider: tests
    Testing all initalized values
      ✔ Tokens intialized
      ✔ Distribution Speed
    Testing all setters
      ✔ Revert on invalid args for initializeTokens
      ✔ Revert on re-intializing token
      ✔ initializeTokens success
      ✔ pauseFundsTransfer
      ✔ resumeFundsTransfer (57ms)
      ✔ Revert on invalid args for setTokensDistributionSpeed
      ✔ Revert on non initialized token
      ✔ Revert on invalid distribution speed for setTokensDistributionSpeed (61ms)
      ✔ setTokensDistributionSpeed success with default max speed (60ms)
      ✔ setTokensDistributionSpeed success (67ms)
      ✔ setMaxTokensDistributionSpeed success
      ✔ Reverts on setting prime address same as previous
      ✔ Revert on invalid prime token address
      ✔ Revert when prime token setter is called by non-owner
      ✔ setPrimeToken success
      ✔ Revert when maxLoopsLimit setter is called by non-owner
      ✔ Revert when new loops limit is less than old limit
      ✔ maxLoopsLimit setter success (43ms)
    Accrue tokens
      ✔ Revert on non initialized token
      ✔ Accrue amount for tokenA (71ms)
      ✔ Accrue amount for multiple tokens (477ms)
    Release funds to prime contract
      ✔ Revert on funds transfer Paused (64ms)
      ✔ Revert on invalid caller
      ✔ Release funds success (95ms)
    Sweep token
      ✔ Revert on insufficient balance
      ✔  Sweep token success (58ms)

  Swap Contract
    ✔ revert if vToken address is not listed
    Setter
      ✔ should reverted if zero address
      ✔ should reverted if vToken not listed
      ✔ setting address for VBNBToken
    Swap
      ✔ revert if path length is 1
      ✔ revert if deadline has passed
      ✔ revert if output amoutn is below minimum
      ✔ should be reverted if tokenA == tokenB
      ✔ should swap tokenA -> tokenB
      ✔ revert if deadline has passed
      ✔ revert if address zero
      ✔ should reverted if first address in not WBNB address
      ✔ should reverted if output amount is below minimum (39ms)
      ✔ should swap BNB -> token (48ms)
      ✔ revert if deadline has passed
      ✔ should swap tokenA -> tokenB  at supporting fee
      ✔ should reverted if deadline passed
      ✔ should swap BNB -> token  at supporting fee
      ✔ should swap EXact token -> BNB at supporting fee  (79ms)
      ✔ should swap tokesn for Exact BNB
      ✔ should swap tokens for Exact Tokens
      ✔ should swap tokens for Exact BNB
      ✔ should swap BNB for Exact Tokens
    Supply
      ✔ revert if deadline has passed
      ✔ swap tokenA -> tokenB --> supply tokenB (90ms)
      ✔ swap BNB -> token --> supply token (101ms)
      ✔ revert if deadline has passed at supporting fee
      ✔ swap tokenA -> tokenB --> supply tokenB at supporting fee (93ms)
      ✔ swap BNB -> token --> supply token at supporting fee (98ms)
      ✔ swap tokenA -> exact tokenB (86ms)
      ✔ swap bnb -> exact tokenB (98ms)
      ✔ Exact tokens -> BNB and supply
      ✔ Exact tokens -> BNB and supply at supporting fee
    Repay
      ✔ revert if deadline has passed
      ✔ swap tokenA -> tokenB --> supply tokenB (80ms)
      ✔ swap BNB -> token --> supply token (84ms)
      ✔ revert if deadline has passed at supporting fee
      ✔ swap tokenA -> tokenB --> reapy tokenB at supporting fee (93ms)
      ✔ swap BNB -> token --> repay token at supporting fee (86ms)
      ✔ swap tokenA -> exact tokenB (87ms)
      ✔ swap tokenA -> full debt of tokenB (85ms)
      ✔ swap bnb -> exact tokenB (96ms)
      ✔ swap bnb -> full tokenB debt (104ms)
      ✔ Exact tokens -> BNB at supporting fee (72ms)
      ✔ Exact tokens -> BNB (52ms)
      ✔ Tokens -> Exact BNB (48ms)
      ✔ Tokens -> Exact BNB and supply
      ✔ Tokens -> full debt of BNB
    Sweep Token
      ✔ Should be reverted if get zero address
      ✔ Sweep ERC-20 tokens (83ms)
    library function
      ✔ Quote function
      ✔ getAmoutIn function
      ✔ getAmoutout function
      ✔ getAmoutout function
      ✔ getAmoutout function

  admin / _setPendingAdmin / _acceptAdmin
    admin()
      ✔ should return correct admin
    pendingAdmin()
      ✔ should return correct pending admin
    _setPendingAdmin()
      ✔ should only be callable by admin
      ✔ should properly set pending admin
      ✔ should properly set pending admin twice (41ms)
      ✔ should emit event
    _acceptAdmin()
      ✔ should fail when pending admin is zero
      ✔ should fail when called by another account (e.g. root)
      ✔ should succeed and set admin and clear pending admin (41ms)
      ✔ should emit log on success

