Quantstamp

E-Mode Core Pool (Venus)

Executive Summary

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

Type	Lending Protocol
Timeline	2025-07-28 through 2025-09-03
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Diff/Fork information	This code was based on the same repository at commit `174670c`.
Source Code	• VenusProtocol/venus-protocol #ce3eb96
Auditors	• Rabib Islam Senior Auditing Engineer • Mustafa Hasan Senior Auditing Engineer • Andrei Stefan Auditing Engineer

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

Summary of Findings

We have performed a diff audit of the Venus Protocol, primarily focusing on a new Efficiency Mode (e-mode) feature.

The feature allows the creation of isolated lending pools for groups of correlated assets. Users are to be able to opt-in to a specific e-mode category, gaining more favorable risk parameters like increased collateral factors and liquidation thresholds. Governance retains control over the creation and management of these pools. The feature is integrated into the `Comptroller`'s Diamond architecture, particularly within the `MarketFacet`. Safety checks prevent users from entering an e-mode with incompatible outstanding debts or an under-collateralized position.

No severe issues were found during the audit. The documentation is of high quality, including functional requirements for the e-mode feature, and the test suite has been bolstered and altered to accommodate the e-mode feature.

Fix-Review Update 2025-09-09:
VENE-1 has been acknowledged and the suggestions have been addressed. Some new testing was added for the e-mode feature.

ID	DESCRIPTION	SEVERITY	STATUS
VENE-1	Mismatched Function Name	• Informational	Acknowledged

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 focuses on a set of contracts modified in PR 614.

Files Included

- contracts/Comptroller/Diamond/facets/FacetBase.sol
- contracts/Comptroller/Diamond/facets/MarketFacet.sol
- contracts/Comptroller/Diamond/facets/PolicyFacet.sol
- contracts/Comptroller/Diamond/facets/RewardFacet.sol
- contracts/Comptroller/Diamond/facets/SetterFacet.sol
- contracts/Comptroller/Diamond/interfaces/IFacetBase.sol
- contracts/Comptroller/Diamond/interfaces/IMarketFacet.sol
- contracts/Comptroller/Diamond/interfaces/ISetterFacet.sol
- contracts/Comptroller/Diamond/Diamond.sol
- contracts/Comptroller/ComptrollerInterface.sol
- contracts/Comptroller/ComptrollerLensInterface.sol
- contracts/Comptroller/ComptrollerStorage.sol
- contracts/Comptroller/Types/PoolMarketId.sol
- contracts/InterfacesV8.sol
- contracts/Lens/ComptrollerLens.sol
- contracts/Lens/VenusLens.sol
- contracts/Liquidator/Liquidator.sol
- contracts/Tokens/VAI/VAIController.sol
- contracts/Tokens/VTokens/VToken.sol
- contracts/Utils/ErrorReporter.sol

Operational Considerations

Complexity of Risk Parameter Management

The per-market risk parameters within each e-mode group provide great flexibility but also increase the complexity of risk management for the Venus governance. Misconfiguration of these parameters for a single asset could introduce unforeseen risks within an e-mode group.

Liquidation Logic with Mixed Collateral

The specification notes a scenario where a liquidator can seize collateral from outside the user's e-mode group. In such cases, the liquidation penalty of the seized asset from the Core Pool is used.

Gas Costs for Users with Many Assets

The `hasValidPoolBorrows()` function iterates through all of a user's entered markets. For "whales" or users with a large number of supplied and borrowed assets, the gas cost of entering an e-mode could be substantial.

Key Actors And Their Capabilities

A few new functions were added that carry access restrictions:

- `createPool()` for creating e-mode pools;
- `removePoolMarket()` for removing markets from e-mode pools;
- `_addPoolMarket()` for adding markets to e-mode pools;
- `setIsBorrowAllowed()` for determining whether a `vToken` pertains to an e-mode pool.

Some other access control gates were altered in order to accommodate structural changes as a result of e-mode incorporation.

Findings

VENE-1 Mismatched Function Name

Informational | Acknowledged

Update

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

```
setCollateralFactor() does not have a name that explicitly reflects its ability to also set the liquidation threshold, but both are co-related, this behavior is documented in the comments, and this design choice was made to align with the Isolated Interface.
```

File(s) affected: `SetterFacet.sol`

Description: The functions `setCollateralFactor()` and `__setCollateralFactor()` are not only involved in setting a market's collateral factor, but also in setting its liquidation threshold. Hence, the function should be named differently, so as to avoid user confusion.

Recommendation: Consider renaming the function something more accurate and consistent with the rest of the codebase, such as `setLTVFactors()`.

Auditor Suggestions

S1 Gas Optimizations

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `c753de668d72d498cccea17801531112d0c1491a`.
The client provided the following explanation:

```
The _poolMarkets mapping is currently read twice—once in the modifier and once in the function which is fine.
  getMarketsDataByPool() may return an empty array, but the label still comes directly from comptroller.pools(i). Since this is a view function, the gas cost is not a concern.
```

We have identified the first two points as acknowledged and the third as fixed.

Description: We identified a few instances where we believe gas usage can be optimized.

1. In `SetterFacet.__setLiquidationIncentive()`, between the use of `compareValue` and the function's logic, the value of `_poolMarkets[getPoolMarketIndex(poolId, vToken)].liquidationIncentiveMantissa` is being loaded twice from storage, whereas it need only be loaded once.

2. In `VenusLens.getAllPoolsData()`, the function returns a `PoolWithMarkets` struct, where the `label` field is included alongside the `markets` field. However, the latter field also contains the `label` in `markets[0].label`. Hence, the function would require less gas if the value were simply referenced once, either by removing the `label` field or by pre-cacheing `getMarketsDataByPool(i, comptroller)`
3. `++i` is more gas-efficient than `i++` for the purpose of incrementing `for` loops.

Recommendation: We recommend making the relevant adjustments.

S2 Documentation Corrections

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `c753de668d72d498cccea17801531112d0c1491a`.

Description: We found areas in the documentation that can be corrected to enhance readability and maintainability of the repository.

1. `SetterFacet`
    1. In `setLiquidationIncentive()`, `vToken` is undocumented
    2. In `__setCollateralFactor()`, `newLiquidationThresholdMantissa` is undocumented
2. `VenusLens`
    1. In `getMarketsDataByPool()`, the arguments are documented out of order.
3. `MarketFacet`
    1. In `getLiquidationParams()`, we have a `maxLiquidationIncentiveMantissa`, whereas in other parts of the codebase, it is simply referred to as a `liquidationIncentiveMantissa`; one term should be used universally.

Recommendation: We recommend making the relevant corrections.

S3 Missing Input Validation

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `d8756733dd249f0c9c16f9ab46b454413b68669f`.
> The client provided the following explanation:
>
> `createPool() is governance controlled and it will be ensured that unique label names are used.`
>
> We identify the first point as fixed and the second point as acknowledged.

