



November 2nd 2022 — Quantstamp Verified

csMATIC Liquid Staking on Polygon (ClayStack)

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










Executive Summary

| | |
|-----------------------|---|
| Type | Cross-chain Staking and Liquidity-balancing |
| Auditors | Ibrahim Abouzied, Auditing Engineer Poming Lee, Senior Research Engineer Rabib Islam, Research Engineer |
| Timeline | 2022-09-20 through 2022-10-21 |
| EVM | Paris |
| Languages | Solidity |
| Methods | Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review |
| Specification | None |
| Documentation Quality |  Medium |
| Test Quality |  High |
| Source Code | |

| Repository | Commit |
|---|--------------------------|
| ClayStack/claystack-matic-polygon | 0596bbc initial audit |
| ClayStack/claystack-matic-polygon | 63f3c29 fixes |

| | |
|---------------------------|------------------------|
| Total Issues | 12 (6 Resolved) |
| High Risk Issues | 0 (0 Resolved) |
| Medium Risk Issues | 0 (0 Resolved) |
| Low Risk Issues | 7 (5 Resolved) |
| Informational Risk Issues | 5 (1 Resolved) |
| Undetermined Risk Issues | 0 (0 Resolved) |



| | |
|---|---|
|  High Risk | The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users. |
|  Medium Risk | The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact. |
|  Low Risk | 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. |
|  Unresolved | Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it. |
|  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). |
|  Fixed | Adjusted program implementation, requirements or constraints to eliminate the risk. |
|  Mitigated | Implemented actions to minimize the impact or likelihood of the risk. |

Summary of Findings

ClayStack allows users to stake their MATIC on Polygon by aggregating user deposits and withdrawals before bridging them over to Ethereum, where the MATIC is staked via ClayMatic. Users are provided the derivative token csMATIC to later unstake their tokens on Polygon.

Generally speaking, the scope of this audit covers the issuance of the csMATIC token in Polygon, the bridging of MATIC from Polygon to Ethereum, and its integration with ClayMatic. ClayMatic, and its staking of MATIC through Polygon's validators, is not in the scope of this audit.

The most important issues we found relate to its reliance on outside components. With the complexity of integrating L1 and L2 systems, it is important that the protocol properly validates all inputs and guards against replayability. We found the test coverage to be strong for the Polygon contracts, but could use improvement on the Ethereum side.

| ID | Description | Severity | Status |
|--------|---|---------------|--------------|
| QSP-1 | Reliance on <code>clayTunnel</code> to Be Working as Expected | Low | Acknowledged |
| QSP-2 | Reliance on <code>fxRootTunnel</code> to Be Working as Expected | Low | Fixed |
| QSP-3 | Privileged Roles and Ownership | Low | Fixed |
| QSP-4 | Fees May Be Uninitialized | Low | Acknowledged |
| QSP-5 | Missing Input Validation | Low | Fixed |
| QSP-6 | Missing Initializers | Low | Fixed |
| QSP-7 | Unchecked Return Value | Low | Fixed |
| QSP-8 | Reliance on Off-Chain Component to Be Working as Expected | Informational | Acknowledged |
| QSP-9 | Block Timestamp Manipulation | Informational | Acknowledged |
| QSP-10 | Timelock Should Allow Adequate Time for Users to Exit | Informational | Acknowledged |
| QSP-11 | Upgradability | Informational | Acknowledged |
| QSP-12 | Unlocked Pragma | Informational | Fixed |

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

DISCLAIMER:

The audit was performed on the following files only:

- `src/ClaimVault.sol`
- `src/ClayManager.sol`
- `src/CsToken.sol`
- `src/UserProxy.sol`

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

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
 - i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. 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:
 - i. 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.
 - ii. 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, clarify, 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.

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 .`

Findings

QSP-1 Reliance on `clayTunnel` to Be Working as Expected

Severity: **Low Risk**

Status: Acknowledged

File(s) affected: `ClayManager.sol`

Description: The `ClayManager._exchangeCsToken()` for calculating how many MATIC can be exchanged from a certain amount of csMatic tokens. This procedure totally relies on the security, correctness, and stability of `clayTunnel.getFunds()`. An incorrect rate could lead to devastating consequences.

Recommendation: Add a sanity check to prevent the exchange rate obtained from being manipulated into a value with a large bias.

Update: The team has said that the `ClayTunnel.sol` contract has been previously audited and its failure would imply a Polygon level failure. Additionally, there aren't any straightforward constraints that can be applied to the exchange rate as there is no precedent regarding slashing and `ClayManager.sol` may see large price increases if it hasn't been run in a long time.

QSP-2 Reliance on `fxRootTunnel` to Be Working as Expected

Severity: Low Risk

Status: Fixed

File(s) affected: `ClayManager.sol`

Description: The `ClayManager` contract is responsible for completing the process of a batch order that is specified by `batchID` in polygon, using `ClayManager._processMessageFromRoot()`. This function is designed and expected to be called only once. However, these functions have no protection against replay attacks. Though `_processMessageFromRoot()` prevents the same `_stateId` from being used twice, it does not prevent the same `_batchId` from being passed into `_registerClaimProceeded()`. `_registerClaimProceeded()` being executed more than once for a `_batchId` is an unexpected condition and could cause negative consequences to the platform.