  Unitroller
    constructor
      ✔ sets admin to caller and addresses to 0
    _setPendingImplementation
      Check caller is admin
        ✔ emits a failure log
        ✔ does not change pending implementation address
      succeeding
        ✔ stores pendingComptrollerImplementation with value newPendingImplementation
        ✔ emits NewPendingImplementation event
    _acceptImplementation
      Check caller is pendingComptrollerImplementation  and pendingComptrollerImplementation ≠ address(0)
        ✔ emits a failure log
        ✔ does not change current implementation address
      the brains must accept the responsibility of implementation
        ✔ Store comptrollerImplementation with value pendingComptrollerImplementation
        ✔ Unset pendingComptrollerImplementation
        ✔ Emit NewImplementation(oldImplementation, newImplementation)
        ✔ Emit NewPendingImplementation(oldPendingImplementation, 0)
      fallback delegates to brains
        ✔ forwards reverts
      ✔ gets addresses
      ✔ gets strings
      ✔ gets bools
      ✔ gets list of ints

  Peg Stability Module
    PSM: 18 decimals
      initialization
        ✔ should revert if contract already deployed
        ✔ should initialize sucessfully
        reverts if init address = 0x0:
          ✔ acm
          ✔ treasury
          ✔ stableToken
        reverts if fee init value is invalid
          ✔ feeIn
          ✔ feeOut
      Admin functions
        pause()
          ✔ should revert if not authorised
          ✔ should pause if authorised
          ✔ should revert if already paused
        resume()
          ✔ should revert if not authorised
          ✔ should resume if authorised
          ✔ should revert if already resumed
        setFeeIn(uint256)
          ✔ should revert if not authorised
          ✔ should revert if fee is invalid
          ✔ set the correct fee
        setFeeOut(uint256)
          ✔ should revert if not authorised
          ✔ should revert if fee is invalid
          ✔ set the correct fee
        setVAIMintCap(uint256)
          ✔ should revert if not authorised
          ✔ should set the correct mint cap
        setVenusTreasury(uint256)
          ✔ should revert if not authorised
          ✔ should revert if zero address
          ✔ should set the treasury address
        setOracle(address)
          ✔ should revert if not authorised
          ✔ should revert if oracle address is zero
          ✔ should set the oracle (69ms)
      Pause logic
        ✔ should revert when paused and call swapVAIForStable(address,uint256)
        ✔ should revert when paused and call swapStableForVAI(address,uint256)
      Swap functions
        swapVAIForStable(address,uint256)
          ✔ should revert if receiver is zero address
          ✔ should revert if sender has insufficient VAI balance  (68ms)
          ✔ should revert if VAI transfer fails  (59ms)
          ✔ should revert if VAI to be burnt > vaiMinted  (74ms)
          should sucessfully perform the swap
            Fees: 10%
              ✔ stable token = 1$  (70ms)
              ✔ stable token < 1$  (106ms)
              ✔ stable token > 1$  (73ms)
            Fees: 0%
              ✔ stable token = 1$  (76ms)
              ✔ stable token < 1$  (77ms)
              ✔ stable token > 1$  (104ms)
        swapStableForVAI(address,uint256)
          ✔ should revert if receiver is zero address
          ✔ should revert if VAI mint cap will be reached  (105ms)
          ✔ should revert if amount after transfer is too small   (68ms)
          should sucessfully perform the swap
            Fees: 10%
              ✔ stable token = 1$  (104ms)
              ✔ stable token > 1$  (77ms)
              ✔ stable token < 1$  (115ms)
            Fees: 0%
              ✔ stable token = 1$  (69ms)
              ✔ stable token > 1$  (108ms)
              ✔ stable token < 1$  (84ms)
    PSM: 8 decimals
      initialization
        ✔ should revert if contract already deployed
        ✔ should initialize sucessfully
        reverts if init address = 0x0:
          ✔ acm
          ✔ treasury
          ✔ stableToken
        reverts if fee init value is invalid
          ✔ feeIn
          ✔ feeOut
      Admin functions
        pause()
          ✔ should revert if not authorised
          ✔ should pause if authorised (43ms)
          ✔ should revert if already paused
        resume()
          ✔ should revert if not authorised (42ms)
          ✔ should resume if authorised
          ✔ should revert if already resumed (43ms)
        setFeeIn(uint256)
          ✔ should revert if not authorised
          ✔ should revert if fee is invalid (41ms)
          ✔ set the correct fee
        setFeeOut(uint256)