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Consider the following instances:

1. The function `addPoolMarkets()` validates that the lengths of the `poolIds` and `vTokens` arrays are equal, however it lacks a check that the lengths are not equal to zero.
2. The function `createPool()` takes a string that represents the name of the pool to be created. Pool names are not validated so they are unique, allowing the creation of multiple pools with the same name, which may confuse protocol users.

Recommendation: We recommend adding the relevant checks.

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 pose 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).

Test Suite Results

Tests were added to interact with the e-mode feature, such as creating pools, entering pool markets, etc.

Update: Some changes/additions have been made to the e-mode tests.

```
VBNBAdmin
  ✔ set VBNBAdmin as vBNB admin
  harvest income
    ✔ reduce BNB reserves (41ms)
  set interest rate model
    ✔ setInterestRateModel (38ms)

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 (62ms)

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
    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 (71ms)
    ✔ adds to the asset list only once (123ms)
    ✔ the market must be listed for add to succeed (67ms)
    ✔ returns a list of codes mapping to user's ultimate membership in given addresses (64ms)
  exitMarket
    ✔ doesn't let you exit if you have a borrow balance (99ms)
    ✔ rejects unless redeem allowed (216ms)
    ✔ accepts when you're not in the market already (99ms)
    ✔ properly removes when there's only one asset (166ms)
    ✔ properly removes when there's only two assets, removing the first (230ms)
    ✔ properly removes when there's only two assets, removing the second (216ms)
    ✔ properly removes when there's only three assets, removing the first (281ms)
    ✔ properly removes when there's only three assets, removing the second (267ms)
    ✔ properly removes when there's only three assets, removing the third (252ms)
```

```
  entering from borrowAllowed
    ✔ enters when called by a vtoken (92ms)
    ✔ reverts when called by not a vtoken
    ✔ adds to the asset list only once (116ms)
  unlistMarkets
    ✔ properly emits events and unlist market (130ms)
    ✔ reverts when unlisting not a listed market (108ms)

Comptroller
  constructor
    ✔ on success it sets admin to creator and pendingAdmin is unset (1403ms)
  setLiquidationIncentive
    ✔ fails if incentive is less than 1e18
    ✔ accepts a valid incentive and emits a NewLiquidationIncentive event (41ms)
    ✔ 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
  _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 (46ms)
    ✔ 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 (47ms)
    ✔ emits IsForcedLiquidationEnabledForUserUpdated event
  _supportMarket
    ✔ fails if asset is not a VToken
    ✔ succeeds and sets market (52ms)
    ✔ cannot list a market a second time (96ms)
    ✔ can list two different markets (61ms)
  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 (51ms)
    ✔ reverts if market is not listed
  redeemVerify
    ✔ should allow you to redeem 0 underlying for 0 tokens
    ✔ should allow you to redeem 5 underlyig for 5 tokens
    ✔ 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 (96ms)
      ✔ reverts if collateral market is not listed (80ms)
      ✔ does not revert if borrowed vToken is VAIController (108ms)
      ✔ 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 (46ms)
    Forced liquidations enabled for entire market
      ✔ reverts if borrowed market is not listed (69ms)
      ✔ reverts if collateral market is not listed (44ms)
      ✔ does not revert if borrowed vToken is VAIController (85ms)
      ✔ 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 (80ms)
    Forced liquidations disabled
      ✔ reverts if borrowed market is not listed (51ms)
      ✔ reverts if collateral market is not listed
      ✔ does not revert if borrowed vToken is VAIController
      ✔ fails if borrower has 0 shortfall (40ms)
      ✔ succeeds if borrower has nonzero shortfall
  borrow
    ✔ allows borrowing if cap is not reached (66ms)
    ✔ reverts borrowing if borrow cap is reached (70ms)
    ✔ reverts borrowing if borrow cap is 0 (71ms)
0x991C36261967d2500B905690C80e53Fb6870888f
    ✔ getBorrowingPower is an alias for getAccountLiquidity
  E-Mode Pool
  createPool
    ✔ reverts if label is empty
    ✔ should increment poolId and stores label (95ms)
  addPoolMarkets
    ✔ reverts if array lengths mismatch
    ✔ reverts if trying to add to core pool (poolId 0)
    ✔ reverts if pool does not exist
    ✔ reverts if market Already exist in the pool
    ✔ reverts if market not listed in core pool
    ✔ should add multiple markets (150ms)
  setIsBorrowAllowed
    ✔ reverts if pool does not exist
    ✔ reverts if market is not listed in the pool
    ✔ should return silenty if borrowAllowed is already set to desired value (52ms)
    ✔ should update borrowAllowed and emits event (43ms)
  setCollateralFactor for specific poolId
    ✔ reverts if pool does not exist
    ✔ reverts if market is not listed in the pool
    ✔ reverts on invalid parameter bounds (75ms)
    ✔ should update collateral factor and liquidation threshold and emits event (45ms)
  setLiquidationIncentive with poolId
    ✔ reverts if pool does not exist
    ✔ reverts if market is not listed in the pool
    ✔ reverts on invalid parameter bounds
    ✔ should update liquidation incentive and emits event
  removePoolMarket
    ✔ reverts if pool does not exist
    ✔ reverts if market is not listed in the pool
    ✔ removes the market and emits event (76ms)
    ✔ should delete pool vTokens array if last market removed
  poolEnter
    ✔ reverts if entering the wrong pool
    ✔ reverts if entering the same pool
    ✔ reverts if user has invalid pool borrows (66ms)
```

```
      ✔ should emit PoolSelected on successful pool switch (55ms)
    effective risk params
      ✔ should return emode params if market included in the emode category, else falls back to core
(169ms)
    Pool isActive Status
      ✔ reverts if pool does not exist
      ✔ reverts if tries to set for core pool
      ✔ should return silenty if isActive is already set to desired value
      ✔ should update isActive and emits event (39ms)
    Market Getters
      ✔ returns correct key for core pool
      ✔ returns correct core pool market info using markets()
      ✔ returns correct pool market info using poolMarkets() (46ms)
      ✔ returns all the markets of the specific pool (117ms)

  Comptroller
    ✔ Revert on check for the function selector (44ms)
    ✔ Add Facet and function selectors to proxy (89ms)
    ✔ 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 (61ms)
    ✔ Replace the function from facet mapping (82ms)
    ✔ Remove all functions (71ms)

  Comptroller
    liquidateCalculateAmountSeize
      ✔ fails if borrowed asset price is 0
      ✔ fails if collateral asset price is 0
      ✔ fails if the repayAmount causes overflow  (42ms)
      ✔ 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
(52ms)
      ✔ returns the correct value for
200000000000000000,100000000000000000,100000000000000000,100000000000000000,100000000000000000
(50ms)
      ✔ returns the correct value for
200000000000000000,200000000000000000,142000000000000000,130000000000000000,245000000000000000
(51ms)
      ✔ returns the correct value for