Recommendation: Add a check to `_registerClaimProceeded()` to make sure that a `_batchId` cannot be processed more than once.

Update: A check was added to `_registerClaimProceeded()` to confirm that the `batchId` has not been previously processed.

QSP-3 Privileged Roles and Ownership

Severity: Low Risk

Status: Fixed

File(s) affected: `ClayManager.sol`, `UserProxy.sol`

Description: Certain contracts have state variables, e.g. `TIMELOCK_UPGRADES_ROLE`, which provide certain addresses with privileged roles. Such roles may pose a risk to end-users. The `ClayManager` contract contains the following privileged roles:

- `onlyProxy` modifier-guarded functions
 - `.initialize()` -- initialize the contract's variables post-deployment.
 - `.upgradeTo()` -- upgrade the implementation of the proxy to a new contract.
- `onlyRole(CS_SERVICE_ROLE)` modifier-guarded functions
 - `.setBridgeFee()` -- set the fee to manually trigger the bridge by user in matic
 - `.setBridgeRequirements()` -- set the days and amounts required for bridge transfers
 - `.setMinClaimDays()` -- set the minimum number of days to claim tokens when netted locally
 - `.pause()` -- pause the contract
 - `.unpause()` -- unpause the contract
- `onlyRole(TIMELOCK_UPGRADES_ROLE)` modifier-guarded functions
 - `.setTreasury()` -- set the treasury wallet
 - `.setClaimVault()` -- set the vault to which MATIC tokens will be sent
 - `.upgradeTo()` -- upgrade the implementation of the proxy
 - `._authorizeUpgrade()` -- require a certain sender to be used (when upgrading the implementation) The `CsToken` contract contains the following privileged roles:
- `onlyClayMain` modifier-guarded functions
 - `.mint()` -- mints tokens to an address, increasing the total supply
 - `.burn()` -- burn an amount of tokens from an address, reducing the total supply The `UserProxy` contract contains the following privileged roles:
- `onlyProxy` modifier-guarded functions
 - `.initialize()` -- initialize the contract's variables post-deployment.
 - `.upgradeTo()` -- upgrade the implementation of the proxy to a new contract.
- `onlyRole(TIMELOCK_ROLE)` modifier-guarded functions
 - `.processInsurance()` -- process insurance towards ClayMatic donation and fees
- `onlyRole(TIMELOCK_UPGRADES_ROLE)` modifier-guarded functions
 - `.setClaimVault()` -- sets the vault to which matic tokens will be sent
 - `.setTreasury()` -- set the treasury wallet
 - `.upgradeTo()` -- upgrade the implementation of the proxy
 - `._authorizeUpgrade()` -- require a certain sender to be used (when upgrading the implementation)
- `onlyRole(CS_SERVICE_ROLE)` modifier-guarded functions
 - `.pause()` -- pause the contract
 - `.unpause()` -- unpause the contract The `ClaimVault` contract contains the following privileged roles:
- `claimFunds()` requires the sender to be `clayManager`.

Recommendation: Clarify the impact of these privileged actions to the end-users via publicly facing documentation.

Update: Documentation has been added detailing the roles and privileges.

QSP-4 Fees May Be Uninitialized

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `CLayManager.sol`

Description: Fees are charged when users deposit, withdraw, instantly withdraw, or make an early claim on funds. However, the only fee guaranteed to be initialized is the deposit fee. If the remaining fees remain uninitialized for any transactions, the protocol may lose out on earnings.

Recommendation: Either initialize the missing fees in `initialize()`, or add guards to check whether the fee has been initialized.

Update: Zero fees are by design.

QSP-5 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `CLayManager.sol`

Related Issue(s): [SWC-123](#)

Description: Some functions do not validate their inputs, which can result in unexpected behavior by the contracts. A non-exhaustive list includes:

- `CLayManager.initialize()`: Validate that `_csToken` is a non-zero address.
- `CLayManager._claim()`: Validate that `claimVault` is a non-zero address.
- `CLayManager.earlyClaim()`:
 - Validate that order exists.
 - Validate that `claimVault` is a non-zero address.
 - Validate that the order cannot be claimed through ordinary means.

Recommendation: Add the missing input validation.

Update: The missing input validation was added.

QSP-6 Missing Initializers

Severity: *Low Risk*

Status: Fixed

File(s) affected: `CLayManager.sol`

Description: `CLayManager` is missing a call to `__UUPSUpgradeable_init()` in the `initialize()` function. Though the function may be empty, it is important to call the initializer in case this changes in future version upgrades.

Recommendation: Add the missing initializer.

Update: The missing initializer was added.