          ✔ should revert if not authorised (44ms)
          ✔ should revert if fee is invalid
          ✔ set the correct fee (43ms)
        setVAIMintCap(uint256)
          ✔ should revert if not authorised (63ms)
          ✔ should set the correct mint cap (47ms)
        setVenusTreasury(uint256)
          ✔ should revert if not authorised
          ✔ should revert if zero address (48ms)
          ✔ should set the treasury address
        setOracle(address)
          ✔ should revert if not authorised (45ms)
          ✔ should revert if oracle address is zero
          ✔ should set the oracle (68ms)
      Pause logic
        ✔ should revert when paused and call swapVAIForStable(address,uint256)
        ✔ should revert when paused and call swapStableForVAI(address,uint256)
      Swap functions
        swapVAIForStable(address,uint256)
          ✔ should revert if receiver is zero address
          ✔ should revert if sender has insufficient VAI balance  (97ms)
          ✔ should revert if VAI transfer fails  (75ms)
          ✔ should revert if VAI to be burnt > vaiMinted  (92ms)
          should sucessfully perform the swap
            Fees: 10%
              ✔ stable token = 1$  (100ms)
              ✔ stable token < 1$  (131ms)
              ✔ stable token > 1$  (85ms)
            Fees: 0%
              ✔ stable token = 1$  (120ms)
              ✔ stable token < 1$  (92ms)
              ✔ stable token > 1$  (127ms)
        swapStableForVAI(address,uint256)
          ✔ should revert if receiver is zero address
          ✔ should revert if VAI mint cap will be reached  (152ms)
          should sucessfully perform the swap
            Fees: 10%
              ✔ stable token = 1$  (116ms)
              ✔ stable token > 1$  (173ms)
              ✔ stable token < 1$  (121ms)
            Fees: 0%
              ✔ stable token = 1$  (147ms)
              ✔ stable token > 1$  (112ms)
              ✔ stable token < 1$  (149ms)
    PSM: 6 decimals
      initialization
        ✔ should revert if contract already deployed
        ✔ should initialize sucessfully
        reverts if init address = 0x0:
          ✔ acm
          ✔ treasury
          ✔ stableToken
        reverts if fee init value is invalid
          ✔ feeIn
          ✔ feeOut
      Admin functions
        pause()
          ✔ should revert if not authorised (49ms)
          ✔ should pause if authorised
          ✔ should revert if already paused (75ms)
        resume()
          ✔ should revert if not authorised
          ✔ should resume if authorised (68ms)
          ✔ should revert if already resumed
        setFeeIn(uint256)
          ✔ should revert if not authorised (63ms)
          ✔ should revert if fee is invalid (46ms)
          ✔ set the correct fee (78ms)
        setFeeOut(uint256)
          ✔ should revert if not authorised
          ✔ should revert if fee is invalid (49ms)
          ✔ set the correct fee
        setVAIMintCap(uint256)
          ✔ should revert if not authorised (53ms)
          ✔ should set the correct mint cap
        setVenusTreasury(uint256)
          ✔ should revert if not authorised (51ms)
          ✔ should revert if zero address (41ms)
          ✔ should set the treasury address (78ms)
        setOracle(address)
          ✔ should revert if not authorised (40ms)
          ✔ should revert if oracle address is zero (54ms)
          ✔ should set the oracle (65ms)
      Pause logic
        ✔ should revert when paused and call swapVAIForStable(address,uint256) (40ms)
        ✔ should revert when paused and call swapStableForVAI(address,uint256)
      Swap functions
        swapVAIForStable(address,uint256)
          ✔ should revert if receiver is zero address
          ✔ should revert if sender has insufficient VAI balance  (75ms)
          ✔ should revert if VAI transfer fails  (127ms)
          ✔ should revert if VAI to be burnt > vaiMinted  (69ms)
          should sucessfully perform the swap
            Fees: 10%
              ✔ stable token = 1$  (160ms)
              ✔ stable token < 1$  (118ms)
              ✔ stable token > 1$  (177ms)
            Fees: 0%
              ✔ stable token = 1$  (97ms)
              ✔ stable token < 1$  (143ms)
              ✔ stable token > 1$  (126ms)
        swapStableForVAI(address,uint256)
          ✔ should revert if receiver is zero address
          ✔ should revert if VAI mint cap will be reached  (97ms)
          should sucessfully perform the swap
            Fees: 10%
              ✔ stable token = 1$  (154ms)
              ✔ stable token > 1$  (128ms)
              ✔ stable token < 1$  (146ms)
            Fees: 0%
              ✔ stable token = 1$  (135ms)
              ✔ stable token > 1$  (174ms)
              ✔ stable token < 1$  (144ms)

