Quantstamp

Time-Based Contracts (Venus)

Executive Summary

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

Type	DeFi
Timeline	2024-03-04 through 2024-03-08
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• VenusProtocol/isolated-pools #1a6cf0c • VenusProtocol/isolated-pools #fb32967 • VenusProtocol/oracle #d8af595 • VenusProtocol/venus-protocol #4d7debd • VenusProtocol/venus-protocol #9cfeba7 • VenusProtocol/venus-protocol #2d31b70 • VenusProtocol/venus-protocol #96574ea
Auditors	• Julio Aguilar Auditing Engineer • Shih-Hung Wang Auditing Engineer • Cameron Biniamow Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	5 Fixed: 3 Acknowledged: 2
High severity findings	0
Medium severity findings	0
Low severity findings	1 Acknowledged: 1
Undetermined severity findings	0
Informational findings	4 Fixed: 3 Acknowledged: 1

Summary of Findings

The Venus protocol is expanding to various EVM-compatible chains, transitioning from a block number-based to a dual timing mechanism that supports both block numbers and timestamps. This is due to the inconsistency of block times across different chains. This audit focuses on verifying these adaptations, which are distributed across several pull requests in two repositories: `venus-protocol` and `isolated-pools`. Additionally, the expansion to layer 2 chains necessitates unique oracle adaptations for their sequencers, requiring the Venus team to enhance their `oracle` repository with new features. A notable update includes the ability of the protocol to confiscate tokens from specific users and redistribute them. The audit found no other major issues but identified several low and informational points.

Update: The Venus team addressed all issues by fixing or acknowledging them. The team also improved the test suite for all three repositories.

ID	DESCRIPTION	SEVERITY	STATUS
VTIME-1	Centralization Risk	● Low	Acknowledged
VTIME-2	Missing Input Validation	● Informational	Fixed
VTIME-3	Missing Check for the Configured Mode of Reward Distributors in `PoolLens`	● Informational	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
VTIME-4	Mismatch Between Function Name and String for Access Control	● Informational ⓘ	Fixed
VTIME-5	Misleading 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 spans across 7 Pull Requests from 3 different repositories.

Files Included

- **Repository:** https://github.com/VenusProtocol/isolated-pools
  - PR 324 - Isolated Lending:
  - **Commit Hash:** `1a6cf0c1aea867cef4657670fb989dbc0b47c91c`
  - **Files Audited:**
    - contracts/JumpRateModelV2.sol
    - contracts/Lens/PoolLens.sol
    - contracts/Rewards/RewardsDistributor.sol
    - contracts/Rewards/RewardsDistributorStorage.sol
    - contracts/Shortfall/Shortfall.sol
    - contracts/Shortfall/ShortfallStorage.sol
    - contracts/VToken.sol

        - contracts/VTokenInterfaces.sol
        - contracts/WhitePaperInterestRateModel.sol
        - contracts/lib/constants.sol
    - PR 337 - Reduce Reserves With Available Cash
    - **Commit Hash**: `fb32967ff93cac2ab007cfc941f9c42ee245dc2b`
    - **Files Audited**:
        - contracts/VToken.sol
- **Repository**: https://github.com/VenusProtocol/venus-protocol
    - PR 418 - Time-based XVSVault:
    - **Commit Hash**: `4d7debdc67141b754217e5286476404eefc5549a`
    - **Files Audited**:
        - contracts/XVSVault/TimeManagerV5.sol
        - contracts/XVSVault/XVSVault.sol
        - contracts/XVSVault/XVSVaultStorage.sol
    - PR 414 - Reduce Reserves With Available Cash
    - **Commit Hash**: `9cfeba718e68aa7294c9895c51037f9e9b81e450`
    - **Files Audited**:
        - contracts/Tokens/VTokens/VToken.sol
    - PR 417 - Miscellaneous: Seize XVS Rewards
    - **Commit Hash**: `2d31b70eb263fb16e31b67e413ad581c683a8bf8`
    - **Files Audited**:
        - contracts/Comptroller/Diamond/facets/RewardFacet.sol
    - PR 410 - Miscellaneous: Dynamically Set Addresses for XVS and XVSVToken
    - **Commit Hash**: `96574ea27041ce9b3935ec03f7b8fce540475ec4`
    - **Files Audited**:
        - contracts/Comptroller/ComptrollerStorage.sol
        - contracts/Comptroller/Diamond/Diamond.sol
        - contracts/Comptroller/Diamond/facets/FacetBase.sol
        - contracts/Comptroller/Diamond/facets/RewardFacet.sol
        - contracts/Comptroller/Diamond/facets/SetterFacet.sol
- **Repository**: https://github.com/VenusProtocol/oracle
    - PR 128 - Add Arbitrum Sequencer Downtime Validation for Chainlink Oracle
    - **Commit Hash**: `d8af5952e53bb5f1131b40d83bce0da3126dd0a7`
    - **Files Audited**:
        - contracts/oracles/SequencerChainlinkOracle.sol
        - contracts/oracles/ChainlinkOracle.sol