QSP-7 Unchecked Return Value

Severity: *Low Risk*

Status: Fixed

File(s) affected: `UserProxy.sol`

Related Issue(s): [SWC-104](#)

Description: Most functions will return a `true` or `false` value upon success. It is important to ensure that every necessary function is checked. `UserProxy.deposit()` does not check the return value of its call to `clayMatic.deposit(amount)`.

Recommendation: Check the return value of `clayMatic.deposit(amount)`.

Update: The return value is now checked.

QSP-8 Reliance on Off-Chain Component to Be Working as Expected

Severity: *Informational*

Status: Acknowledged

File(s) affected: `UserProxy.sol`

Description: The `UserProxy` contract is responsible for completing the process of a batch order that is specified by `batchID` in polygon, using the functions below:

1. `UserProxy.deposit()`
2. `UserProxy.withdraw()`

These functions are designed and expected to be called only once by the tunnel. The tunnel is expected to guard against replay attacks. However, we note that it is outside the scope of this audit.

Recommendation: Add a check to the functions listed to make sure that a `_batchId` cannot be processed more than once.

Update: The team is confident that `FxBaseRootTunnel.sol` will ensure no replay attacks as the contract is supplied by Polygon.

QSP-9 Block Timestamp Manipulation

Severity: *Informational*

Status: Acknowledged

File(s) affected: `CLayManager.sol`

Related Issue(s): [SWC-116](#)

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes. If a smart contract relies on a timestamp, it must take this into account.

Recommendation: Either test for whether block timestamp manipulation will affect protocol behavior, or add user-facing documentation explaining the potential for such an attack and its consequences.

Update: The team has acknowledged the issue saying that timestamp manipulation should not have any effect on the protocol behavior.

QSP-10 Timelock Should Allow Adequate Time for Users to Exit

Severity: *Informational*

Status: Acknowledged

Description: Some functions in `CLayManager` are restricted to the `TIMELOCK_ROLE` and `TIMELOCK_UPGRADES_ROLE`. This gives users time to exit before a potentially dangerous change is applied. However, with withdrawals potentially taking several days before they can be claimed, it is important that the timelock be configured such that users have adequate time to complete their exit if they choose to do so.

Recommendation: Though the timelock and role manager are outside the scope of this audit, we advise that the timelock duration be sufficient for a user to withdraw their MATIC without having to pay the early claim or instant withdrawal fee.

Update: The team has said that community announcements will be made with sufficient time for users to act accordingly.

QSP-11 Upgradability

Severity: *Informational*

Status: Acknowledged

Description: Many contracts within the project are upgradeable. While this is not a vulnerability, users should be aware that the behavior of the contracts could drastically change if the contracts are upgraded. Furthermore, new vulnerabilities not present during the audit could be introduced in upgraded versions of the contract, or if contract upgrade deployments are not done correctly.

Recommendation: The contracts upgradeability and any reasons for future upgrades should be communicated to users beforehand.

Update: Upgradeability will be properly communicated to the users.

QSP-12 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `UserProxy.sol`

Related Issue(s): [SWC-103](#)

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

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

Update: The solidity version has been locked to `0.8.15`.

Adherence to Specification

Although there is substantial NatSpec and documentation describing the system at a high level, there is relatively little documentation regarding the specifics of the protocol's operation (like exchange rates).

Code Documentation

- Typos
 - `CLayManager.sol#246-247` complaint compliant
 - `CLayManager.sol#306` contract
 - `CLayManager.sol#826` daydays
- `CLayManager.sol#503` calls the section "PUBLIC VIEWS" whereas many of the views below are `external` and `internal`.

Adherence to Best Practices

- Rename `_extraFee` at `CLayManager.sol#413` to something more descriptive like `_instantClaimFee`.
- Rename `claimableBy` at `CLayManager.sol#138` to `claimableAt`.

- In `ClayManager.sol#L543`, rename `maxMatic` to avoid confusion around its purpose, given that it is assigned using the `_min()` function.