  VAIController
```

```
    ✔ check wallet usdt balance
  #getMintableVAI
    ✔ oracle
    ✔ getAssetsIn
    ✔ getAccountSnapshot
    ✔ getUnderlyingPrice (72ms)
    ✔ getComtroller
    ✔ success (187ms)
  #mintVAI
    ✔ success (368ms)
    ✔ fails if there's not enough collateral (258ms)
    ✔ fails if minting beyond mint cap (443ms)
    ✔ fails if can't set the minted amount in comptroller (310ms)
    ✔ puts previously accrued interest to pastInterest (801ms)
  #repayVAI
    ✔ reverts if the protocol is paused
    ✔ success for zero rate (185ms)
    ✔ success for 1.2 rate repay all (317ms)
    ✔ success for 1.2 rate repay half (294ms)
    ✔ fails if can't set the new minted amount in comptroller (173ms)
  #repayVAIBehalf
    ✔ reverts if called with borrower = zero address
    ✔ reverts if the protocol is paused
    ✔ success for zero rate (177ms)
    ✔ success for 1.2 rate repay all (275ms)
    ✔ success for 1.2 rate repay half (247ms)
  #getHypotheticalAccountLiquidity
    ✔ success for zero rate 0.9 vusdt collateralFactor (307ms)
    ✔ success for 1.2 rate 0.9 vusdt collateralFactor (381ms)
  #liquidateVAI
    ✔ liquidationIncentiveMantissa
    ✔ reverts if the protocol is paused (39ms)
    ✔ success for zero rate 0.2 vusdt collateralFactor (1047ms)
    ✔ success for 1.2 rate 0.3 vusdt collateralFactor (1190ms)
  #getVAIRepayRate
    ✔ success for zero baseRate (38ms)
    ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 1e18 (193ms)
    ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 0.5 * 1e18 (195ms)
  #getVAIRepayAmount
    ✔ reverts if the protocol is paused
    ✔ success for zero rate
    ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 1e18 (187ms)
    ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 0.5 * 1e18 (234ms)
  #getVAICalculateRepayAmount
    ✔ success for zero rate (45ms)
    ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 1e18 (327ms)
    ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 0.5 * 1e18 (356ms)
  #getMintableVAI
    ✔ include current interest when calculating mintable VAI (333ms)
  #accrueVAIInterest
    ✔ success for called once (118ms)
    ✔ success for called twice (190ms)
  #setBaseRate
    ✔ fails if access control does not allow the call (40ms)
    ✔ emits NewVAIBaseRate event (48ms)
    ✔ sets new base rate in storage (39ms)
  #setFloatRate
    ✔ fails if access control does not allow the call (38ms)
    ✔ emits NewVAIFloatRate event (39ms)
    ✔ sets new float rate in storage (39ms)
  #setMintCap
    ✔ fails if access control does not allow the call (38ms)
    ✔ emits NewVAIMintCap event (41ms)
    ✔ sets new mint cap in storage (46ms)
  #setReceiver
    ✔ fails if called by a non-admin
    ✔ reverts if the receiver is zero address
    ✔ emits NewVAIReceiver event
    ✔ sets VAI receiver address in storage
  #setAccessControl
    ✔ reverts if called by non-admin
    ✔ reverts if ACM is zero address
```

```
        ✔ emits NewAccessControl event (51ms)
        ✔ sets ACM address in storage (47ms)
      #prime
        ✔ prime integration (1496ms)

    VAIVault
      ✔ claim reward (740ms)
      setVenusInfo
        ✔ fails if called by a non-admin
        ✔ fails if XVS address is zero
        ✔ fails if VAI address is zero
        ✔ disallows configuring tokens twice (42ms)

    VRTVault
      unit tests
        setLastAccruingBlock
          ✔ fails if ACM disallows the call (40ms)
          ✔ fails if trying to set lastAccuringBlock to some absurdly high value
          ✔ fails if lastAccuringBlock has passed (76ms)
          ✔ fails if trying to set lastAccuringBlock to some past block (39ms)
          ✔ fails if trying to set lastAccuringBlock to the current block
          ✔ correctly sets lastAccuringBlock to some future block (58ms)
          ✔ can move lastAccuringBlock to a later block (101ms)
          ✔ can move lastAccuringBlock to an earlier block (100ms)
          ✔ fails if trying to move lastAccuringBlock to a block in the past (89ms)
      scenario
        ✔ deposit (129ms)
        ✔ should claim reward (76ms)
        ✔ should not claim reward after certain block (123ms)