278900000000000000,523048084200000000,771320000000000000000,130000000000000000,1.000245e+22 (48ms)
      ✔ returns the correct value for
7.009232529961056e+24,2.5278726317240445e+24,2.6177112093242585e+23,1179713989619784000,7.790468414639561
e+24 (51ms)
      ✔ returns the correct value for
9.417034645351715e+24,5.831106681034997e+24,8.980407126656101e+24,1103931383598152600,6.662485090369619e+
23 (50ms)

  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 (39ms)
      ✔ can pause several actions on one market (45ms)
      ✔ can pause and unpause several actions on several markets (138ms)

  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 (114ms)
    ✔ fails if comptrollers don't match (55ms)
    ✔ fails if repayBorrowBehalf returns a non-zero error code (41ms)
    ✔ fails if borrowBehalf returns a non-zero error code (81ms)
    ✔ transfers repayAmount of vTokenToRepay.underlying() from the sender (92ms)
    ✔ approves vToken to transfer money from the contract (96ms)
    ✔ calls repayBorrowBehalf after transferring the underlying to self (94ms)
    ✔ converts the amounts using the oracle exchange rates (102ms)
    ✔ uses the actually repaid amount rather than specified amount (99ms)
    ✔ transfers the actually borrowed amount to the owner (106ms)
  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 (72ms)
    ✔ fails if repayBorrowBehalf returns a non-zero error code (57ms)
    ✔ fails if borrowBehalf returns a non-zero error code (106ms)
    ✔ transfers repayAmount of underlying from the sender (123ms)
    ✔ approves vToken to transfer money from the contract (126ms)
    ✔ calls repayBorrowBehalf after transferring the underlying to self (131ms)
    ✔ converts the amounts using the oracle exchange rates (134ms)
    ✔ uses the actually repaid amount rather than specified amount (133ms)
    ✔ transfers the actually borrowed amount to the owner (135ms)
  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 (1062ms)

  BUSDLiquidator
    setLiquidatorShare
      ✔ should set liquidator share
      ✔ should emit NewLiquidatorShare event
      ✔ should revert if caller is not owner
      ✔ should revert if new liquidator share is > MANTISA_ONE
    liquidateEntireBorrow
      ✔ should repay entire borrow (884ms)
      ✔ should seize collateral and split correctly between liquidator and treasury (971ms)
    liquidateBorrow
      ✔ should repay a part of the borrow (896ms)
      ✔ should seize collateral correctly for partial repay and split between liquidator and treasury
(1025ms)

  TokenRedeemer
    redeemAndTransfer
      ✔ should fail if called by a non-owner
      ✔ should fail if redeem fails (47ms)
      ✔ should succeed with zero amount (139ms)
      ✔ should redeem all vTokens (219ms)
      ✔ should transfer all underlying to the receiver (211ms)
    redeemUnderlyingAndTransfer
      ✔ should fail if called by a non-owner
      ✔ should revert if redeemer does not have vToken balance (112ms)
      ✔ should redeem and transfer successfully (314ms)
    redeemUnderlyingAndRepayBorrowBehalf
      ✔ should revert if redeemer does not have vToken balance (94ms)
      ✔ should redeem and repay successfully (731ms)
    redeemAndBatchRepay
      Generic
```

```
    ✔ fails if called by a non-owner
  Full repayment
    Native asset
      ✔ redeems just the required amount of vTokens (358ms)
      ✔ repays all borrows in full (425ms)
      ✔ transfers the excess vTokens to the receiver (356ms)
      ✔ transfers the excess BNB to the receiver (388ms)
    Tokens
      ✔ redeems just the required amount of vTokens (559ms)
      ✔ repays up to specified caps (532ms)
      ✔ repays all borrows in full (547ms)
      ✔ transfers the excess vTokens to the receiver (514ms)
      ✔ transfers the excess underlying to the receiver (513ms)
  Partial repayment
    Native asset
      ✔ redeems all available vTokens, up to 1 vToken wei (308ms)
      ✔ repays the three borrows: [in full, partially, no repayment] (369ms)
      ✔ uses the excess BNB to repay the debt in full (498ms)
      ✔ does not keep any vBNB or BNB balance (394ms)
    Tokens
      ✔ redeems all available vTokens, up to 1 vToken wei (439ms)
      ✔ repays the three borrows: [in full, partially, no repayment] (524ms)
      ✔ uses the excess underlying to repay the debt in full (554ms)
      ✔ does not keep any vToken or underlying balance (525ms)
    batchRepayVAI
      ✔ fails if called by a non-owner
      ✔ repays one borrow successfully (396ms)
      ✔ repays multiple borrows successfully and transfers refund to treasury (995ms)
      ✔ repays up to caps (983ms)
      ✔ partially repays borrows if insufficient VAI (887ms)
      ✔ can repay small amounts without failure (1111ms)
    sweepTokens
      ✔ fails if called by a non-owner
      ✔ sweeps tokens to destination if called by owner (64ms)
      ✔ sweeps native asset to destination (39ms)

  Two Kinks Interest Rate Model Tests
    ✔ Utilization rate: borrows is zero
    ✔ Utilization rate
    ✔ Borrow Rate: below kink1 utilization (43ms)
    ✔ Borrow Rate: above kink1 and below kink2 utilization (54ms)
    ✔ Borrow Rate: above kink2 utilization (62ms)
    ✔ Borrow Rate: above kink2 utilization and negative multipliers (93ms)
    ✔ Supply Rate

  VenusLens: Rewards Summary
    ✔ Should get summary for all markets (297ms)

    Liquidator
      splitLiquidationIncentive
        ✔ splits liquidationIncentive between Treasury and Liquidator with correct amounts
      distributeLiquidationIncentive
        ✔ distributes the liquidationIncentive between Treasury and Liquidator with correct amounts (76ms)
        ✔ reverts if transfer to liquidator fails
        ✔ reverts if underlying transfer to protocol share reserves fails (69ms)

    Liquidator
      liquidateBorrow
        liquidating BEP-20 debt
network block skew detected; skipping block events (emitted=2651 blockNumber3658)
          ✔ fails if borrower is zero address
          ✔ fails if some BNB is sent along with the transaction (58ms)
          ✔ transfers the seized collateral to liquidator and protocolShareReserve (193ms)
          ✔ transfers tokens from the liquidator (222ms)
          ✔ approves the borrowed VToken to spend underlying (191ms)
          ✔ calls liquidateBorrow on borrowed VToken (190ms)
          ✔ emits LiquidateBorrowedTokens event (207ms)
        liquidating VAI debt
          ✔ transfers VAI from the liquidator (192ms)
          ✔ approves VAIController to spend VAI (174ms)
          ✔ calls liquidateVAI on VAIController (167ms)
      liquidating BNB debt
```

```
      ✔ fails if msg.value is not equal to repayment amount (116ms)
      ✔ transfers BNB from the liquidator (135ms)
      ✔ calls liquidateBorrow on VBNB (129ms)
      - forwards BNB to VBNB contract
    setTreasuryPercent
      ✔ updates treasury percent in storage (45ms)
      ✔ fails when permission is not granted
      ✔ fails when the percentage is too high
      ✔ uses the new treasury percent during distributions (223ms)
    Force VAI Liquidation
      ✔ Should able to liquidate any token when VAI debt is lower than minLiquidatableVAI (137ms)
      ✔ Should not able to liquidate any token when VAI debt is greater than minLiquidatableVAI (49ms)
      ✔ 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 (170ms)
      ✔ Should able to liquidate any token and VAI token when force Liquidation is off (221ms)

  Liquidator
    Restricted liquidations
      addToAllowlist
        ✔ fails if not allowed to call