Test Results

Test Suite Results

The test suite was run using `npm run test`.

```
Running 1 test for test/ClaimDaysSetter.t.sol:ClaimDaysSetter
[PASS] testSetDays() (gas: 88567)
Test result: ok. 1 passed; 0 failed; finished in 12.07ms

Running 3 tests for test/ProcessMessage.t.sol:ProcessMessage
[PASS] testProcessMessageToChildClaim() (gas: 1280089)
[PASS] testProcessMessageToRootDeposit() (gas: 909936)
[PASS] testProcessMessageToRootWithdraw() (gas: 1115820)
Test result: ok. 3 passed; 0 failed; finished in 23.43ms

Running 13 tests for test/RoleManager.t.sol:TestRoleManager
[PASS] testCannotGrantRole() (gas: 33464)
[PASS] testCannotPauseClayManager() (gas: 82731)
[PASS] testCannotPauseUserProxy() (gas: 174119)
[PASS] testCannotSetFee() (gas: 147526)
[PASS] testCannotUpgrade() (gas: 13432)
[PASS] testCannotUpgradeClayManager() (gas: 165583)
[PASS] testCannotUpgradeUserProxy() (gas: 165474)
[PASS] testGrantRole() (gas: 51448)
[PASS] testPauseClayManager() (gas: 81370)
[PASS] testRevokeRole() (gas: 42863)
[PASS] testSetTreasury() (gas: 88505)
[PASS] testSetupRoleAdmin() (gas: 78616)
[PASS] testUpgrade() (gas: 25531)
Test result: ok. 13 passed; 0 failed; finished in 10.17ms

Running 11 tests for test/AutobalanceRun.t.sol:AutobalanceRun
[PASS] testAutobalanceRunForExtraGasPassed() (gas: 892266)
[PASS] testAutobalanceRunForHighPendingAmountHigh() (gas: 2052696)
[PASS] testAutobalanceRunForLowPendingAmount() (gas: 1825148)
[PASS] testAutobalanceRunForMedPendingAmount() (gas: 1827182)
[PASS] testAutobalanceRunForWithdrawalThresholdPassed() (gas: 1093182)
[PASS] testAutobalanceRunNetStakeLowPending() (gas: 726632)
[PASS] testAutobalanceRunNetStakeSwap() (gas: 1087834)
[PASS] testAutobalanceRunNetUnstakeLowPending() (gas: 890122)
[PASS] testAutobalanceRunWhenNoPendingDeposits() (gas: 38585)
[PASS] testAutobalanceRunWhenZeroDepositsLowFreqTimePassed() (gas: 156928)
[PASS] testCannotAutobalanceRunWhenPaused() (gas: 92538)
Test result: ok. 11 passed; 0 failed; finished in 42.30ms

Running 13 tests for test/Autobalance.t.sol:AutoBalance
[PASS] testAutoNetStake() (gas: 671034)
[PASS] testAutoNetStake2() (gas: 928012)
[PASS] testAutoNetStakeExchange() (gas: 1009345)
[PASS] testAutoNetStakeSlashing() (gas: 1155994)
[PASS] testAutoNetStakeWithdrawDonation() (gas: 1350974)
[PASS] testAutoNetUnstake1() (gas: 1047520)
[PASS] testAutoNetUnstake2() (gas: 885920)
[PASS] testAutoNetUnstakeExchange() (gas: 1114278)
[PASS] testAutoNetUnstakeExchange2() (gas: 1069418)
[PASS] testAutoNetUnstakeSlashing() (gas: 1733774)
[PASS] testAutoNetZero() (gas: 598203)
[PASS] testAutoStake() (gas: 479436)
[PASS] testAutoUnstakeIncreaseRate() (gas: 1729756)
Test result: ok. 13 passed; 0 failed; finished in 61.36ms

Running 4 tests for test/SetFee.t.sol:TestSetFee
[PASS] testCannotSetFeesWithoutTimeLockRole() (gas: 55659)
[PASS] testCannotSetInvalidFees() (gas: 109323)
[PASS] testSetFee(uint8) (runs: 256,  $\mu$ : 79174, -: 81289)
[PASS] testSetFeeEvent(uint8) (runs: 256,  $\mu$ : 77445, -: 80458)
Test result: ok. 4 passed; 0 failed; finished in 47.93ms

Running 8 tests for test/CsToken.t.sol:TestCsToken
[PASS] testApproval(uint256,uint8) (runs: 256,  $\mu$ : 92471, -: 116495)
[PASS] testBurnAuth() (gas: 55393)
[PASS] testCannotBurnMoreThanBalance(uint256) (runs: 256,  $\mu$ : 51080, -: 52108)
[PASS] testCannotTransferToZeroAddress(uint256,uint8) (runs: 256,  $\mu$ : 58927, -: 67276)
[PASS] testMintAuth() (gas: 70225)
[PASS] testMintBurn(uint256,uint8) (runs: 256,  $\mu$ : 47932, -: 54964)
[PASS] testSetClayMain() (gas: 19721)
[PASS] testTransfer(uint256,uint8) (runs: 256,  $\mu$ : 73043, -: 97047)
Test result: ok. 8 passed; 0 failed; finished in 103.50ms

Running 6 tests for test/EarlyClaim.t.sol:TestEarlyClaim
[PASS] testEarlyClaimFullInstantWithdraw() (gas: 836816)
[PASS] testEarlyClaimNetUnstake() (gas: 1070811)
