Quantstamp

Venus - Mutichain Governance

Executive Summary

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

Type	Governance
Timeline	2024-04-17 through 2024-04-29
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	multichain-governance
Source Code	• VenusProtocol/governance-contracts #266ae02
Auditors	• Julio Aguilar Auditing Engineer • Jennifer Wu Auditing Engineer • Nikita Belenkov Auditing Engineer

Documentation quality	High
Test quality	Medium
Total Findings	9 Fixed: 3 Acknowledged: 6
High severity findings	0
Medium severity findings	0
Low severity findings	4 Fixed: 1 Acknowledged: 3
Undetermined severity findings	0
Informational findings	5 Fixed: 2 Acknowledged: 3

Summary of Findings

The Venus protocol is expanding to include more networks, and is therefore creating a multichain governance system for proposal submissions and executions across these new networks. An `OmnichainProposalSender` on the BNB Chain facilitates the transmission of proposals to destination chains using LayerZero endpoints. Correspondingly, `OmnichainProposalExecutor` contracts deployed on the destination chains receive and time-lock transaction payloads from Venus Improvement Proposals (VIP), allowing execution based on defined priorities. These operations are managed through templates from LayerZero's omni-chain governance executor contracts. The system's non-blocking messaging allows for the local storage and future reattempt of failed transmissions due to issues like insufficient gas, with a daily limit imposed on the number of commands sent and received.

We commend the Venus team for their well-written code and excellent documentation. The test suite is comprehensive, but there is room for improvement in test coverage as highlighted in the test section. The audit team did not identify any major severity issue. We still advise addressing all 9 low and informational issues.

Fix-Review Update: The Venus team has addressed all the issues by fixing or acknowledging them.

ID	DESCRIPTION	SEVERITY	STATUS
VEN-1	**Possibility to Skip Proposals By Changing Timelock**	Low	Acknowledged
VEN-2	`timelocks_` **Entries Should Be Unique**	Low	Acknowledged
VEN-3	**It Is Possible to Have an Invalid Value Inside** `trustedRemoteLookup`	Low	Fixed
VEN-4	**The Internal State of** `BaseOmnichainControllerSrc` **Is Not Reverted After an Execution Failure**	Low	Acknowledged
VEN-5	**Critical Role Transfer Not Following Two-Step Pattern**	Informational	Acknowledged
VEN-6	**Possibility of Accepting Messages From Compromised Remotes**	Informational	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
VEN-7	Missing `_zroPaymentAddress` Specification	Informational	Fixed
VEN-8	Critical Governance Proposal Can Be Blocked by Daily Command Limits	Informational	Acknowledged
VEN-9	Proposal Execution Order Not Guaranteed in Venus Multichain Governance	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

Files Included

contracts/Cross-chain/BaseOmnichainControllerDest.sol
contracts/Cross-chain/BaseOmnichainControllerSrc.sol
contracts/Cross-chain/OmnichainExecutorOwner.sol
contracts/Cross-chain/OmnichainGovernanceExecutor.sol
contracts/Cross-chain/OmnichainProposalSender.sol
contracts/Cross-chain/interfaces/ITimelock.sol
contracts/Governance/TimelockV8.sol

Files Excluded

contracts/legacy/*
contracts/Utils/*
contracts/test/*
contracts/Governance/AccessControlledV5.sol
contracts/Governance/AccessControlledV8.sol
contracts/Governance/AccessControlManager.sol
contracts/Governance/GovernorBravoDelegate.sol
contracts/Governance/GovernorBravoDelegator.sol
contracts/Governance/GovernorBravoInterfaces.sol
contracts/Governance/IAccessControlManagerV5.sol
contracts/Governance/IAccessControlManagerV8.sol
contracts/Governance/Timelock.sol

Findings

VEN-1 Possibility to Skip Proposals By Changing Timelock

Low | Acknowledged

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

The addTimelock() function is overseen by the owner, referred to as OmnichainExecutorOwner, with its ownership managed by the Access Control Manager. Therefore, the responsibility to add timelocks lies with the Timelock, ensuring the secure addition of timelocks without malicious processes.

File(s) affected: `OmnichainGovernanceExecutor.sol`

Description: The `addTimelocks()` function allows the admin to update the timelock contracts. However, it does not check if the timelocks to be updated have queued transactions, which could lead to an inconsistent state where transactions can never be executed.

The usage of Timelock functions such as `cancelTransaction()` and `executeTransaction()` rely on the fact that the underlying timelock has not changed; if that is the case, the old queued transactions would need to be resent as a new proposal since the old ones could neither be canceled nor executed. However, this could make the whole process questionable for users since proposals can be silently canceled or skipped.