        ✔ adds address to allowlist (43ms)
        ✔ fails if already in the allowlist (48ms)
        ✔ emits LiquidationPermissionGranted event
      removeFromAllowlist
        ✔ fails if not allowed to call
        ✔ fails if not in the allowlist
        ✔ removes address from allowlist (80ms)
        ✔ emits LiquidationPermissionRevoked event (53ms)
      restrictLiquidation
        ✔ fails if not allowed to call
        ✔ restricts liquidations for the borrower (39ms)
        ✔ fails if already restricted (64ms)
        ✔ emits LiquidationRestricted event
      unrestrictLiquidation
        ✔ fails if not allowed to call
        ✔ removes the restrictions for the borrower (81ms)
        ✔ fails if not restricted
        ✔ emits LiquidationRestricted event (49ms)
      liquidateBorrow
        ✔ fails if the liquidation is restricted (49ms)
        ✔ proceeds with the liquidation if the guy is allowed to (76ms)

  PrimeScenario Token
    setMaxLoopsLimit()
Warning: Potentially unsafe deployment of
contracts/Tokens/Prime/PrimeLiquidityProvider.sol:PrimeLiquidityProvider

    You are using the `unsafeAllow.internal-function-storage` flag.
    Internal functions are code pointers which will no longer be valid after an upgrade.
    Make sure you reassign internal functions in storage variables during upgrades.

Warning: Potentially unsafe deployment of contracts/test/PrimeScenario.sol:PrimeScenario

    You are using the `unsafeAllow.internal-function-storage` flag.
    Internal functions are code pointers which will no longer be valid after an upgrade.
    Make sure you reassign internal functions in storage variables during upgrades.

      ✔ Revert when maxLoopsLimit setter is called by non-owner
      ✔ Revert when new loops limit is less than old limit
      ✔ maxLoopsLimit setter success (45ms)
    protocol setup
      ✔ markets added
      ✔ borrow balance
      ✔ get markets in prime
    mint and burn
      ✔ stake and mint (468ms)
      ✔ stake and unstake (319ms)
      ✔ stake manually (309ms)
      ✔ burn revocable token (858ms)
      ✔ cannot burn irrevocable token (777ms)
      ✔ manually burn irrevocable token (620ms)
```

```
      ✔ issue (785ms)
      ✔ upgrade (589ms)
      ✔ stake, issue and unstake (1086ms)
network block skew detected; skipping block events (emitted=3731 blockNumber8643727)
network block skew detected; skipping block events (emitted=3731 blockNumber8643727)
network block skew detected; skipping block events (emitted=3724 blockNumber8643727)
      ✔ issue, stake and burn (973ms)
    boosted yield
network block skew detected; skipping block events (emitted=3724 blockNumber7779728)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
network block skew detected; skipping block events (emitted=3724 blockNumber7779729)
      ✔ calculate score (175ms)
network block skew detected; skipping block events (emitted=3726 blockNumber7779729)
network block skew detected; skipping block events (emitted=3726 blockNumber7779736)
network block skew detected; skipping block events (emitted=3726 blockNumber7779736)
network block skew detected; skipping block events (emitted=3726 blockNumber7779736)
network block skew detected; skipping block events (emitted=3726 blockNumber7779736)
      ✔ accrue interest - prime token minted after market is added (582ms)
      ✔ claim interest (364ms)
      update score
        ✔ add existing market after issuing prime tokens - update score gradually (991ms)
        ✔ add existing market after issuing prime tokens - update score manually (1840ms)
    PLP integration
      ✔ claim interest (553ms)
      ✔ APR Estimation (112ms)
      ✔ Hypothetical APR Estimation (376ms)

  PrimeLiquidityProvider: tests
    Testing all initalized values
Warning: Potentially unsafe deployment of
contracts/Tokens/Prime/PrimeLiquidityProvider.sol:PrimeLiquidityProvider

    You are using the `unsafeAllow.internal-function-storage` flag.
    Internal functions are code pointers which will no longer be valid after an upgrade.
    Make sure you reassign internal functions in storage variables during upgrades.

      ✔ Tokens intialized
      ✔ Distribution Speed
    Testing all setters
      ✔ Revert on invalid args for initializeTokens
      ✔ Revert on re-intializing token
      ✔ initializeTokens success
      ✔ pauseFundsTransfer (45ms)
      ✔ resumeFundsTransfer (67ms)
      ✔ Revert on invalid args for setTokensDistributionSpeed
      ✔ Revert on non initialized token
      ✔ Revert on invalid distribution speed for setTokensDistributionSpeed (72ms)
      ✔ setTokensDistributionSpeed success with default max speed (73ms)
      ✔ setTokensDistributionSpeed success (82ms)
      ✔ 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 (76ms)
      ✔ Accrue amount for multiple tokens (513ms)
    Release funds to prime contract
      ✔ Revert on funds transfer Paused (46ms)
      ✔ Revert on invalid caller
      ✔ Release funds success (91ms)
    Sweep token
      ✔ Revert on insufficient balance
```

```
    ✔  Sweep token success (63ms)

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  (46ms)
  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 (62ms)
    ✔ 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 (48ms)
    ✔ should swap BNB -> token (65ms)
    ✔ 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  (93ms)
    ✔ 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 (127ms)
    ✔ swap BNB -> token --> supply token (137ms)
    ✔ revert if deadline has passed at supporting fee
    ✔ swap tokenA -> tokenB --> supply tokenB at supporting fee (127ms)
    ✔ swap BNB -> token --> supply token at supporting fee (132ms)
    ✔ swap tokenA -> exact tokenB (121ms)
    ✔ swap bnb -> exact tokenB (130ms)
    ✔ Exact tokens -> BNB and supply
    ✔ Exact tokens -> BNB and supply at supporting fee
  Repay
    ✔ revert if deadline has passed
    ✔ swap tokenA -> tokenB --> supply tokenB (124ms)
    ✔ swap BNB -> token --> supply token (126ms)
    ✔ revert if deadline has passed at supporting fee
    ✔ swap tokenA -> tokenB --> reapy tokenB at supporting fee (127ms)
    ✔ swap BNB -> token --> repay token at supporting fee (128ms)
    ✔ swap tokenA -> exact tokenB (125ms)
    ✔ swap tokenA -> full debt of tokenB (129ms)
    ✔ swap bnb -> exact tokenB (130ms)
    ✔ swap bnb -> full tokenB debt (144ms)
    ✔ Exact tokens -> BNB at supporting fee (94ms)
    ✔ Exact tokens -> BNB (78ms)
    ✔ Tokens -> Exact BNB (72ms)
    ✔ Tokens -> Exact BNB and supply
    ✔ Tokens -> full debt of BNB
  Sweep Token
    ✔ Should be reverted if get zero address
    ✔ Sweep ERC-20 tokens (84ms)
  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 (45ms)
    ✔ should emit event
  _acceptAdmin()
    ✔ should fail when pending admin is zero
    ✔ should fail when called by another account (e.g. root) (43ms)
    ✔ should succeed and set admin and clear pending admin (46ms)
    ✔ should emit log on success

Unitroller
  constructor
    ✔ sets admin to caller and addresses to 0 (39ms)
  _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

CheckpointView tests (using interest rate models as data sources)
  ✔ should revert if dataSource1 address is zero
  ✔ should revert if dataSource2 address is zero
  ✔ should use old rate model before checkpoint (52ms)
  ✔ should use new rate model after checkpoint (49ms)
  ✔ should return the correct current data source

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 (45ms)
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  (41ms)