    VToken
      _setReserveFactorFresh
        ✔ rejects change by non-admin (43ms)
        ✔ rejects change if market not fresh (45ms)
  network block skew detected; skipping block events (emitted=7780070 blockNumber7790259)
  network block skew detected; skipping block events (emitted=7780070 blockNumber7790259)
  network block skew detected; skipping block events (emitted=7780070 blockNumber7790259)
  network block skew detected; skipping block events (emitted=7780068 blockNumber7790259)
        ✔ rejects newReserveFactor that descales to 1 (73ms)
        ✔ accepts newReserveFactor in valid range and emits log (97ms)
        ✔ accepts a change back to zero (189ms)
      _setReserveFactor
        ✔ emits a reserve factor failure if interest accrual fails (123ms)
        ✔ returns error from setReserveFactorFresh without emitting any extra logs (94ms)
        ✔ returns success from setReserveFactorFresh (139ms)
      _reduceReservesFresh
        ✔ fails if called by non-admin (58ms)
        ✔ fails if market not fresh (61ms)
        ✔ fails if amount exceeds available cash (395ms)
        ✔ if there isn't enough cash, reduces with available cash (187ms)
        ✔ increases admin balance and reduces reserves on success (242ms)
      _reduceReserves
        ✔ emits a reserve-reduction failure if interest accrual fails (115ms)
        ✔ returns error from _reduceReservesFresh without emitting any extra logs (187ms)
        ✔ returns success code from _reduceReservesFresh and reduces the correct amount (221ms)

    XVSVault
      setXvsStore
        ✔ fails if XVS is a zero address
        ✔ fails if XVSStore is a zero address
        ✔ fails if the vault is already initialized
      add
        ✔ reverts if ACM does not allow the call (39ms)
        ✔ reverts if xvsStore is not set (40ms)
        ✔ reverts if a pool with this (staked token, reward token) combination already exists (52ms)
        ✔ reverts if staked token exists in another pool (38ms)
        ✔ reverts if reward token is a zero address
        ✔ reverts if staked token is a zero address (38ms)
        ✔ reverts if alloc points parameter is zero (41ms)
        ✔ emits PoolAdded event (70ms)
        ✔ adds a second pool to an existing rewardToken (87ms)
        ✔ sets pool info (94ms)
        ✔ configures reward token in XVSStore (88ms)
      set
        ✔ reverts if ACM does not allow the call (43ms)
        ✔ reverts if pool is not found (38ms)
        ✔ reverts if total alloc points after the call is zero (63ms)
        ✔ succeeds if the pool alloc points is zero but total alloc points is nonzero (242ms)
        ✔ emits PoolUpdated event (65ms)
      setRewardAmountPerBlockOrSecond
        ✔ reverts if ACM does not allow the call (43ms)
        ✔ reverts if the token is not configured in XVSStore (97ms)
        ✔ emits RewardAmountPerBlockUpdated event (97ms)
        ✔ updates reward amount per block (174ms)
      setWithdrawalLockingPeriod
        ✔ reverts if ACM does not allow the call (41ms)
        ✔ reverts if pool does not exist
        ✔ reverts if the lock period is 0
        ✔ reverts if the lock period is absurdly high
        ✔ emits WithdrawalLockingPeriodUpdated event (78ms)
        ✔ updates lock period (180ms)
      pendingReward
        ✔ includes the old withdrawal requests in the rewards computation (301ms)
        ✔ excludes the new withdrawal requests from the rewards computation (305ms)
      deposit
        ✔ reverts if the vault is paused (66ms)
        ✔ reverts if pool does not exist
        ✔ transfers pool token to the vault (106ms)
        ✔ updates user's balance (104ms)
        ✔ fails if there's a pre-upgrade withdrawal request (143ms)
        ✔ succeeds if the pre-upgrade withdrawal request has been executed (487ms)
        ✔ uses the safe _transferReward under the hood (311ms)
      executeWithdrawal
  network block skew detected; skipping block events (emitted=7790816 blockNumber7791819)
  network block skew detected; skipping block events (emitted=7790816 blockNumber7791819)
  network block skew detected; skipping block events (emitted=7790816 blockNumber7791819)
  network block skew detected; skipping block events (emitted=7790816 blockNumber7791819)
        ✔ fails if the vault is paused (70ms)
        ✔ only transfers the requested amount for post-upgrade requests (334ms)
        ✔ handles pre-upgrade withdrawal requests (326ms)
        ✔ handles pre-upgrade and post-upgrade withdrawal requests (591ms)
      requestWithdrawal
        ✔ fails if the vault is paused (68ms)
        ✔ transfers rewards to the user (293ms)
        ✔ uses the safe _transferReward under the hood (302ms)
        ✔ fails if there's a pre-upgrade withdrawal request (216ms)
      claim
        ✔ fails if there's a pre-upgrade withdrawal request (124ms)
        ✔ succeeds if the pre-upgrade withdrawal request has been executed (351ms)
        ✔ excludes pending withdrawals from the user's shares (452ms)
        ✔ correctly accounts for updates in reward per block (326ms)
        ✔ uses the safe _transferReward under the hood (177ms)
      _transferReward
        ✔ sends the available funds to the user (123ms)
        ✔ emits VaultDebtUpdated event if vault debt is updated (95ms)
        ✔ does not emit VaultDebtUpdated event if vault debt is not updated (118ms)
        ✔ records the pending transfer (109ms)
        ✔ records several pending transfers (227ms)
        ✔ sends out the pending transfers in addition to reward if full amount <= funds available (390ms)
        ✔ sends a part of the pending transfers and reward if full amount > funds available (320ms)
      pendingWithdrawalsBeforeUpgrade
        ✔ returns zero if there were no pending withdrawals
        ✔ returns zero if there is only a new-style pending withdrawal (130ms)
        ✔ returns the requested amount if there is an old-style pending withdrawal (49ms)
        ✔ returns the total requested amount if there are multiple old-style pending withdrawals (80ms)
        ✔ returns zero if the pending withdrawal was executed (184ms)
      Scenarios
        ✔ works correctly with multiple claim, deposit, and withdrawal requests (1228ms)