[PASS] testEarlyClaimStakeSlashingBefore() (gas: 1490701)
[PASS] testEarlyClaimStatusChange() (gas: 904011)
[PASS] testEarlyClaimUnstakeSlashingAfter() (gas: 1993280)
[PASS] testEarlyClaimUnstakeSlashingBefore() (gas: 1490913)
Test result: ok. 6 passed; 0 failed; finished in 27.71ms

Running 6 tests for test/UserProxyClaim.t.sol:TestUserProxyClaim
[PASS] testUserProxyClaim() (gas: 584099)
[PASS] testUserProxyClaimCannot() (gas: 436304)
[PASS] testUserProxyClaimIncreaseRateDonation() (gas: 802542)
[PASS] testUserProxyClaimSlashedAfterWithdraw() (gas: 909357)
[PASS] testUserProxyClaimSlashedBeforeWithdraw() (gas: 907325)
[PASS] testUserProxyIsClaimable() (gas: 415654)
Test result: ok. 6 passed; 0 failed; finished in 19.63ms

Running 11 tests for test/Setters.t.sol:TestSetters
[PASS] testSettersBridgeRequirementsFuzz(uint8,uint8,uint8,uint8,uint8) (runs: 256,  $\mu$ : 4234, -: 4230)
[PASS] testSettersCannotSetEthGasFee() (gas: 113636)
[PASS] testSettersCannotSetMinClaimDays() (gas: 175896)
[PASS] testSettersCannotSetTrfReq() (gas: 130229)
[PASS] testSettersEthGasFeeFuzz(uint256) (runs: 256,  $\mu$ : 92900, -: 92900)
[PASS] testSettersInvalidTrfReq() (gas: 124740)
[PASS] testSettersManagerCannotSetTreasury() (gas: 162829)
[PASS] testSettersManagerSetTreasury() (gas: 39968)
[PASS] testSettersMinClaimDays(uint8) (runs: 256,  $\mu$ : 72824, -: 74318)
[PASS] testSettersProxyCannotSetTreasury() (gas: 162862)
[PASS] testSettersProxySetTreasury() (gas: 40065)
Test result: ok. 11 passed; 0 failed; finished in 179.19ms

Running 6 tests for test/Slashing.t.sol:Slashing
[PASS] testSlashingDonationFromBefore() (gas: 1071571)
[PASS] testSlashingNetStakeAfter() (gas: 1246057)
[PASS] testSlashingNetStakeBefore() (gas: 1250207)
[PASS] testSlashingNetUnstakeBefore() (gas: 1393007)
[PASS] testSlashingNetZeroAfter() (gas: 995667)
[PASS] testSlashingNetZeroBefore() (gas: 997675)
Test result: ok. 6 passed; 0 failed; finished in 25.55ms

Running 5 tests for test/StandardFlow.t.sol:StandardFlow
[PASS] testDeposit() (gas: 574234)
[PASS] testExchangeRate() (gas: 260371)
[PASS] testStandardFlowTransfer() (gas: 1446990)
[PASS] testSwap() (gas: 256013)
[PASS] testWithdraw() (gas: 741029)
Test result: ok. 5 passed; 0 failed; finished in 15.94ms

Running 4 tests for test/Swap.t.sol:TestSwap
[PASS] testSwap() (gas: 454214)
[PASS] testSwapCannotWithoutApproval() (gas: 324326)
```

```

[PASS] testSwapCannotZeroTokens() (gas: 55012)
[PASS] testSwapFuzz(uint96) (runs: 256, μ: 262070, ~: 262077)
Test result: ok. 4 passed; 0 failed; finished in 168.97ms

Running 4 tests for test/TimeLock.t.sol:TestTimeLock
[PASS] testCannotUpgrade() (gas: 105201)
[PASS] testExecuteCall() (gas: 68920)
[PASS] testMinDelay() (gas: 18559)
[PASS] testUpgrade() (gas: 46728)
Test result: ok. 4 passed; 0 failed; finished in 8.44ms

Running 1 test for test/UserClaimsView.sol:TestUserClaimsView
[PASS] testUserClaimsView() (gas: 3218420)
Test result: ok. 1 passed; 0 failed; finished in 14.73ms

Running 9 tests for test/UserProxyDeposit.t.sol:TestUserProxy
[PASS] testUserProxyDeposit() (gas: 272808)
[PASS] testUserProxyDepositFuzz(uint128,uint8) (runs: 256, μ: 286955, ~: 286972)
[PASS] testUserProxyDepositInsurance() (gas: 319263)
[PASS] testUserProxyDepositMismatch() (gas: 182997)
[PASS] testUserProxyDepositNoWithInvalidCode() (gas: 19820)
[PASS] testUserProxyDepositWhenBalanceIsLow() (gas: 52366)
[PASS] testUserProxyDepositWhenPaused() (gas: 93680)
[PASS] testUserProxyDepositWithExtraParams() (gas: 254752)
[PASS] testUserProxyDepositWithFee() (gas: 309240)
Test result: ok. 9 passed; 0 failed; finished in 274.10ms

Running 3 tests for test/Vault.t.sol:Vault
[PASS] testVaultClaimFunds() (gas: 317196)
[PASS] testVaultSetInManager() (gas: 112540)
[PASS] testVaultSetInProxy() (gas: 112502)
Test result: ok. 3 passed; 0 failed; finished in 6.87ms