    ✔ should revert if VAI transfer fails  (52ms)
    ✔ should revert if VAI to be burnt > vaiMinted  (41ms)
    should sucessfully perform the swap
      Fees: 10%
        ✔ stable token = 1$  (80ms)
        ✔ stable token < 1$  (73ms)
        ✔ stable token > 1$  (74ms)
      Fees: 0%
        ✔ stable token = 1$  (60ms)
        ✔ stable token < 1$  (60ms)
        ✔ stable token > 1$  (65ms)
    swapStableForVAI(address,uint256)
      ✔ should revert if receiver is zero address
      ✔ should revert if VAI mint cap will be reached  (62ms)
      ✔ should revert if amount after transfer is too small   (60ms)
      should sucessfully perform the swap
        Fees: 10%
          ✔ stable token = 1$  (80ms)
          ✔ stable token > 1$  (83ms)
          ✔ stable token < 1$  (80ms)
        Fees: 0%
          ✔ stable token = 1$  (74ms)
          ✔ stable token > 1$  (75ms)
          ✔ stable token < 1$  (75ms)
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
    ✔ 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 (49ms)
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  (50ms)
    ✔ should revert if VAI transfer fails  (117ms)
    ✔ should revert if VAI to be burnt > vaiMinted  (47ms)
    should sucessfully perform the swap
      Fees: 10%
        ✔ stable token = 1$  (82ms)
        ✔ stable token < 1$  (85ms)
        ✔ stable token > 1$  (81ms)
      Fees: 0%
        ✔ stable token = 1$  (70ms)
        ✔ stable token < 1$  (72ms)
        ✔ stable token > 1$  (72ms)
    swapStableForVAI(address,uint256)
      ✔ should revert if receiver is zero address
      ✔ should revert if VAI mint cap will be reached  (73ms)
      should sucessfully perform the swap
        Fees: 10%
          ✔ stable token = 1$  (94ms)
          ✔ stable token > 1$  (93ms)
          ✔ stable token < 1$  (93ms)
        Fees: 0%
          ✔ stable token = 1$  (85ms)
          ✔ stable token > 1$  (84ms)
          ✔ stable token < 1$  (87ms)
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
      ✔ 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 (53ms)
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  (54ms)
    ✔ should revert if VAI transfer fails  (67ms)
    ✔ should revert if VAI to be burnt > vaiMinted  (53ms)
    should sucessfully perform the swap
      Fees: 10%
        ✔ stable token = 1$  (97ms)
        ✔ stable token < 1$  (98ms)
        ✔ stable token > 1$  (97ms)
      Fees: 0%
        ✔ stable token = 1$  (82ms)
        ✔ stable token < 1$  (84ms)
        ✔ stable token > 1$  (86ms)
  swapStableForVAI(address,uint256)
      ✔ should revert if receiver is zero address
      ✔ should revert if VAI mint cap will be reached  (79ms)
      should sucessfully perform the swap
        Fees: 10%
          ✔ stable token = 1$  (107ms)
          ✔ stable token > 1$  (107ms)
          ✔ stable token < 1$  (106ms)
        Fees: 0%
          ✔ stable token = 1$  (97ms)
          ✔ stable token > 1$  (96ms)
          ✔ stable token < 1$  (95ms)

VAIController
  ✔ check wallet usdt balance
  #getMintableVAI
    ✔ oracle
    ✔ getAssetsIn
    ✔ getAccountSnapshot
    ✔ getUnderlyingPrice (42ms)
    ✔ getComtroller
    ✔ success (158ms)
  #mintVAI
    ✔ success (310ms)
    ✔ fails if there's not enough collateral (252ms)
    ✔ fails if minting beyond mint cap (374ms)
    ✔ fails if can't set the minted amount in comptroller (259ms)
    ✔ puts previously accrued interest to pastInterest (668ms)
  #repayVAI
    ✔ reverts if the protocol is paused
    ✔ success for zero rate (188ms)
    ✔ success for 1.2 rate repay all (249ms)
    ✔ success for 1.2 rate repay half (250ms)
    ✔ fails if can't set the new minted amount in comptroller (176ms)
  #repayVAIBehalf
    ✔ reverts if called with borrower = zero address
    ✔ reverts if the protocol is paused
    ✔ success for zero rate (185ms)
    ✔ success for 1.2 rate repay all (265ms)
    ✔ success for 1.2 rate repay half (254ms)
  #getHypotheticalAccountLiquidity
    ✔ success for zero rate 0.9 vusdt collateralFactor (274ms)
    ✔ success for 1.2 rate 0.9 vusdt collateralFactor (352ms)
  #liquidateVAI
    ✔ liquidationIncentiveMantissa
    ✔ reverts if the protocol is paused
    ✔ success for zero rate 0.2 vusdt collateralFactor (851ms)
network block skew detected; skipping block events (emitted=7780098 blockNumber100000000)
network block skew detected; skipping block events (emitted=7780098 blockNumber100000000)
```

```
network block skew detected; skipping block events (emitted=8643727 blockNumber100000000)
network block skew detected; skipping block events (emitted=8643727 blockNumber100000000)
network block skew detected; skipping block events (emitted=7780098 blockNumber100000000)
network block skew detected; skipping block events (emitted=7780098 blockNumber100000000)
network block skew detected; skipping block events (emitted=8643727 blockNumber100000000)
network block skew detected; skipping block events (emitted=7780098 blockNumber100000000)
      ✔ success for 1.2 rate 0.3 vusdt collateralFactor (969ms)
    #getVAIRepayRate
      ✔ success for zero baseRate
      ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 1e18 (151ms)
      ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 0.5 * 1e18 (156ms)
    #getVAIRepayAmount
      ✔ reverts if the protocol is paused (51ms)
      ✔ success for zero rate
      ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 1e18 (199ms)
      ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 0.5 * 1e18 (198ms)
    #getVAICalculateRepayAmount
      ✔ success for zero rate (57ms)
      ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 1e18 (307ms)
      ✔ success for baseRate 0.1 floatRate 0.1 vaiPirce 0.5 * 1e18 (324ms)
    #getMintableVAI
      ✔ include current interest when calculating mintable VAI (306ms)
    #accrueVAIInterest
      ✔ success for called once (115ms)
      ✔ success for called twice (167ms)
    #setBaseRate
      ✔ fails if access control does not allow the call
      ✔ emits NewVAIBaseRate event
      ✔ sets new base rate in storage
    #setFloatRate
      ✔ fails if access control does not allow the call
      ✔ emits NewVAIFloatRate event
      ✔ sets new float rate in storage
    #setMintCap
      ✔ fails if access control does not allow the call
      ✔ emits NewVAIMintCap event
      ✔ sets new mint cap in storage
    #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 (46ms)
      ✔ sets ACM address in storage (39ms)
    #prime
      ✔ prime integration (2076ms)

  VAIVault
    ✔ claim reward (578ms)
    setVenusInfo
      ✔ fails if called by a non-admin
      ✔ fails if XVS address is zero
      ✔ fails if VAI address is zero
      ✔ disallows configuring tokens twice

  VRTVault
    unit tests
      setLastAccruingBlock
        ✔ fails if ACM disallows the call
        ✔ fails if trying to set lastAccuringBlock to some absurdly high value
        ✔ fails if lastAccuringBlock has passed (60ms)
        ✔ fails if trying to set lastAccuringBlock to some past block
        ✔ fails if trying to set lastAccuringBlock to the current block
        ✔ correctly sets lastAccuringBlock to some future block (50ms)
        ✔ can move lastAccuringBlock to a later block (112ms)
        ✔ can move lastAccuringBlock to an earlier block (81ms)
        ✔ fails if trying to move lastAccuringBlock to a block in the past (63ms)
    scenario
      ✔ deposit (105ms)
```

```
      ✔ should claim reward (69ms)
      ✔ should not claim reward after certain block (107ms)

