Quantstamp Security Assessment Certificate

September 18th 2020 — Quantstamp Verified

Teller Protocol

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Lending platform
Auditors	Alex Murashkin, Senior Software Engineer Poming Lee, Research Engineer Kevin Feng, Software Engineer
Timeline	2020-06-25 through 2020-07-27
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Medium article README.md Inline code comments
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
teller-protocol-v1	dd7d2ac
teller-protocol-v1	6c40d64

Total Issues	**15** (8 Resolved)
High Risk Issues	**2** (1 Resolved)
Medium Risk Issues	**3** (3 Resolved)
Low Risk Issues	**8** (4 Resolved)
Informational Risk Issues	**2** (0 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).
Resolved	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

During auditing, we found 15 potential issues of various levels of severity: two high-severity, three medium-severity, eight low-severity issues, as well as two informational-level findings. The code looks well-structured and concise, however, there are significant security considerations that need to be addressed before the live deployment.
Test coverage is nearing 100%, however, we suggest adding coverage to certain lines and branches, as well as covering more edge cases and access control scenarios that are not reflected in the coverage measures.
And finally, we made 20 best practices recommendations which include naming, consistency, input validation, and other suggestions.
We highly recommend addressing the findings before going live.
Update as of commit dd7d2ac:
The team addressed issues in multiple pull requests and documentation changes that were reviewed by the Quantstamp team.

1. Best practices were addressed in the pull request #68.
2. Major findings were addressed in the pull requests: #70, #71, and #74.
3. Documentation and unit test improvements were made in the pull requests #81, #83, #84, and #85.

The findings QSP-14, and QSP-15 were deemed as non-issues (not worth fixing and already matching the expected behaviour, respecively) after discussing with the Teller Protocol team. QSP-3 was marked as a false-positive (but kept in the report in order not to break the numbering sequence).
Next, the findings QSP-2, QSP-6, QSP-9, and QSP-11 represent risks of various severities that **remain present in the code**. For example, QSP-2 involves a design choice of using off-chain components that take part in security-critical operations such as deciding on interest rates and maximum loan amount. The Teller team acknowledged the risks and documented them.
And finally, QSP-13 is not an issue as long as the number of off-chain nodes remain low: the gas limits would be reached at the simultaneous processing ~140 responses, which is higher than the number the Teller Protocol team is planning to use (6-8).

ID	Description	Severity	Status
QSP-1	Potential reuse of response signatures across contracts	High	Fixed
QSP-2	Risky reliance on extremely privileged off-chain components	High	Acknowledged
QSP-3	[False-positive] Collateral value disconnected from real value	Medium	Fixed
QSP-4	Potential DoS due to uninitialized `timeLastAccrued`	Medium	Fixed
QSP-5	Sybil attack risks	Medium	Fixed
QSP-6	Risky reliance on price oracles	Low	Acknowledged
QSP-7	Discrepancy of `liquidationPayment(...)` with the documentation	Low	Fixed
QSP-8	Missing balance checks for transfers of token that are not fully ERC-20 compliant	Low	Fixed
QSP-9	Privileged Roles and Ownership	Low	Acknowledged
QSP-10	`isWithinTolerance(...)` not working for small positive values	Low	Fixed
QSP-11	Block Timestamp Manipulation	Low	Acknowledged
QSP-12	Lack of integer overflow checks in oracle computations	Low	Fixed
QSP-13	Gas Usage / `for` Loop Concerns	Low	Acknowledged
QSP-14	Method visibility is too open	Informational	Acknowledged
QSP-15	Disparity between `ChainlinkPairAggregator` and `InverseChainlinkPairAggregator`	Informational	Acknowledged

Quantstamp Audit Breakdown

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

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

Note: All line numbers in the findings are relative to the commit `6c40d64`.