Findings

VTIME-1 Centralization Risk

● Low ⓘ Acknowledged

Update

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

> seizeVenus(address[],address) will be executable only by authorized address in the AccessControlManager contract deployed at 0x4788629abc6cfca10f9f969efdeaa1cf70c23555 Only the normal, fast-track and critical timelocks will be authorized to execute this function. So, it will be executed only via the Venus governance process, with the votes of the community.

File(s) affected: `contracts/Comptroller/Diamond/RewardFacet.sol`

Description: The new function added to the `RewardFacet` in the core protocol called `seizeVenus()` allows the owner of the protocol to seize tokens from a given number of holders and transfer them to a provided recipient. A compromised owner could drain the protocol.

Recommendation: We are aware of and appreciate the Venus team's diligent approach to security. Nonetheless, we would like to remind the team to keep ensuring that privileged accounts utilize a multi-sig or an equivalent mechanism, and to adhere to the latest key management practices to prevent any breaches. Furthermore, please make sure to communicate this new function to your users.

VTIME-2 Missing Input Validation

● Informational ⓘ Fixed

Update

Marked as "Fixed" by the client. Addressed in: `6ff148d8aa3a2d7adcce7ced0967bac8b5f1f9f8` , `d61191cb1a8c198deea6cc362425144efbc312f5` , `82f1e429aa887ba1fdd7787cfe77bee41e4a79f4` .

File(s) affected: `venus-protocol: contracts/Comptroller/Diamond/facets/SetterFacet.sol, isolated-pools: VToken.sol, SequencerChainlinkOracle.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following is a non-exhaustive list of missing input validations:

1. In the `SetterFacet` contract, the `_setXVSVToken()` function sets the VToken contract corresponding to the XVS token. As a best practice, there can be a check on the provided argument `xvsVToken_` to ensure that the underlying token of `xvsVToken_` matches the configured `xvs`.
2. The constructor of the VToken contract should validate that the value of `MAX_BORROW_RATE_MANTISSA` is not greater than `1e18`.
3. In the `constructor()` of the `SequencerChainlinkOracle` contract, check that sequencer is not set to the zero address.

Recommendation: Consider adding the corresponding checks.

VTIME-3 Missing Check for the Configured Mode of Reward Distributors in `PoolLens`

● Informational ⓘ Fixed

Update
Marked as "Fixed" by the client. Addressed in: `e57c00d7df5f84d4eb19e10d6cf2704d18b0fdff`.

File(s) affected: `isolated-pools: contracts/Lens/PoolLens.sol`

Description: In the `_calculateNotDistributedAwards()` function of the `PoolLens` contract, the rewards are calculated based on the mode (time-based or block-based) that the reward distributor is configured. Since all the reward distributor contracts deployed on the same chain should use the same mode as `PoolLens`, consider adding a check to ensure they are in the same mode to reduce the risk of misconfiguration during contract deployment.

Also, the local variable, `isTimeBased`, shadows a state variable inherited from the `TimeManagerV8`. It is generally recommended to avoid shadow variables as a best practice.

Recommendation: Consider ensuring the returned value from `rewardsDistributor.isTimeBased()` matches the state variable `isTimeBased` and renaming the local variable `isTimeBased`.

VTIME-4 Mismatch Between Function Name and String for Access Control

● Informational ⓘ Fixed

Update
Marked as "Fixed" by the client. Addressed in: `4626a1032510390627728a998b0aa19e8eaa3682`.

File(s) affected: `isolated-pools: contracts/Rewards/RewardsDistributor.sol`

Description: In the `setLastRewardingBlocks()` function of the `RewardsDistributor` contract, the input string to `_checkAccessAllowed()` is `setLastRewardingBlock` instead of `setLastRewardingBlocks` as the function name.

Recommendation: Consider modifying `setLastRewardingBlock` to `setLastRewardingBlocks`

VTIME-5 Misleading Function Name

● Informational ⓘ Acknowledged

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

> To maintain the backward compatibility. Maybe third parties are calling the public view rewardTokenAmountsPerBlock, so we want to maintain it in the contract

File(s) affected: `venus-protocol: XVSVault.sol`

Description: The function `rewardTokenAmountsPerBlock(address _rewardToken)` returns the value stored in the `rewardTokenAmountsPerBlockOrSecond` mapping at the given `_rewardToken`. If the value stored in the mapping is in seconds, a misleading value would be returned.