Running 9 tests for test/UserProxyWithdraw.t.sol:TestUserProxy
[PASS] testFailUserProxyWithdrawWithLowMaxAmountCs() (gas: 992605)
[PASS] testFailUserProxyWithdrawWithZeroBalance() (gas: 59251)
[PASS] testUserProxyMultipleWithdrawInBatch() (gas: 710521)
[PASS] testUserProxyWithdraw() (gas: 499747)
[PASS] testUserProxyWithdrawFuzz(uint128,uint16) (runs: 256, μ: 416934, ~: 416934)
[PASS] testUserProxyWithdrawInBatches() (gas: 1701957)
[PASS] testUserProxyWithdrawWhenPaused() (gas: 93629)
[PASS] testUserProxyWithdrawWithExtraParams() (gas: 406614)
[PASS] testUserProxyWithdrawWithInvalidCode() (gas: 19776)
Test result: ok. 9 passed; 0 failed; finished in 257.13ms

Running 4 tests for test/DepositMismatch.t.sol:DepositMismatch
[PASS] testDepositMismatchAtDepositOnProxy() (gas: 866585)
[PASS] testDepositMismatchAtDepositOnProxyFuzz(uint96) (runs: 256, μ: 867344, ~: 867345)
[PASS] testDepositMismatchOnManager() (gas: 717130)
[PASS] testDepositMismatchOnManagerFuzz(uint96,uint96) (runs: 256, μ: 717933, ~: 717932)
Test result: ok. 4 passed; 0 failed; finished in 1.42s

Running 8 tests for test/UserProxyMisc.t.sol:UserProxyMisc
[PASS] testUserProxyCannotDonateWithLowBalance(uint8) (runs: 256, μ: 31638, ~: 31638)
[PASS] testUserProxyCannotProcessInsuranceWithoutInsurance() (gas: 40690)
[PASS] testUserProxyCannotProcessInsuranceWithoutTimeLock() (gas: 36030)
[PASS] testUserProxyDonate(uint256) (runs: 256, μ: 223469, ~: 226469)
[PASS] testUserProxyInit() (gas: 8824582)
[PASS] testUserProxyProcessInsuranceFuzz(uint16,uint16) (runs: 256, μ: 316725, ~: 317489)
[PASS] testUserProxyUnpause() (gas: 107977)
[PASS] testUserProxyUpgradeTo() (gas: 3764503)
Test result: ok. 8 passed; 0 failed; finished in 1.14s

Running 9 tests for test/Deposit.t.sol:TestDeposit
[PASS] testDelegateDepositFuzz(uint96) (runs: 256, μ: 582313, ~: 585623)
[PASS] testDeposit() (gas: 574397)
[PASS] testDepositDelegate() (gas: 574756)
[PASS] testDepositDiffExRateFuzz(uint96,uint96) (runs: 256, μ: 381680, ~: 400712)
[PASS] testDepositETH() (gas: 245556)
[PASS] testDepositETHFuzz(uint96) (runs: 256, μ: 399200, ~: 400221)
[PASS] testDepositFuzz(uint96) (runs: 256, μ: 586993, ~: 586994)
[PASS] testDepositNoZero() (gas: 419290)
[PASS] testDepositNoZeroApproval() (gas: 482101)
Test result: ok. 9 passed; 0 failed; finished in 1.53s

Running 9 tests for test/Claim.t.sol:TestClaim
[PASS] testClaimCannotAlreadyClaimed() (gas: 676271)
[PASS] testClaimCannotInvalidOrder() (gas: 1047708)
[PASS] testClaimEoaFuzz(uint96) (runs: 256, μ: 788856, ~: 788856)
[PASS] testClaimMultiFuzz(uint96) (runs: 256, μ: 891393, ~: 891393)
[PASS] testClaimWithFee() (gas: 852485)
[PASS] testClaimWithWithdrawFeeFuzz(uint96,uint16) (runs: 256, μ: 849643, ~: 849643)
[PASS] testClaim_1() (gas: 682507)
[PASS] testClaim_2() (gas: 798602)
[PASS] testClaim_3() (gas: 763504)
Test result: ok. 9 passed; 0 failed; finished in 1.62s

Running 8 tests for test/Withdraw.t.sol:TestWithdraw
[PASS] testCannotWithdrawMoreThanBalance() (gas: 382531)
[PASS] testCannotWithdrawZeroTokens() (gas: 119290)
[PASS] testWithdraw() (gas: 855540)
[PASS] testWithdrawFuzz(uint96) (runs: 256, μ: 751282, ~: 751283)
[PASS] testWithdrawMaxCsTokens() (gas: 853540)
[PASS] testWithdrawVariableRateFuzz(uint96,uint96) (runs: 256, μ: 751642, ~: 751641)
[PASS] testWithdrawWithFees() (gas: 1030068)
[PASS] testWithdrawWithFeesFuzz(uint96) (runs: 256, μ: 570325, ~: 570324)
Test result: ok. 8 passed; 0 failed; finished in 579.08ms

Running 4 tests for test/Donations.t.sol:Donations
[PASS] testDonationToFillMismatchAmountInProxy() (gas: 510646)
[PASS] testDonationToFillMismatchAmountInProxyFuzz(uint96) (runs: 256, μ: 515263, ~: 515263)
[PASS] testDonationToManagePendingAmtInManager() (gas: 702808)
[PASS] testDonationToManagePendingAmtInManagerFuzz(uint96) (runs: 256, μ: 703583, ~: 703584)
Test result: ok. 4 passed; 0 failed; finished in 1.59s