Recommendation: Verify that there are no queued transactions before allowing to update the timelock.

VEN-2 `timelocks_` Entries Should Be Unique

Low | Acknowledged

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

The addTimelock() function is overseen by the owner, referred to as OmnichainExecutorOwner, with its ownership managed by the Access Control Manager. Therefore, the responsibility to add timelocks lies with the Timelock, ensuring the secure addition of timelocks without malicious processes.

File(s) affected: `OmnichainGovernanceExecutor.sol`

Description: When a proposal is received from the layer zero endpoint, it is queued with the relevant timelock based on the proposal type. Timelock contracts can be updated via `addTimelocks()`, but there is no check that enforces them to be unique. Hence, the contract can use the same timelock for all 3 proposal types.

This could lead to issues where `proposalTimelocks[proposalType_].queuedTransactions` transactions will revert if the same transaction has been sent with multiple proposal types, for example, to get it expedited.

Recommendation: Consider enforcing the uniqueness of `timelocks_`.

VEN-3 It Is Possible to Have an Invalid Value Inside `trustedRemoteLookup`

Low | Fixed

Update
The length of the remote address is checked.

Update

Marked as "Fixed" by the client.
Addressed in: `4a7768b7f4f425948a772a6729b98460dcc3da89`.
The client provided the following explanation:

To maintain the consistency this check is included in both sender and executor contracts.

File(s) affected: `OmnichainProposalSender.sol`

Description: `trustedRemoteLookup` stores the allowed path for sending messages in the format of remote app address + local app address. There is no limitation on the size or structure of this mapping. The value field of the mapping is passed to `LZ_ENDPOINT.send()`, which expects as one of the parameters: remote address concatenated with local address packed into 40 bytes.

It is not guaranteed that the value in the `trustedRemoteLookup` mapping is 40 bytes, which could lead to unexpected behavior.

Recommendation: Consider enforcing the 40-byte size rule.

VEN-4
The Internal State of `BaseOmnichainControllerSrc` Is Not Reverted After an Execution Failure

Low | Acknowledged

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

We acknowledge this behaviour that count will be incremented whenever execute function of OmnichainProposalSender is called.

File(s) affected: `BaseOmnichainControllerSrc.sol`, `OmnichainProposalSender.sol`

Description: When a proposal is sent via the `execute()` method in the `OmnnichainProposalSender` contract, the `_validateProposal()` function increments the command count within a 24-hour window. If the proposal dispatch fails, this increment is not reverted, potentially obstructing future proposals in the same window. Moreover, retrying the execution could erroneously increase the command count twice if `_validateProposal()` is called again within the same period, further risking the execution of subsequent proposals.

Recommendation: Consider calling `_validateProposal()` inside the try-catch block before emitting the event `ExecuteRemoteProposal`.

VEN-5
Critical Role Transfer Not Following Two-Step Pattern

Informational | Acknowledged

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

That would be handled carefully; we don't want to make any modifications to LayerZero's current code because we inherit LayerZero's and LzApp is extended by Ownable.

File(s) affected: `OmnichainExecutorOwner.sol`

Description: The owner of the contract can call `transferOwnership()` to transfer the ownership to a new address. If an uncontrollable address is accidentally provided as the new owner address then the contract will no longer have an active owner, and functions with the `onlyOwner` modifier can no longer be executed.

Recommendation: Consider using OpenZeppelin's `Ownable2Step` contract to adopt a two-step ownership pattern in which the new owner must accept their position before the transfer is complete.

VEN-6
Possibility of Accepting Messages From Compromised Remotes

Informational | Fixed

Update
Trusted remote check has been added to `retryMessage()`.

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `70de9ffb550cafbe7b1d5cbc35dba8b24e89ede3`.

File(s) affected: `OmnichainGovernanceExecutor.sol`

Description: The `OmnichainGovernanceExecutor` contract is set up as a non-blocking app, which allows for the replaying of unsuccessful messages via `retryMessage()` which makes a subsequent call to `_nonblockingLzReceive()` that attempts to retry the failed message. This can be an issue if a trusted remote address is compromised and is removed as a trusted remote since neither of the above mentioned functions perform any validation checks on whether the address behind a `_srcChainId` is still a `trustedRemote`.

Recommendation: We don't see any major risk since there's only going to be a single remote - the `OmnichainProposalSender` - and the `retryMessage()` was overriden to only be called by the owner of the executor. However, for a more robust security, we recommend adding additional checks in the `retryMessage()` to validate the remote.