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:

- Maian commit sha: ab387e1
- Mythril v0.2.7
- Slither v0.6.6

Steps taken to run the tools:

1. Cloned the MAIAN tool: `git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian`
2. Ran the MAIAN tool on each contract: `cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol`
3. Installed the Mythril tool from Pypi: `pip3 install mythril`
4. Ran the Mythril tool on each contract: `myth -x path/to/contract`
5. Installed the Slither tool: `pip install slither-analyzer`
6. Run Slither from the project directory: `slither .`s

Findings

QSP-1 Potential reuse of response signatures across contracts

Severity: *High Risk*

Status: Fixed

File(s) affected: `Multiple`

Description: One example: the contracts `TokenCollateralLoans` and `EtherCollateralLoans` share the same format for `LoanRequest` and `LoanResponse`. If the two contracts are initialized with different values for `loanTermsConsensusAddress`, the two contracts become vulnerable, since someone could reuse the values for approving one loan type for getting a loan of a different type.
The same issue could exist for other contracts. For example, if two instances of `Lenders.sol` is deployed, the same exploit scenario is possible.

Recommendation:

1. Enforcing a single location for keeping track of the used requests and signatures.
2. When releasing a new version of the contracts, making sure no two stacks are run in parallel or signature verification logic is incompatible across the stacks.

QSP-2 Risky reliance on extremely privileged off-chain components

Severity: *High Risk*

Status: Acknowledged

Description: By design of the Teller Protocol, off-chain components are involved in security-critical operations such as deciding on interest rates and maximum loan amount. While an issue localized to a small fraction of the nodes can be handled by the system, a significant malfunction of nodes can cause catastrophic consequences to the system. Examples include hypothetical scenarios when all nodes incorrectly return a zero for interest rates or a disproportionally high maximum loan amount after, for instance, a simultaneous software upgrade of nodes to a version containing a bug.
Additionally, it is unclear how available liquidity is tracked in the system. In case loans get mistakenly approved, all available tokens would be drained from Compound and given out as loans until it is no longer possible to do so. It is unclear if such a risk is expected.

Recommendation: The off-chain components are not in the scope of the current audit engagement. However, we recommend:

1. Acknowledging and documenting the risks, including a description of the risks in public user-facing documentation.
2. Considering adding alerting and testing the off-chain components to reduce the likelihood of a malfunction.
3. Considering adding additional guards to the contract: for instance, not allowing to loan above a certain soft-cap, to mitigate the risks.
4. Adding tracking for liquidity and the necessary checks to enforce the minimum liquidity.

QSP-3 [False-positive] Collateral value disconnected from real value

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `base/LoansBase.sol`

Description: In `base/LoansBase.sol`, `L255`: the function `liquidateLoan(...)` only checks if the `collateral` falls below the `loans[loanID].loanTerms.collateralRatio`, which is only set when the loan is created and is never updated. In case of a significant price drop of collateral, the liquidator is unable to liquidate any collateral since the condition check is only based on the `loans[loanID].loanTerms.collateralRatio` without taking the actual USD price of the collateral into consideration.

Recommendation: Considering relying on the fiat value of collateral for loans.
Update: the team provided an explanation, and the issue was marked as false-positive.

QSP-4 Potential DoS due to uninitialized `timeLastAccrued`

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `base/Lenders.sol`

Description: In `base/Lenders.sol`, `L91`: the requirement `accruedInterest[request.lender].timeLastAccrued == request.startTime` is likely to be violated for the first call `setAccruedInterest(...)` since `timeLastAccrued` is not initialized. This causes the method `setAccruedInterest` to be unusable unless `request.startTime` is set to `0` initially, and it is unclear whether this is expected.

Recommendation: Clarifying the expectation. Initializing `timeLastAccrued` with `request.endTime`.

QSP-5 Sybil attack risks

Severity: *Medium Risk*

Status: Fixed

Description: In OpenZeppelin's implementation, the `SignerRole` is designed in a way that allows signers to add other signers:

```
    function addSigner(address account) public onlySigner {
        _addSigner(account);
    }
```

This opens an opportunity for a misbehaving node to perform a Sybil attack by whitelisting multiple other accounts and taking part in consensus, thus

bypassing the `requiredSubmissions` threshold, undermining the fault-tolerance properties of the system.