Recommendation: Since the `rewardTokenAmountsPerBlockOrSecond` mapping already has public visibility, consider removing the `rewardTokenAmountsPerBlock()` function and reference the `rewardTokenAmountsPerBlockOrSecond()` getter function instead.

Definitions

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

Appendix

File Signatures

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

Contracts

- `260...a16 ./TimeManagerV5.sol`
- `0e6...1a2 ./ComptrollerStorage.sol`
- `ac6...e80 ./XVSVault.sol`
- `b63...0ea ./XVSVaultStorage.sol`
- `71b...96c ./Diamond.sol`
- `ab9...5ae ./FacetBase.sol`
- `c49...66f ./RewardFacet.sol`
- `3c5...cf1 ./SetterFacet.sol`
- `fb3...592 ./SequencerChainlinkOracle.sol`
- `b80...d78 ./ChainlinkOracle.sol`
- `5c5...d85 ./contracts/VToken.sol`
- `ac6...d1c ./contracts/VTokenInterfaces.sol`
- `8f7...c16 ./contracts/JumpRateModelV2.sol`
- `0d2...429 ./contracts/WhitePaperInterestRateModel.sol`
- `da0...0e8 ./contracts/lib/constants.sol`
- `a9f...1ed ./contracts/Shortfall/Shortfall.sol`
- `f6a...005 ./contracts/Shortfall/ShortfallStorage.sol`
- `71e...9db ./contracts/Lens/PoolLens.sol`
- `355...492 ./contracts/Rewards/RewardsDistributor.sol`
- `6df...420 ./contracts/Rewards/RewardsDistributorStorage.sol`

Tests

- `751...54e ./PivotPythOracle.ts`
- `61f...67c ./PivotTwapOracle.ts`
- `ec2...6a4 ./SequencerChainlinkOracle.ts`
- `a88...8dc ./types.ts`
- `ccf...840 ./BoundValidator.ts`

- `5da...d9e ./ResilientOracle.ts`
- `703...62f ./BinanceOracle.ts`
- `e96...e47 ./ChainlinkOracle.ts`
- `65d...bdc ./data.ts`
- `779...550 ./makePair.ts`
- `b71...e8f ./makeToken.ts`
- `6c8...d85 ./makeVToken.ts`
- `7d1...77e ./makeChainlinkOracle.ts`
- `3c4...dde ./time.ts`
- `4be...566 ./validate-price-config.ts`
- `a27...d7b ./utils.ts`
- `fb2...f78 ./core-compatibility.ts`
- `049...965 ./FeeToken.sol`
- `b64...7ec ./ERC20.sol`
- `41f...08c ./VTokenHarness.sol`
- `676...572 ./HarnessMaxLoopsLimitHelper.sol`
- `c1d...f4e ./ComptrollerHarness.sol`
- `20e...e13 ./SafeMath.sol`
- `cdf...521 ./ComptrollerScenario.sol`
- `aed...560 ./UpgradedVToken.sol`
- `ec0...114 ./XVSVaultScenario.sol`
- `7f2...c0f ./XVSStoreScenario.sol`
- `175...97e ./PrimeScenario.sol`
- `398...b83 ./EvilToken.sol`
- `468...2e7 ./PrimeLiquidityProviderScenario.sol`
- `82a...afe ./FaucetToken.sol`
- `380...acd ./MockDeflationaryToken.sol`
- `b61...518 ./TokenDebtTrackerHarness.sol`
- `68c...8a3 ./ApproveOrRevertHarness.sol`
- `a4c...091 ./ProtocolShareReserve.sol`
- `841...868 ./MockToken.sol`
- `e5b...395 ./MockPriceOracle.sol`
- `a05...e93 ./MockPancakeSwap.sol`
- `a94...24b ./tests/hardhat/WhitePaperInterestRateModel.ts`
- `423...510 ./tests/hardhat/MaxLoopsLimitHelper.ts`
- `9e9...21c ./tests/hardhat/Rewards.ts`
- `7cc...fc4 ./tests/hardhat/JumpRateModelV2.ts`
- `fef...55c ./tests/hardhat/PoolRegistry.ts`
- `417...728 ./tests/hardhat/Shortfall.ts`
- `c26...8b5 ./tests/hardhat/UpgradedVToken.ts`
- `04c...fd7 ./tests/hardhat/Prime.ts`
- `d4e...ae1 ./tests/hardhat/AccessControl.ts`
- `975...430 ./tests/hardhat/Tokens/mintAndRedeemTest.ts`
- `071...f8d ./tests/hardhat/Tokens/liquidateTest.ts`
- `138...029 ./tests/hardhat/Tokens/reservesTest.ts`
- `77e...5be ./tests/hardhat/Tokens/transferTest.ts`
- `5bb...13d ./tests/hardhat/Tokens/accrueInterestTest.ts`
- `ead...1b2 ./tests/hardhat/Tokens/setters.ts`
- `832...bbd ./tests/hardhat/Tokens/borrowAndRepayTest.ts`
- `108...bb1 ./tests/hardhat/lib/TokenDebtTracker.ts`
- `bf6...cc4 ./tests/hardhat/lib/ApproveOrRevert.ts`
- `795...7cd ./tests/hardhat/Comptroller/pauseTest.ts`
- `b7b...256 ./tests/hardhat/Comptroller/liquidateAccountTest.ts`
- `4ad...c19 ./tests/hardhat/Comptroller/assetsListTest.ts`
- `6f4...120 ./tests/hardhat/Comptroller/liquidateCalculateAmountSeizeTest.ts`