VEN-7 Missing `_zroPaymentAddress` Specification

Informational | Fixed

> **Update**
> In-line comment has been added.

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `ebe888c73e1ad9b77eea9bec1b2596c75a96cc6f`.

File(s) affected: `OmnichainProposalSender.sol`

Description: The `send()` function in `OmnichainProposalSender` is used to send proposal data to the LayerZero endpoint and includes the `_zroPaymentAddress` parameter. This parameter is required when the executor chooses to pay transaction fees using ZRO tokens instead of native currency. The address provided must meet the following conditions:

```
require(_zroPaymentAddress == ua || _zroPaymentAddress == tx.origin, "LayerZero: must be paid by sender or origin");
```

This requirement ensures that the transaction fees are paid either by the user application (ua, typically `OmnichainProposalSender`) or the transaction's origin (`tx.origin`).

Recommendation: The documentation for the parameter `_zroPaymentAddress` of the `send()` function should clearly state that if fees are paid with ZRO tokens, the tokens should be sent to `OmnichainProposalSender` before calling `execute()` or `_zroPaymentAddress` should be set to the transaction origin to avoid transaction failure. Add tests to ensure messages can be sent using ZRO tokens.

VEN-8
Critical Governance Proposal Can Be Blocked by Daily Command Limits

Informational | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> We will take care of this, as soon as the limit is reached to the lowest we will increase it with a proposal

File(s) affected: `OmnichainGovernanceExecutor.sol`, `OmnichainProposalSender.sol`

Description: The `OmnichainProposalSender` and `OmnichainGovernanceExecutor` include mechanisms to limit the number of commands that can be sent and received within 24 hours. These daily limits may prevent critical proposals from being executed promptly if the limit is reached, potentially delaying important governance actions.

Recommendation: The daily limit can be adjusted by the contract owner through the function `setMaxDailyReceiveLimit()`. If the daily limit is reached, governance can increase this limit for critical purposes.

VEN-9
Proposal Execution Order Not Guaranteed in Venus Multichain Governance

Informational | Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> We acknowledge this behaviour

Description: The execution order of proposals is not guaranteed in the Venus multichain governance. Due to the non-blocking nature of the message receiver, proposals sent across chains may not be executed in the order they were intended. This lack of a guaranteed execution order can be particularly problematic for interdependent or sequence-sensitive proposals. Although the scenario presented below is unlikely, it demonstrates the potential impact of out-of-order proposal execution.

Additionally, this misordering can be compounded by potential race conditions such as:

1. Calling `setSrcChainId()` will invalidate any proposals that are yet to be received from that chain or any proposals in the `failedMessages` mapping.
2. If `execute()` and `cancel()` are called at a similar time, the behavior of the proposal will be based on the ordering of the transactions in the block.

Exploit Scenario: 1. The client decides to list a new market on Arbitrum and divides the action into three proposals: * The first proposal lists the market. * The second proposal mints some market tokens to address an empty market vulnerability. * The third proposal sets the collateral factor.

2. All three proposals are accepted and executed through `OmnichainProposalSender.execute()`.
3. The proposals are received and queued in the following order:
    - The first proposal is received and queued first.
    - The second proposal fails due to insufficient gas and is stored in `failedMessages`.
    - The third proposal is received and queued next.
    - The second proposal is received through `retryMessage` and subsequently queued.
4. Due to the order of queuing, the first and third proposals are executed before the second proposal. This sequence of execution leaves the protocol vulnerable to an empty market vulnerability.

Recommendation: To mitigate risks associated with out-of-sequence execution, we recommend the client bundle interdependent proposals into the same proposal, ensuring they are processed as a unit. Also, consider updating the source chain id only if there are no failed messages to retry.

Definitions

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

Appendix

File Signatures

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

Files

- `762...758 ./TimelockV8.sol`
- `df8...584 ./Cross-chain/BaseOmnichainControllerDest.sol`
- `8bd...9f2 ./Cross-chain/OmnichainGovernanceExecutor.sol`
- `485...1cb ./Cross-chain/OmnichainExecutorOwner.sol`

- `eba...d31 ./Cross-chain/OmnichainProposalSender.sol`
- `fab...d6e ./Cross-chain/BaseOmnichainControllerSrc.sol`
- `939...bf2 ./Cross-chain/interfaces/ITimelock.sol`
- `c55...0af ./Cross-chain/interfaces/IOmnichainGovernanceExecutor.sol`