    Prime Token
      mint and burn
  network block skew detected; skipping block events (emitted=7792364 blockNumber7794663)
  network block skew detected; skipping block events (emitted=7792364 blockNumber7794663)
  network block skew detected; skipping block events (emitted=7792364 blockNumber7794663)
  network block skew detected; skipping block events (emitted=7792364 blockNumber7794663)
        ✔ should alias setPrimeToken to _setPrimeToken
        ✔ stake and mint (1470ms)
        ✔ burn revocable token (3308ms)
  network block skew detected; skipping block events (emitted=7794663 blockNumber15570681)
  network block skew detected; skipping block events (emitted=7794663 blockNumber15570681)
  network block skew detected; skipping block events (emitted=7794663 blockNumber15570681)
  network block skew detected; skipping block events (emitted=7794663 blockNumber15570681)
        ✔ cannot burn irrevocable token (3425ms)
        ✔ issue and stake token concurrently (2133ms)
      boosted yield
        ✔ claim interest for multiple users (7772ms)


    718 passing (5m)




  governance-contracts:

  Compiled 97 Solidity files successfully (evm targets: istanbul, paris).


    Access Control
      Access Control
        ✔ only default admin role can give call permissions (64ms)
        ✔ should not have permissions
        ✔ should have permissions (41ms)
        ✔ should revoke role (56ms)
        ✔ should be able to call the function only for the given contract
        ✔ should be able to call the function on every contract

    Omnichain:
      ✔ Reverts if EOA called owner function of bridge
      ✔ Reverts if EOA call execute() without grant permission
      ✔ Reverts when zero value passed
      ✔ Revert if trusted remote is removed by non owner
      ✔ Revert if non trusted remote is removed
      ✔ Reverts when trusted remote is not set
      ✔ Reverts if remote address is more than 20 bytes
      ✔ Reverts with Daily Transaction Limit Exceed
      ✔ Reverts if EOA call setMaxDailyLimit() without grant permisssion
      ✔ Revert if function in not found in function registry
      ✔ Reverts if any user other than owner try to add function in function registry
      ✔ Function registry should not emit event if nonexistant function is removed
      ✔ Function registry should not emit event if function is added twice
      ✔ Reverts if invalid parameters passed in trusted remote (53ms)
      ✔ Reverts if EOA called owner function of Executor
      ✔ Revert if call by non guardian
      ✔ Revert if zero address is passed in guardian
      ✔ should set new guardian for Executor (65ms)
      ✔ Emit TimelocksAdded event (103ms)
      ✔ Emit SetTrustedRemoteAddress event (66ms)
      ✔ Emit ExecuteRemoteProposal event (46ms)
      ✔ Revert initially, success on retry (76ms)
      ✔ Revert when daily limit exceeds in retry (65ms)
      ✔ Emit ProposalExecuted event (73ms)
      ✔ Should update delay of timelock on destination (78ms)
      ✔ Admin can set the new pending admin of Timelock (119ms)
      ✔ Set new pending admin of Timelock through proposal (104ms)
      ✔ should revert when invalid proposalType is passed (40ms)
      ✔ Revert when zero address passed as pending admin
      ✔ Revert when non owner sets the pending admin of Timelock (38ms)
      ✔ Revert if empty proposal
      ✔ Revert on invalid proposal type
      ✔ Revert if same proposal come twice (83ms)
      ✔ Retry message on destination on failure (375ms)
      ✔ Revert retry message on destination if trusted remote is changed (171ms)
      ✔ Retry messages that failed due to low gas at the destination using the Endpoint. (71ms)
      ✔ Reverts when other than guardian call cancel of executor (60ms)
      ✔ Revert if proposal is not queued
      ✔ Revert when proposal is not queued
```