Recommendation: Checking if the behavior is expected. If it is expected, acknowledging the potential security risk. If it is unexpected, changing the logic to only allowing the owner to whitelist the signers.

QSP-6 Risky reliance on price oracles

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `base/LoansBase.sol`

Description: Using third-party smart contracts such as price oracles may impose security risks. The lending platform's security depends on how reliable the oracle implementation is.

Recommendation: We suggest clearly documenting the risks of reliance on third-party price feeds to end-users.

QSP-7 Discrepancy of `liquidationPayment(...)` with the documentation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `base/LendingPool.sol`

Description: In `LendingPool.sol`, `L176`: the method `liquidationPayment(...)` transfers tokens from the liquidator to the lending pool. The comment on `L172` says "Once a loan is liquidated, it transfers the amount in tokens **to** the liquidator address" which contradicts the actual behavior of the method - transferring **from** the liquidator address. In addition, `L174` says "liquidator address to receive the tokens", and the method `liquidationPayment(...)` is equivalent to `repay(...)` in terms of the actual behavior.

Recommendation: Fixing the comment to clarify the expected behavior. Fixing the `liquidationPayment` method accordingly.

QSP-8 Missing balance checks for transfers of token that are not fully ERC-20 compliant

Severity: *Low Risk*

Status: Fixed

File(s) affected: `base/TokenCollateralLoans.sol`

Description: In `base/TokenCollateralLoans.sol`, the checks in `L161` and `L178` in `contracts\base\TokenCollateralLoans.sol` are insufficient because not all ERC20 contacts correctly implement the return value. Some of them may simply return `true` without actually checking the success of a transfer operation.

Recommendation: Implement a balance check right after lines `L161` and `L178` to guarantee the success of the transfer.

QSP-9 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `base/Settings.sol`

Description: Smart contracts will often have an owner-like variable to designate the person with special privileges to make modifications to the smart contract.
`Settings.sol` is fully controlled by the "Pauser" which is the Teller team. Functions like `setMaximumTolerance` can be misused to lower tolerance in order to temporarily break the system's security.

Recommendation: Clearly documenting the privileged roles in the user documentation. Being aware of the potential risks of the private keys leaking to the public (e.g., through social engineering). Considering a recovery mechanism in case a private key is lost.

QSP-10 `isWithinTolerance(...)` not working for small positive values

Severity: *Low Risk*

Status: Fixed

File(s) affected: `utils/NumbersList.sol`

Description: In `utils/NumbersList.sol`, in the function `isWithinTolerance(...)`, the call `minTolerance = average.sub(toleranceAmount)` (`L60`) would revert in case the `threshold`parameter is above 100%.

Recommendation: Adding input validation for threshold in the `Settings` contract.

QSP-11 Block Timestamp Manipulation

Severity: *Low Risk*

Status: Acknowledged

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.
`now` is used in `LoansBase.sol`, `Consensus.sol`. This allows miners to manipulate timestamps in order to:

1. Take out a loan later than expected (i.e., with expired loan terms)
2. Liquidate a loan earlier (i.e., before its expiry)

Recommendation:

1. Acknowledging and clearly documenting the risks associated with timestamp manipulation.
2. Confirming that the timestamp granularity (+-15s) is acceptable for the current domain.

QSP-12 Lack of integer overflow checks in oracle computations

Severity: *Low Risk*

Status: Fixed

File(s) affected: `providers/chainlink/ChainlinkPairAggregator.sol`

Description: In `providers/chainlink/ChainlinkPairAggregator.sol`, `L73`: the `_normalizeResponse(...)` method performs unbounded exponentiation and multiplication without overflow checks.

Recommendation: It is recommended to add integer overflow checks or using `SafeMath` or an equivalent logic.