Tests

- `b7d...7aa ./AccessControl.ts`
- `554...cde ./intializeTests.ts`
- `ac5...497 ./timelock.ts`
- `a40...0b4 ./stateTest.ts`
- `d3f...eb8 ./queueTest.ts`
- `aa3...e51 ./castVoteTest.ts`
- `65b...20d ./proposeTest.ts`
- `716...ec2 ./Proposals.ts`
- `266...19d ./BSC.js`
- `546...4ff ./Omnichain.ts`
- `bfb...bd5 ./test/MockAccessTest.sol`
- `190...336 ./test/TestTimelockV8.sol`
- `f04...e78 ./test/MockXVSVault.sol`

Automated Analysis

N/A

Test Suite Results

All 100 tests pass successfully. To run the tests execute the following commands:

yarn install npx hardhat test

```
Compiled 68 Solidity files successfully (evm targets: istanbul, shanghai).


  Access Control
    Access Control
      ✔ only default admin role can give call permissions (79ms)
      ✔ should not have permissions
      ✔ should have permissions
      ✔ should revoke role
      ✔ should be able to call the function only for the given contract
      ✔ should be able to call the function on every contract

  Omnichain:
    ✔ Reverts if EOA called owner function of bridge
    ✔ Reverts if EOA call execute() without grant permission
    ✔ Reverts when zero value passed
    ✔ Revert if trusted remote is removed by non owner
    ✔ Revert if non trusted remote is removed
    ✔ Reverts when trusted remote is not set
    ✔ Reverts with Daily Transaction Limit Exceed
    ✔ Reverts if EOA call setMaxDailyLimit() without grant permisssion
    ✔ Revert if function in not found in function registry
    ✔ Reverts if any user other than owner try to add function in function registry
    ✔ Function registry should not emit event if nonexistant function is removed
    ✔ Function registry should not emit event if function is added twice
    ✔ Reverts if EOA called owner function of Executor
    ✔ Emit TimelocksAdded event (105ms)
    ✔ Emit SetTrustedRemoteAddress event (92ms)
    ✔ Emit ExecuteRemoteProposal event (50ms)
    ✔ Revert initially, success on retry (79ms)
    ✔ Revert when daily limit exceeds in retry (48ms)
```

```
    ✔ Emit ProposalExecuted event (73ms)
    ✔ Should update delay of timelock on destination (72ms)
    ✔ Admin can set the new pending admin of Timelock (124ms)
    ✔ Set new pending admin of Timelock through proposal (97ms)
    ✔ should revert when invalid proposalType is passed
    ✔ Revert when zero address passed as pending admin
    ✔ Revert when non owner sets the pending admin of Timelock (38ms)
    ✔ Revert if empty proposal
    ✔ Revert on invalid proposal type (40ms)
    ✔ Revert if same proposal come twice (97ms)
    ✔ Retry message on destination on failure (530ms)
    ✔ Retry messages that failed due to low gas at the destination using the Endpoint. (72ms)
    ✔ Reverts when other than guardian call cancel of executor (64ms)
    ✔ Revert if proposal is not queued
    ✔ Revert when proposal is not queued
    ✔ Emit ProposalCanceled event when proposal gets canceled (72ms)
    ✔ Reverts when cancel is called after execute (73ms)
    ✔ Proposal fails if any number of commands fail on destination
    ✔ Reverts when number of parameters mismatch
    ✔ Refund stucked gas in contract, to given address (39ms)
    ✔ Reverts on passing zero values in parameters in fallback withdraw (40ms)
    ✔ Reverts when value exceeds contract's balance in fallback withdraw
    ✔ Reverts when different parameters passed in fallback withdraw (40ms)
    ✔ Reverts when receiver is unable to receive in fallback withdraw
    ✔ Refund stucked gas in contract, to given address (38ms)
    ✔ Reverts on passing zero values in parameters in fallback withdraw (38ms)
    ✔ Reverts when different parameters passed in fallback withdraw (41ms)
    ✔ Reverts when receiver is unable to receive in fallback withdraw
    ✔ Reverts when daily limit of sending transaction reached
    ✔ Proposal failed when receiving limit reached (118ms)

  Governor Bravo Cast Vote Test
    We must revert if:
      ✔ We cannot propose without enough voting power by depositing xvs to the vault
      after we deposit xvs to the vault
        ✔ There does not exist a proposal with matching proposal id where the current block number is
between the proposal's start block (exclusive) and end block (inclusive)
        ✔ Such proposal already has an entry in its voters set matching the sender
        Otherwise
          ✔ we add the sender to the proposal's voters set
        and we take the balance returned by GetPriorVotes for the given sender and the proposal's start
block, which may be zero,
          ✔ and we add that ForVotes (46ms)
          ✔ or AgainstVotes corresponding to the caller's support flag. (44ms)
        castVoteBySig
          ✔ reverts if the signatory is invalid
          ✔ casts vote on behalf of the signatory (47ms)

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

  Governor Bravo Propose Tests
    simple initialization
      ✔ ID is set to a globally unique identifier
      ✔ Proposer is set to the sender
      ✔ Start block is set to the current block number plus vote delay
      ✔ End block is set to the current block number plus the sum of vote delay and vote period
      ✔ ForVotes and AgainstVotes are initialized to zero
      ✔ Executed and Canceled flags are initialized to false
      ✔ ETA is initialized to zero
      ✔ Targets, Values, Signatures, Calldatas are set according to parameters
      ✔ This function returns the id of the newly created proposal. # proposalId(n) = succ(proposalId(n-
1))
      ✔ emits log with id and description (51ms)
      This function must revert if
        ✔ the length of the values, signatures or calldatas arrays are not the same length, (41ms)
```

```
        ✔ or if that length is zero or greater than Max Operations.
        Additionally, if there exists a pending or active proposal from the same proposer, we must
revert.
          ✔ reverts with pending
          ✔ reverts with active

  Governor Bravo Queue Tests
    overlapping actions
      ✔ reverts on queueing overlapping actions in same proposal (53ms)
      ✔ reverts on queueing overlapping actions in different proposals (88ms)

  Governor Bravo State Tests
    ✔ Invalid for proposal not found
    ✔ Pending
    ✔ Active
    ✔ Canceled (38ms)
    ✔ Canceled by Guardian
    ✔ Defeated
    ✔ Succeeded (46ms)
    ✔ Expired (61ms)
    ✔ Queued (61ms)
    ✔ Executed (86ms)

  TimelockV8 Tests
    ✔ Production timelock returns constant values
    ✔ Production timelock requires setting appropriate delay
    ✔ Production timelock does not allow a null address
    ✔ Test Timelock returns 1 for constants
    ✔ Test Timelock allows setting low delay
    ✔ Test timelock does not allow a null address


  100 passing (8s)
```