```
      ✔ Emit ProposalCanceled event when proposal gets canceled (65ms)
      ✔ Reverts when cancel is called after execute (78ms)
      ✔ Proposal fails if any number of commands fail on destination
      ✔ Reverts when number of parameters mismatch
      ✔ Refund stucked gas in contract, to given address (50ms)
      ✔ Reverts on passing zero values in parameters in fallback withdraw (59ms)
      ✔ Reverts when value exceeds contract's balance in fallback withdraw (47ms)
      ✔ Reverts when different parameters passed in fallback withdraw (59ms)
      ✔ Reverts when receiver is unable to receive in fallback withdraw (46ms)
      ✔ Refund stucked gas in contract, to given address (49ms)
      ✔ Reverts on passing zero values in parameters in fallback withdraw (57ms)
      ✔ Reverts when different parameters passed in fallback withdraw (59ms)
      ✔ Reverts when receiver is unable to receive in fallback withdraw (46ms)
      ✔ Reverts when daily limit of sending transaction reached
      ✔ Proposal failed when receiving limit reached (109ms)

  Governor Bravo Cast Vote Test
    We must revert if:
      ✔ We cannot propose without enough voting power by depositing xvs to the vault
      after we deposit xvs to the vault
        ✔ There does not exist a proposal with matching proposal id where the current block number is
between the proposal's start block (exclusive) and end block (inclusive)
        ✔ Such proposal already has an entry in its voters set matching the sender (56ms)
        Otherwise
          ✔ we add the sender to the proposal's voters set
        and we take the balance returned by GetPriorVotes for the given sender and the proposal's start
block, which may be zero,
          ✔ and we add that ForVotes (55ms)
          ✔ or AgainstVotes corresponding to the caller's support flag. (56ms)
        castVoteBySig
          ✔ reverts if the signatory is invalid
          ✔ casts vote on behalf of the signatory (57ms)

  Governor Bravo Initializing Test
    initilizer
      ✔ should revert if not called by admin
      ✔ should revert if invalid xvs address
      ✔ should revert if invalid guardian address
      ✔ should revert if timelock adress count differs from governance routes count
      ✔ should revert if proposal config count differs from governance routes count
      ✔ should revert if initialized twice (42ms)

  Governor Bravo Propose Tests
    simple initialization
      ✔ ID is set to a globally unique identifier
      ✔ Proposer is set to the sender
      ✔ Start block is set to the current block number plus vote delay
      ✔ End block is set to the current block number plus the sum of vote delay and vote period
      ✔ ForVotes and AgainstVotes are initialized to zero
      ✔ Executed and Canceled flags are initialized to false
      ✔ ETA is initialized to zero
      ✔ Targets, Values, Signatures, Calldatas are set according to parameters
      ✔ This function returns the id of the newly created proposal. # proposalId(n) = succ(proposalId(n-
1))
      ✔ emits log with id and description (59ms)
      This function must revert if
        ✔ the length of the values, signatures or calldatas arrays are not the same length, (73ms)
        ✔ or if that length is zero or greater than Max Operations.
        Additionally, if there exists a pending or active proposal from the same proposer, we must
revert.
          ✔ reverts with pending
          ✔ reverts with active

  Governor Bravo Queue Tests
    overlapping actions
      ✔ reverts on queueing overlapping actions in same proposal (101ms)
      ✔ reverts on queueing overlapping actions in different proposals (132ms)

  Governor Bravo State Tests
    ✔ Invalid for proposal not found
    ✔ Pending
    ✔ Active
    ✔ Canceled (51ms)
    ✔ Canceled by Guardian (50ms)
    ✔ Defeated
    ✔ Succeeded (69ms)
    ✔ Expired (90ms)
    ✔ Queued (92ms)
    ✔ Executed (117ms)