- `347...169 ./tests/hardhat/Comptroller/healAccountTest.ts`
- `3ea...bc7 ./tests/hardhat/Comptroller/setters.ts`
- `e6e...cbe ./tests/hardhat/Comptroller/hooks.ts`
- `08c...41b ./tests/hardhat/Comptroller/accountLiquidityTest.ts`
- `2e9...375 ./tests/hardhat/Lens/RewardsSummary.ts`
- `87c...1ec ./tests/hardhat/Lens/PoolLens.ts`
- `cd3...7c7 ./tests/hardhat/util/AddressOrContract.ts`
- `4e7...556 ./tests/hardhat/util/Errors.ts`
- `95f...f9d ./tests/hardhat/util/types.ts`
- `e3b...855 ./tests/hardhat/util/ComptrollerTestHelpers.ts`
- `cc3...e74 ./tests/hardhat/util/TokenTestHelpers.ts`
- `296...b50 ./tests/hardhat/util/descriptionHelpers.ts`
- `9f0...adc ./tests/hardhat/Fork/liquidation.ts`
- `73c...9d2 ./tests/hardhat/Fork/utils.ts`
- `4fc...df9 ./tests/hardhat/Fork/supply.ts`
- `c71...010 ./tests/hardhat/Fork/Shortfall.ts`
- `623...73d ./tests/hardhat/Fork/borrowAndRepayTest.ts`
- `13c...177 ./tests/hardhat/Fork/reduceReservesTest.ts`
- `bd8...600 ./tests/hardhat/Fork/RewardsForkTest.ts`
- `5fd...115 ./tests/hardhat/Fork/RiskFund.ts`
- `c86...588 ./tests/hardhat/Fork/RiskFundSwap.ts`
- `349...057 ./tests/integration/index.ts`

Toolset

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

Setup

Tool Setup:
- Slither v0.8.3

Steps taken to run the tools:
1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

All the Slither results were either identified as false positives or included in the findings of this report.

Test Suite Results

`yarn install & yarn test`

Since the test output is too large, we mention the number of passing tests for each repository and commit hash.

- Isolated Pools:
    - Commit hash: `1a6cf0c1aea867cef4657670fb989dbc0b47c91c`
        - 513 passing tests. No failing tests.
    - PR 337 initial commit hash: `fb32967ff93cac2ab007cfc941f9c42ee245dc2b`
        - 32 passing tests. 50 failing tests.
    - PR 337 test suite fix commit hash: `7ab9c1fc0e590722480ab8d429c2e2c4bf48605d`
        - 433 passing tests. No failing tests.
- Oracle:
    - Commit hash: `d8af5952e53bb5f1131b40d83bce0da3126dd0a7`
        - 81 passing tests. No failing tests.
- Core Venus Protocol:
    - PRs 410, 414 and 417 were merged into the develop branch to fix possible merge conflicts and to have matching file hashes with this audit report. Commit hash: `f8051fa1fb6ab714736cf307d5f57a6214855b38`.

        - 651 passing tests, no failing tests.
    - PR 418:
        - 634 passing tests, no failing tests.

Update: The team added more tests after addressing the issues in the report.

- Isolated Pools:
    - Commit hash: `d61191cb1a8c198deea6cc362425144efbc312f5`
        - 531 passing tests, no failing tests.
- Oracle:
    - Commit hash: `82f1e429aa887ba1fdd7787cfe77bee41e4a79f4`
        - 85 passing tests, no failing tests.
- Core Venus Protocol:
    - Commit hash: `d31397027faa99e4e7eab2f36a14b5beedb40c57`
        - 656 passing tests, no failing tests.