  VToken
    _setReserveFactorFresh
network block skew detected; skipping block events (emitted=7780094 blockNumber7790238)
network block skew detected; skipping block events (emitted=7780094 blockNumber7790238)
network block skew detected; skipping block events (emitted=7780094 blockNumber7790238)
network block skew detected; skipping block events (emitted=7780094 blockNumber7790239)
network block skew detected; skipping block events (emitted=7780094 blockNumber7790239)
network block skew detected; skipping block events (emitted=7780094 blockNumber7790239)
network block skew detected; skipping block events (emitted=7780094 blockNumber7790239)
      ✔ rejects change by non-admin
network block skew detected; skipping block events (emitted=7780112 blockNumber7790279)
      ✔ rejects change if market not fresh
      ✔ rejects newReserveFactor that descales to 1 (72ms)
      ✔ accepts newReserveFactor in valid range and emits log (90ms)
      ✔ accepts a change back to zero (155ms)
    _setReserveFactor
      ✔ emits a reserve factor failure if interest accrual fails (100ms)
      ✔ returns error from setReserveFactorFresh without emitting any extra logs (86ms)
      ✔ returns success from setReserveFactorFresh (118ms)
    _reduceReservesFresh
      ✔ fails if called by non-admin (47ms)
      ✔ fails if market not fresh (49ms)
      ✔ fails if amount exceeds available cash (393ms)
      ✔ if there isn't enough cash, reduces with available cash (181ms)
      ✔ increases admin balance and reduces reserves on success (205ms)
    _reduceReserves
      ✔ emits a reserve-reduction failure if interest accrual fails (86ms)
      ✔ returns error from _reduceReservesFresh without emitting any extra logs (169ms)
      ✔ returns success code from _reduceReservesFresh and reduces the correct amount (169ms)

  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
      ✔ reverts if xvsStore is not set
      ✔ reverts if a pool with this (staked token, reward token) combination already exists (46ms)
      ✔ reverts if staked token exists in another pool
      ✔ reverts if reward token is a zero address
      ✔ reverts if staked token is a zero address
      ✔ reverts if alloc points parameter is zero
      ✔ emits PoolAdded event (50ms)
      ✔ adds a second pool to an existing rewardToken (66ms)
      ✔ sets pool info (63ms)
      ✔ configures reward token in XVSStore (64ms)
    set
      ✔ reverts if ACM does not allow the call
      ✔ reverts if pool is not found
      ✔ reverts if total alloc points after the call is zero (46ms)
      ✔ succeeds if the pool alloc points is zero but total alloc points is nonzero (185ms)
      ✔ emits PoolUpdated event (50ms)
    setRewardAmountPerBlockOrSecond
      ✔ reverts if ACM does not allow the call
      ✔ reverts if the token is not configured in XVSStore (58ms)
      ✔ emits RewardAmountPerBlockUpdated event (65ms)
      ✔ updates reward amount per block (78ms)
    setWithdrawalLockingPeriod
      ✔ reverts if ACM does not allow the call
      ✔ reverts if pool does not exist
      ✔ reverts if the lock period is 0
      ✔ reverts if the lock period is absurdly high
      ✔ emits WithdrawalLockingPeriodUpdated event (67ms)
      ✔ updates lock period (99ms)
    pendingReward
      ✔ includes the old withdrawal requests in the rewards computation (200ms)
      ✔ excludes the new withdrawal requests from the rewards computation (286ms)
    deposit
```

```
      ✔ reverts if the vault is paused (46ms)
      ✔ reverts if pool does not exist
      ✔ transfers pool token to the vault (92ms)
      ✔ updates user's balance (92ms)
      ✔ fails if there's a pre-upgrade withdrawal request (134ms)
      ✔ succeeds if the pre-upgrade withdrawal request has been executed (422ms)
      ✔ uses the safe _transferReward under the hood (250ms)
    executeWithdrawal
      ✔ fails if the vault is paused (46ms)
      ✔ only transfers the requested amount for post-upgrade requests (259ms)
      ✔ handles pre-upgrade withdrawal requests (264ms)
      ✔ handles pre-upgrade and post-upgrade withdrawal requests (419ms)
    requestWithdrawal
      ✔ fails if the vault is paused (45ms)
      ✔ transfers rewards to the user (235ms)
      ✔ uses the safe _transferReward under the hood (244ms)
      ✔ fails if there's a pre-upgrade withdrawal request (114ms)
    claim
      ✔ fails if there's a pre-upgrade withdrawal request (56ms)
      ✔ succeeds if the pre-upgrade withdrawal request has been executed (267ms)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
network block skew detected; skipping block events (emitted=7790813 blockNumber7791817)
      ✔ excludes pending withdrawals from the user's shares (318ms)
      ✔ correctly accounts for updates in reward per block (205ms)
      ✔ uses the safe _transferReward under the hood (147ms)
    _transferReward
      ✔ sends the available funds to the user (122ms)
      ✔ emits VaultDebtUpdated event if vault debt is updated (86ms)
      ✔ does not emit VaultDebtUpdated event if vault debt is not updated (93ms)
      ✔ records the pending transfer (98ms)
      ✔ records several pending transfers (199ms)
      ✔ sends out the pending transfers in addition to reward if full amount <= funds available (329ms)
      ✔ sends a part of the pending transfers and reward if full amount > funds available (316ms)
    pendingWithdrawalsBeforeUpgrade
      ✔ returns zero if there were no pending withdrawals
      ✔ returns zero if there is only a new-style pending withdrawal (128ms)
      ✔ returns the requested amount if there is an old-style pending withdrawal (46ms)
      ✔ returns the total requested amount if there are multiple old-style pending withdrawals (56ms)
      ✔ returns zero if the pending withdrawal was executed (147ms)
    Scenarios
      ✔ works correctly with multiple claim, deposit, and withdrawal requests (1000ms)

  Prime Token
    mint and burn
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7791817 blockNumber7794627)
network block skew detected; skipping block events (emitted=7790816 blockNumber7794634)
Warning: Potentially unsafe deployment of
contracts/Tokens/Prime/PrimeLiquidityProvider.sol:PrimeLiquidityProvider

    You are using the `unsafeAllow.internal-function-storage` flag.
    Internal functions are code pointers which will no longer be valid after an upgrade.
    Make sure you reassign internal functions in storage variables during upgrades.

Warning: Potentially unsafe deployment of contracts/Tokens/Prime/Prime.sol:Prime

    You are using the `unsafeAllow.internal-function-storage` flag.
    Internal functions are code pointers which will no longer be valid after an upgrade.
    Make sure you reassign internal functions in storage variables during upgrades.

      ✔ should alias setPrimeToken to _setPrimeToken
      ✔ stake and mint (1040ms)
```

```
network block skew detected; skipping block events (emitted=7794634 blockNumber15570683)
      ✔ burn revocable token (2401ms)
      ✔ cannot burn irrevocable token (2227ms)
      ✔ issue and stake token concurrently (1697ms)
    boosted yield
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=7794681 blockNumber15570696)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346709)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
network block skew detected; skipping block events (emitted=15570696 blockNumber23346711)
      ✔ claim interest for multiple users (6324ms)

  755 passing (8m)
  1 pending
```