  TimelockV8 Tests
    ✔ Production timelock returns constant values
    ✔ Production timelock requires setting appropriate delay
    ✔ Production timelock does not allow a null address
    ✔ Test Timelock returns correct for constants
    ✔ Test Timelock allows setting low delay
    ✔ Test timelock does not allow a null address

  Risk Steward
    Access Control
      ✔ should revert if access is not granted for setting risk parameter config
      ✔ should revert if access is not granted for toggling config active
      ✔ should revert if access is not granted for pausing
      ✔ should revert if access is not granted for unpausing
      ✔ should revert if access is not granted for processing update
      ✔ should revert if access is not granted for setting max increase bps
    Upgradeable
      ✔ new implementation should update core comptroller (57ms)
      ✔ new implementation should update risk oracle (89ms)
    Risk Parameter Config
      ✔ should get original risk parameter configs
      ✔ should pause risk parameter configs
      ✔ should revert if pausing unsupported update type
      ✔ should update risk parameter configs
      ✔ should emit RiskParameterConfigSet event
      ✔ should revert if empty updateType is set
      ✔ should revert if debounce is 0
      ✔ should not support zero risk steward address
      ✔ should revert if debounce is less than UPDATE_EXPIRATION_TIME
      ✔ should revert if maxDeltaBps is 0
      ✔ should revert if maxDeltaBps is 10000 or greater
    Risk Steward Pause
      ✔ should toggle paused state
      ✔ should revert if contract is paused
      ✔ should revert if contract is paused
    Risk Parameter Update Reverts under incorrect conditions
      ✔ should revert if updateType is unknown
      ✔ should revert if updateType is implemented (82ms)
      ✔ should revert if updateType is not active (122ms)
      ✔ should revert if the update is expired (86ms)
      ✔ should revert if market is not supported (103ms)
      ✔ should revert if the update is too frequent (197ms)
      ✔ should error on invalid update ID
      ✔ should revert if the update has already been applied (69ms)
      ✔ should revert if the update is out of bounds (201ms)
    Risk Parameter Updates under correct conditions
      ✔ should process update by id (221ms)
      ✔ should process increase updates by parameter and market (211ms)
      ✔ should process decrease updates by parameter and market (210ms)


  140 passing (12s)
```

Code Coverage

The code coverage for the contracts in scope within the `venus-protocol` repository ranges between 63% and 84%. For improved robustness, we recommend achieving coverage above 90%. In contrast, the contracts in scope within the `governance-contracts` repository already exceed 90% coverage.

venus-protocol repository:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
InterfacesV8.sol	100	100	100	100	
contracts/Comptroller/	100	90	100	100	
ComptrollerInterface.sol	100	100	100	100	
ComptrollerLensInterface.sol	100	100	100	100	
ComptrollerStorage.sol	100	100	100	100	
Unitroller.sol	100	90	100	100	
contracts/Comptroller/Diamond/	97.26	59.09	100	95.35	
Diamond.sol	97.26	59.09	100	95.35	109,228,229,230
DiamondConsolidated.sol	100	100	100	100	
contracts/Comptroller/Diamond/facets/	75.55	63.67	80.91	76.22	
FacetBase.sol	62.22	55.88	86.67	59.18	... 128,211,224
MarketFacet.sol	98.78	66.67	94.12	98.98	66
PolicyFacet.sol	85.5	72.86	100	85.93	... 384,405,406
RewardFacet.sol	1.67	0	10	1.52	... 234,235,246
SetterFacet.sol	83.44	75	79.55	83.89	... 573,574,657
XVSRewardsHelper.sol	94.12	80	100	95.45	79,108
contracts/Comptroller/Diamond/interfaces/	100	100	100	100	
IDiamondCut.sol	100	100	100	100	
IMarketFacet.sol	100	100	100	100	
IPolicyFacet.sol	100	100	100	100	
IRewardFacet.sol	100	100	100	100	
ISetterFacet.sol	100	100	100	100	
All files	79.2	63.87	84.09	79.74	

governance-contracts repository

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
RiskSteward/	97.37	86.96	100	96	
IRiskSteward.sol	100	100	100	100	
IRiskStewardReceiver.sol	100	100	100	100	
MarketCapsRiskSteward.sol	97.67	79.17	100	94.55	109,126,127
RiskStewardReceiver.sol	96.97	95.45	100	97.78	136
All files	98.40	92.32	100	97,67	

Changelog

- 2025-02-14 - Initial report
- 2025-02-17 - Final report

About Quantstamp

Timeliness of content

Notice of confidentiality

Links to other websites

Disclaimer

Quantstamp