Code Coverage

Note that the full output of the coverage data was pruned to include only the relevant files in scope. Coverage in the isolated-pools repository is exceptionally high. We recommend increasing the coverage for all the relevant contracts in the venus-protocol and oracle repositories to at least 90%.

- Isolated Pools:
    - Commit hash: `1a6cf0c1aea867cef4657670fb989dbc0b47c91c`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
JumpRateModelV2.sol	100	70	100	94.29	98,167
VToken.sol	99.65	69.87	100	96.5	... 9,1324,1361
VTokenInterfaces.sol	100	100	100	100	
WhitePaperInterestRateModel.sol	100	75	100	94.12	109
PoolLens.sol	98.23	63.64	94.74	93.84	... 551,563,600
RewardsDistributor.sol	96.15	69.57	88.89	96.48	... 301,309,564
RewardsDistributorStorage.sol	100	100	100	100	
Shortfall.sol	100	85.29	100	100	
ShortfallStorage.sol	100	100	100	100	
constants.sol	100	100	100	100	

- Commit hash: `7ab9c1fc0e590722480ab8d429c2e2c4bf48605d`.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
VToken.sol	99.65	73.42	100	97.86	... 3,1274,1346

- Oracle:
    - Commit hash: `d8af5952e53bb5f1131b40d83bce0da3126dd0a7`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ChainlinkOracle.sol	72.97	53.85	88.89	85.71	... 86,87,88,89

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
SequencerChainlinkOracle.sol	100	100	100	100	

- Core Venus Protocol:
  - PRs 410, 414 and 417 were merged into the develop branch to fix possible merge conflicts and to have matching file hashes with this audit report. Commit hash: `f8051fa1fb6ab714736cf307d5f57a6214855b38`.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ComptrollerStorage.sol	100	100	100	100	
Diamond.sol	97.26	59.09	100	95.35	109,228,229,230
FacetBase.sol	62.22	55.88	86.67	59.18	... 128,211,224
RewardFacet.sol	1.67	0	10	1.52	... 234,235,246
SetterFacet.sol	87.69	80.36	88.46	87.42	... 598,600,601
IRewardFacet.sol	100	100	100	100	
ISetterFacet.sol	100	100	100	100	
VToken.sol	69.13	47.39	70.91	71.73	... 2,1643,1648
VTokenInterfaces.sol	100	100	100	100	

- PR 418:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
TimeManagerV5.sol	55.56	50	75	75	40,43,65
XVSVault.sol	68.27	53.13	63.46	71.66	... 859,865,866
XVSVaultStorage.sol	100	100	100	100	

Update: The team added more tests after addressing the issues in the report.

- Isolated Pools:
  - Commit hash: `d61191cb1a8c198deea6cc362425144efbc312f5`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
JumpRateModelV2.sol	100	70	100	94.29	98,167
VToken.sol	99.65	73.13	100	97.86	... 4,1295,1367
VTokenInterfaces.sol	100	100	100	100	
WhitePaperInterestRateModel.sol	100	75	100	94.12	109
PoolLens.sol	98.25	70.83	94.74	95.24	... 552,564,601

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
RewardsDistributor.sol	96.15	69.57	88.89	96.48	... 301,309,564
RewardsDistributorStorage.sol	100	100	100	100	
Shortfall.sol	100	85.29	100	100	
ShortfallStorage.sol	100	100	100	100	
constants.sol	100	100	100	100	

- Oracle:
    - Commit hash: `82f1e429aa887ba1fdd7787cfe77bee41e4a79f4`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ChainlinkOracle.sol	100	92.31	100	100	
SequencerChainlinkOracle.sol	100	87.5	100	100	

- Core Venus Protocol:
    - Commit hash: `d31397027faa99e4e7eab2f36a14b5beedb40c57`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ComptrollerStorage.sol	100	100	100	100	
Diamond.sol	97.26	59.09	100	95.35	109,228,229,230
FacetBase.sol	62.22	55.88	86.67	59.18	... 128,211,224
RewardFacet.sol	1.67	0	10	1.52	... 234,235,246
SetterFacet.sol	86.36	77.59	88.46	86.34	... 601,603,604
XVSVault.sol	68.42	53.16	64	71.8	... 836,842,843
XVSVaultStorage.sol	100	100	100	100	

Changelog

- 2024-03-08 - Initial report
- 2024-03-18 - 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