Code Coverage

Code coverage was obtained using `yarn coverage`. We note that overall branch coverage of the repository is not great. We recommend increasing repository-wide coverage of contracts to over 90%.

Update: Overall coverage data has changed only slightly.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
InterfacesV8.sol	100	100	100	100	
contracts/Admin/	90.48	40.91	85.71	88.46	
VBNBAdmin.sol	90.48	40.91	85.71	88.46	71,72,73
VBNBAdminStorage.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	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/Comptroller/Diamond/facets/	79.64	67.96	83.97	79.91	
FacetBase.sol	65.31	55.88	88.89	62.26	... 133,222,235
MarketFacet.sol	96.2	71.15	90.91	95.74	... 451,452,598
PolicyFacet.sol	85.61	72.22	100	85.4	... 416,417,538
RewardFacet.sol	1.67	0	10	1.52	... 243,244,255
SetterFacet.sol	88.07	82.5	84.78	88.04	... 609,611,612
XVSRewardsHelper.sol	94.12	80	100	95.45	79,108
contracts/Comptroller/Diamond/interfaces/	100	100	100	100	
IDiamondCut.sol	100	100	100	100	
IFacetBase.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	
contracts/Comptroller/Types/	100	100	100	100	
PoolMarketId.sol	100	100	100	100	
contracts/DelegateBorrowers/	100	89.47	100	100	
MoveDebtDelegate.sol	100	91.67	100	100	
SwapDebtDelegate.sol	100	85.71	100	100	
contracts/Governance/	73.15	44.87	68.18	68	
TokenRedeemer.sol	97.53	70	100	91.4	... 169,176,180
VTreasury.sol	0	0	0	0	... 65,67,70,72
VTreasuryV8.sol	0	0	0	0	... 90,91,98,99
contracts/InterestRateModels/	72.55	59.09	64.29	74.03	
InterestRateModel.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
InterestRateModelV8.sol	100	100	100	100	
JumpRateModel.sol	71.43	100	75	78.95	114,115,116,117
TwoKinksInterestRateModel.sol	100	56.25	100	93.33	100,104,173
WhitePaperInterestRateModel.sol	0	0	0	0	... 88,89,90,91
contracts/Lens/	42.41	33.87	37.93	42.13	
ComptrollerLens.sol	88.37	68.18	100	91.07	... 195,239,253
SnapshotLens.sol	0	0	0	0	... 122,124,146
VenusLens.sol	33.33	17.65	26.32	34.36	... 622,632,654
contracts/Liquidator/	84.05	59.52	86.05	83.25	
BUSDLiquidator.sol	97.56	58.33	91.67	98.04	99
Liquidator.sol	79.51	59.72	83.87	78.48	... 514,515,516
LiquidatorStorage.sol	100	100	100	100	
contracts/Oracle/	100	100	100	100	
PriceOracle.sol	100	100	100	100	
contracts/PegStability/	87.91	84.48	85	88.1	
IVAI.sol	100	100	100	100	
PegStability.sol	87.91	84.48	85	88.1	... 424,425,428
contracts/Swap/	92.68	57.14	96.49	87.38	
IRouterHelper.sol	100	100	100	100	
RouterHelper.sol	98.75	70.83	100	92.52	... 308,312,323
SwapRouter.sol	89.76	54.13	95.35	84.65	... 942,943,944
contracts/Swap/interfaces/	100	100	100	100	
CustomErrors.sol	100	100	100	100	
IPancakePair.sol	100	100	100	100	
IPancakeSwapV2Factory.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IPancakeSwapV2Router.sol	100	100	100	100	
IVBNB.sol	100	100	100	100	
IVtoken.sol	100	100	100	100	
IWBNB.sol	100	100	100	100	
InterfaceComptroller.sol	100	100	100	100	
contracts/Swap/lib/	100	52.63	100	81.03	
PancakeLibrary.sol	100	50	100	82.61	... 121,141,167
TransferHelper.sol	100	62.5	100	75	18,33,62
contracts/Tokens/	100	100	100	100	
EIP20Interface.sol	100	100	100	100	
EIP20NonStandardInterface.sol	100	100	100	100	
contracts/Tokens/Prime/	95.61	71.59	95.65	96.38	
IPrime.sol	100	100	100	100	
IPrimeV5.sol	100	100	100	100	
Prime.sol	94.87	68.94	95.83	96.35	... 4,1024,1133
PrimeLiquidityProvider.sol	97.65	79.55	95.24	96.49	123,215,309,351
PrimeLiquidityProviderStorage.sol	100	100	100	100	
PrimeStorage.sol	100	100	100	100	
contracts/Tokens/Prime/Interfaces/	100	100	100	100	
IPoolRegistry.sol	100	100	100	100	
IPrime.sol	100	100	100	100	
IPrimeLiquidityProvider.sol	100	100	100	100	
IVToken.sol	100	100	100	100	
IXVSVault.sol	100	100	100	100	
InterfaceComptroller.sol	100	100	100	100	
contracts/Tokens/Prime/libs/	90.38	75.76	100	89.86	
FixedMath.sol	100	50	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
FixedMath0x.sol	88.24	76	100	88.52	... 211,217,223
Scores.sol	90	90	100	100	
contracts/Tokens/VAI/	78.78	52.7	85.96	81.5	
IVAI.sol	100	100	100	100	
VAI.sol	57.69	30	66.67	65.85	... 156,157,158
VAIController.sol	85.02	59.2	97.14	87.21	... 580,581,583
VAIControllerInterface.sol	100	100	100	100	
VAIControllerStorage.sol	100	100	100	100	
VAIUnitroller.sol	44	25	66.67	48.48	... 123,124,126
lib.sol	100	100	100	100	
contracts/Tokens/VRT/	21.32	8.87	25.64	19.66	
VRT.sol	36.71	18.97	52.63	38.04	... 305,306,309
VRTConverter.sol	0	0	0	0	... 153,158,159
VRTConverterProxy.sol	0	0	0	0	... 168,178,180
VRTConverterStorage.sol	100	100	100	100	
contracts/Tokens/VTokens/	61.39	47.29	53.6	64.01	
VBNB.sol	0	0	0	0	... 180,181,183
VBep20.sol	63.33	0	62.5	64.52	... 140,141,181
VBep20Delegate.sol	50	25	66.67	42.86	29,40,41,44
VBep20Delegator.sol	26.92	50	26.32	32.14	... 460,498,501
VBep20Immutable.sol	100	100	100	100	
VToken.sol	72.75	51.3	80	74.57	... 0,1671,1676
VTokenInterfaces.sol	100	100	100	100	
contracts/Tokens/VTokens/legacy/	0	0	0	0	
ComptrollerInterface.sol	100	100	100	100	
IProtocolShareReserveV5.sol	100	100	100	100	
VBep20DelegateR1.sol	0	0	0	0	... 30,38,39,42