```

Code Coverage

Code coverage was gathered by running `npm run coverage` followed by `genhtml lcov.info` to view the coverage report.

| Filename | Line Coverage | Functions |
|--------------------|------------------|-------------------|
| ClaimVault.sol | 100.0% 3/3 | 50.0% 1/2 |
| ClayManager.sol | 92.1% 257/279 | 97.3% 36/37 |
| CsToken.sol | 85.7% 6/7 | 100.0% 3/3 |
| UserProxy.sol | 97.9% 93/95 | 94.1% 16/17 |

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

c04da9b4c7c5126424be90ce6e21978a7558648ccbfa1aebb2b7da15091278e ./src/ClaimVault.sol
bd11d46ec120e160e3a9b2f96e0b1c35d34862b63a269791358eff82234fe1b6 ./src/RoleManager.sol
a04aeb9854d0f5a9b14a99059bb4ed39bc17265979bba3d8c56fc54f90cd6e69 ./src/ClayManager.sol
282c254a179ed963c5038a2cf9b659febbf981922f1a01d6051221367bf92092 ./src/UserProxy.sol
340cc7c53814ed92e17fd39a1ab646ef278b85c21b3be151972555dc03261b9d ./src/TimeLock.sol
d8c396a3138c8e14930e7a55e9ebf83d45b73a438f548e783bf050223f293b25 ./src/CsToken.sol
e4e7346b5df477bb1565c487db4b1f925cebbd174021ccaa6abfb21327d74fcf ./src/interfaces/IClayBase.sol
26dedb22efe0b55e5c27369474323606036dfbf5a90c1ff16c505fd965447e25 ./src/interfaces/IClayManager.sol
e2117f25344e95061f6ff836ee4d73f47cdac03a90ab810c9e9310c28f780429 ./src/interfaces/IRoleManager.sol
f5d214fd260a99692c0a5926a59b6dceca597be4c1b1d2fcec0bd3bf71f3eed5 ./src/interfaces/IWETH9.sol
695b777f8d0f5f86e9ae6417b7a9bf79a8485c9760caf027a6da6caa4dce26cb ./src/interfaces/IRootChainManager.sol
6873a0a8cb09dca7b26571c370a4d39ada078fb8930466b8cd17c43688f922b9 ./src/interfaces/IChildToken.sol
39afcdcb2e2804bb7b0b3a5fa2fed316eb59f77e295b9f8cf2b473d63b544ea2 ./src/interfaces/IFxRoot.sol
a8b2f5540532051e2c49f97fc25fab3b66d31f8b2b4a5b533ecbf2b286ff8ea ./src/interfaces/IClayMain.sol
bf5601197b555ec1d6af9f8052f21e73e83286b79f1e81b9cd8e937a413f505c ./src/interfaces/IERC20Predicate.sol
c12e02640f5135e4d8532f376a893439f33cb5c5c5a23f99af02f5476c329d98 ./src/interfaces/IDepositManager.sol
ddb53689bf09742dadfd91acf91d1bec87cdcc4366332f5868c4e0c1d08c00ff ./src/interfaces/IClaimVault.sol
c747116af8f1b508c0b1008edf0f107d65e7677138ea8da75bd6de1228faa74f ./src/interfaces/ICSToken.sol
5ca60f3c5e829a531a07764fedeedae503499d8d89b067a93ae53b8e68d865a9 ./src/interfaces/IClayTunnel.sol
419df79a868761cd7483b17157c02157d203e247bc20ec468089ab72adad6622 ./src/lib/RLPReader.sol
a5d1c31f60db444a843321c40eafd38550edd5ff567fd1c58773a8b9ee047408 ./src/lib/ExitPayloadReader.sol
7b47e823e8d181d74317272908b214ae84545db99a6232b9b12998c473cdd6ea ./src/lib/Merkle.sol
fa9e730fcf474d4587a176cab164f20263344c8ac8411f309eb548bcd57d1777 ./src/lib/MerklePatriciaProof.sol
5be23cba2a6d15b6b25f5a0a710a9b317b0b1bd7cc03b9fb2f81d001c0db79ab ./src/tunnel/FxBaseChildTunnel.sol
ef62073f28d09bddc461007fb2f134781b65e3f7505781947195a8f69776e822 ./src/tunnel/FxBaseRootTunnel.sol