Code Coverage

We recommend adding more tests to improve the robustness of the testing suite by increasing the branch and function coverage to at least 90%. To get the coverage report execute the following commands:

yarn install npx hardhat coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Cross-**chain/**	89.94	70.49	74.42	90.59	
BaseOmnichainControllerDest.sol	100	70	83.33	100	
BaseOmnichainControllerSrc.sol	66.67	70	50	73.08	... 84,94,95,96
OmnichainExecutorOwner.sol	81.82	65	66.67	80.77	79,80,95,96,97
OmnichainGovernanceExecutor.sol	98.33	71.43	91.67	97.5	153,154
OmnichainProposalSender.sol	90.2	72.5	72.73	91.23	... 290,291,301
Cross-**chain/interfaces/**	100	100	100	100	
IOmnichainGovernanceExecutor.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ITimelock.sol	100	100	100	100	
Governance/	94.59	55.26	91.67	89.58	
TimelockV8.sol	94.59	55.26	91.67	89.58	... 130,131,239
All files	90.82	66.88	78.18	90.4	

Fix-Review Update:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Cross-**chain/**	90.36	72.31	75	90.91	
BaseOmnichainControllerDest.sol	100	70	83.33	100	
BaseOmnichainControllerSrc.sol	66.67	70	50	73.08	... 84,94,95,96
OmnichainExecutorOwner.sol	85.19	70.83	71.43	83.87	... 111,112,113
OmnichainGovernanceExecutor.sol	98.36	72.73	91.67	97.53	153,154
OmnichainProposalSender.sol	90.38	73.81	72.73	91.38	... 291,292,302
Cross-**chain/interfaces/**	100	100	100	100	
IOmnichainGovernanceExecutor.sol	100	100	100	100	
ITimelock.sol	100	100	100	100	
Governance/	78.38	42.11	75	81.25	
TimelockV8.sol	78.38	42.11	75	81.25	... ,92,170,175
All files	88.18	65.48	75	89.11	

Changelog

- 2024-04-29 - Initial report
- 2024-05-10 - Final report

About Quantstamp

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

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

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked

with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

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

Notable Collaborations & Customers:

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

Timeliness of content

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

Notice of confidentiality

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

Links to other websites

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

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.