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
VBep20DelegatorR1.sol	0	0	0	0	... 522,523,528
VBep20R1.sol	0	0	0	0	... 259,275,284
VTokenInterfaceR1.sol	100	100	100	100	
VTokenR1.sol	0	0	0	0	... 3,1687,1691
VTokenStorageR1.sol	100	100	100	100	
contracts/Tokens/VTokens/legacy/Utils/	0	0	0	0	
CarefulMath.sol	0	0	0	0	... 76,78,79,82
ErrorReporter.sol	0	100	0	0	... 272,279,281
Exponential.sol	0	0	0	0	... 168,170,179
ExponentialNoError.sol	0	0	0	0	... 188,189,193
contracts/Tokens/XVS/	19.08	8.46	23.26	18.52	
IXVS.sol	100	100	100	100	
IXVSVesting.sol	100	100	100	100	
XVS.sol	36.71	18.97	52.63	38.04	... 305,306,309
XVSVesting.sol	0	0	0	0	... 218,223,224
XVSVestingProxy.sol	0	0	0	0	... 151,161,163
XVSVestingStorage.sol	100	100	100	100	
contracts/Utils/	51.07	30.43	50.91	52.21	
Address.sol	42.86	0	33.33	50	44,66,70,71
CarefulMath.sol	80	66.67	100	84	35,46,77,88
CheckpointView.sol	100	100	100	100	
Context.sol	0	100	0	0	19,23,24
ECDSA.sol	0	0	0	0	... 6,97,98,101
ErrorReporter.sol	50	100	50	50	... 320,327,329
Exponential.sol	54	40.91	53.85	54	... 169,171,180
ExponentialNoError.sol	76.09	62.5	67.65	76.09	... 173,177,181

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IBEP20.sol	100	100	100	100	
Ownable.sol	0	0	0	0	... 63,70,71,72
Owned.sol	0	0	33.33	20	13,14,18,19
SafeBEP20.sol	53.85	33.33	50	53.85	... 41,42,46,50
SafeCast.sol	8.33	4.17	8.33	8.33	... 193,204,205
SafeMath.sol	85	58.33	77.78	85	145,160,161
Tokenlock.sol	33.33	16.67	33.33	33.33	18,20,24,26
contracts/VAIVault/	45.95	45.16	51.85	50.49	
VAIVault.sol	75.56	51.85	73.68	78.79	... 217,218,236
VAIVaultErrorReporter.sol	0	100	0	0	26,28,35,37
VAIVaultProxy.sol	0	0	0	0	... 125,135,137
VAIVaultStorage.sol	100	100	100	100	
contracts/VRTVault/	47.66	36.29	53.33	48.59	
VRTVault.sol	62.2	40.18	72.73	64.49	... 277,304,305
VRTVaultProxy.sol	0	0	0	0	... 144,154,156
VRTVaultStorage.sol	100	100	100	100	
contracts/XVSVault/	60.67	50	55.71	63.37	
XVSStore.sol	60	46.15	66.67	60.71	... 1,93,94,125
XVSVault.sol	67.73	53.13	62.26	71.2	... 868,921,922
XVSVaultErrorReporter.sol	0	100	0	0	26,28,35,37
XVSVaultProxy.sol	0	0	0	0	... 125,135,137
XVSVaultStorage.sol	100	100	100	100	
contracts/external/	100	100	100	100	
IProtocolShareReserve.sol	100	100	100	100	
IWBNB.sol	100	100	100	100	
contracts/lib/	100	71.43	100	88.89	
Currency.sol	100	90	100	92.86	58

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
approveOrRevert.sol	100	25	100	75	25
All files	60.23	45.44	57.73	61.54	

Changelog

- 2025-09-05 - Initial report
- 2025-09-12 - 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

identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or 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 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.