Tests

b62f33f72d886eeaaecfa6a41765f06a9a0fe2473e435eb9953b1a96bb93ab70 ./test/Vault.t.sol
db793c31a580671a37f1985f9447cb0a13ee8347acbd3df4f82557631f7abb5a ./test/EarlyClaim.t.sol
edc479e662ec2260bef1e250c6ea7bc7f3a2ae1ca6f49410caae89c8b4b9e17d ./test/Deposit.t.sol
7dd92e8405f588ae3f1adb087b271397cf5b48049367661ddde974f4c1bad9be ./test/DepositMismatch.t.sol
c047848678be094c0a72f04780c9d1ab4b929b0335ce41aa81d26a8ef9acd961 ./test/SetFee.t.sol
6830493c66e8c5e98f6cc0f6065f55449eaae5e658f56fc9f536140f3483f784 ./test/RoleManager.t.sol
7330df64f121ee8dad55e07bb4639e3b579c45c273704b6a6e17c6fd970b5fdc ./test/Setters.t.sol
bcbb220c6e8c66c50a656b1a8b7a5f2df03bb6d4b847b17acca224d9700aeb65 ./test/Donations.t.sol
4cd0ec6c1726cc4466165b3f8a8d117ba2f68f296ba5b565bb2f7c160ef431fb ./test/Slashing.t.sol
f1b32b1a63200de72c59f667840612c4687c196198704e49a7770ed298b3a1a3 ./test/Autobalance.t.sol
314ab47febf53546f1e8ab190db30e42193e0a007478ee011784cdf2230847ea ./test/UserProxyClaim.t.sol
4c44ac33061860981f223541362c4bbdd3af9693225dc13fbbdd4ea1bab6d97b7 ./test/TestUtil.t.sol
bed29f1fcf75f4ea579c1d802849375496f17e48abaf60eed75a2a806fe86866 ./test/Claim.t.sol
58791e314fc2b2cf0ff75a953d8c07ffe818d7a4c0d2a25b31f7d74da5a50 ./test/UserClaimsView.sol
39d8b02c83ee0bdfbdc6feb4b7b2ffeff38028e836323df44eea806b4d3cb6 ./test/AutobalanceRun.t.sol
ab38ac13305e0696a7aeed934f6a5319c574cd1024ac56b7d4b5b54a36d232c1 ./test/StandardFlow.t.sol
889cd52aba4de8bb371fcf61e07b57072391a25bb328196695bc50e43bf6c756 ./test/Withdraw.t.sol
6b11137760abd241ceaf065d8f4f9148dfd9164de99ef5fef01a0c7d34af4d43 ./test/CsToken.t.sol
fd5f630602ab0d7c3185209c41d36e193641ccba50e57048ee4b4483c1853ca1 ./test/ProcessMessage.t.sol
7e13b96643636289c576888f28d47e3dabd66bfb69340a5276865b1ecdff89c7 ./test/TimeLock.t.sol
140a4f01b107463a923455203306dfd01e4b21f4ae6d9730bbe18a4833c4f08b ./test/UserProxyMisc.t.sol
c2e4910383c8ff1f4e05398fc2b2821e42f7846738303efd397d76c8ba5c9cce ./test/UserProxyWithdraw.t.sol
f46b728659860fec6b92a049212693353080068f8b32370ff65abf255a800be5 ./test/UserProxyDeposit.t.sol
42985cc51a8863de6a2efccc793e1cf94cef2999e47158f6f74d6a1f1f37cc02 ./test/ClaimDaysSetter.t.sol
232608e53b16f7ae9aaee872ebb25981839003bf2fee3b0b25468eb514f5ee33 ./test/Swap.t.sol
bed93a12aea87516ecff796c62d43485d87036c85cb899f9db1d702d0012f3d8 ./test/mocks/MockClayMatic.sol
36e206c4581a202d6bdd0c83a830732efa8c9a77f9e1ac0305cdf0a5b4449e26 ./test/mocks/MockERC20.sol
980338b3ad1311488a506fd3483b6e9ac33459f7d12a79c3196f1e012eef1440 ./test/mocks/MockRoleManager.sol
03ad327173d08160022940b6507a3a99a3912e2713d09c11cd209979d12393c8 ./test/mocks/MockClayManager.sol
cdc161e6884f3a5483d23529944815751d2d312c32865d96e43fb899f3655c66 ./test/mocks/MockDepositManager.sol

3cd1bbd2209f32809f174902954db489a46f18baed0d94bd7fd0f544ad52f965 ./test/mocks/MockChildToken.sol

a39c7031c5d480e1bb058e5d045763d56ef49a97f0de7ba67b99c1d88b28f483 ./test/mocks/MockUserProxy.sol

9c4c007a8dd5abc9d378cd5866bd43a40f4da6d43ba644554a1d9532dbf1a589 ./test/mocks/MockFxRoot.sol

Changelog

- 2022-09-26 - Initial report
- 2022-10-20 - Fix Review

About Quantstamp

Quantstamp is a global leader in blockchain security backed by Pantera, Softbank, and Commonwealth among other preeminent investors. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its white glove security and risk assessment services.

The team consists of web3 thought leaders hailing from top organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Many of the auditors 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 250 audits and secured over \$200 billion in digital asset risk from hackers. In addition to providing an array of security services, Quantstamp facilitates the adoption of blockchain technology through strategic investments within the ecosystem and acting as a trusted advisor to help projects scale.

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, Aave, 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.

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, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' 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 the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim 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. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, 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, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. 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. 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.