QSP-13 Gas Usage / `for` Loop Concerns

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `base/InterestConsensus.sol`, `base/LoanTermsConsensus.sol`

Description: Gas usage is the main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. In addition, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely.
In the consensus logic, there are iterations over multiple responses (`base/InterestConsensus.sol`, `L46` and `base/LoanTermsConsensus.sol`, `L51`):

```
for (uint256 i = 0; i < responses.length; i++) {
    _processResponse(request, responses[i], requestHash);
}
```

At a certain `responses.length`, a transaction may take too much gas to execute and will exceed the block gas limit. This imposes a hard limit on the number of nodes that are possible to use for consensus.

Recommendation: It is recommended to analyze the gas usage to determine the maximum number of `responses` that the contract is capable of handling. If this number is satisfactory, no action is needed. If it is too low, it is recommended to change the logic accordingly.
Update: the team conducted gas usage analysis and concluded that gas limits will not be exceeded for the number of responses the team is planning to use.

QSP-14 Method visibility is too open

Severity: *Informational*

Status: Acknowledged

File(s) affected: `base\InterestConsensus.sol`, `base/Lenders.sol`

Description: 1. In `base\InterestConsensus.sol`: `_hashResponse`, `_hashRequest` could be made `private`.

1. In `base\Lenders.sol`: `_areAddressesEqual` should be set to `private`.

Recommendation: It is suggested to adjust the method visibility to `private`.
Update: the Teller Protocol team explained that marking method visibility as "private" would lead to inability to use the methods in tests. In such case, we suggested not to make any changes.

QSP-15 Disparity between `ChainlinkPairAggregator` and `InverseChainlinkPairAggregator`

Severity: *Informational*

Status: Acknowledged

Description: An expected relationship between `ChainlinkPairAggregator` and `InverseChainlinkPairAggregator` is unclear, however, `L34`, `_inverseValue(...)` of the latter does not appear to be an inverse of `_normalizeResponse` of the former.

Recommendation: Clarifying the expected behavior and fixing the actual behavior as necessary.
Update: the Teller Protocol team clarified the behaviour, and the disparity is expected, since the values calculated in the two classes are not expected to be the mathematical inverses of one another.

Code Documentation

1. The codebase is lacking developer documentation in most of the contract files. It is recommended to add developer documentation, describing the purpose for each method, parameter, and return values.

2. The codebase is lacking overall specification and documentation. Details on how consensus works and interacts with Teller Protocol contacts should be made clear to the end-users. Additionally, there needs to be a specification for the off-chain components.

Adherence to Best Practices

There are several suggestions on improving best practices:

1. `L335` in `contracts\base\LoansBase.sol`: it is suggested to rename `requireCollateral` into `moreCollateralRequired`. **Fixed.**

2. `L411` in `base\LoansBase.sol`, `tokenValue` would be a more suitable variable name than `weiValue`. **Fixed.**

3. `L420` in `base\LoansBase.sol`, `weiValue` would be a more suitable variable name than `tokenValue`. **Fixed.**

4. `L29` in `util\NumbersList.sol` function name `totalValues` is recommended to change into `valuesCount` instead. **Fixed.**

5. `L49` in `util\NumbersList.sol` parameter name `tolerance` is suggested to change into `tolerancePercentage` instead. **Fixed.**

6. `function initialize` in `contracts\base\Consensus.sol` should validate that `settingAddress` isn't 0x0 by using function `requireNotEmpty`. **Not an issue.**

7. `L51` in `base\Consensus.sol` should use the function `requireNotEmpty` instead. **Fixed.**

8. `base\Base.sol`, the function `_initialize(...)` in `L64` would be a good practice to check if `settingsAddress` is also a contract (using OpenZeppelin utils). **Fixed.**

9. In `base/Lenders.sol`, `L118`: the function `withdrawInterest` the function should revert if the amount that is to be withdrawn is greater than the amount available to avoid unexpected behavior. **Fixed.**

10. `base\EtherCollateralLoans.sol`, `L25`: the public variable `collateralToken` is not used. **Not an issue** (it is not used, however, it serves the purpose of implementing the interface definition).

11. `base\LendingPool.sol` the function `lendingToken()` (specified in `LendingPoolInterface`) is not implemented in this contract. **Not an issue.**

12. `base\LoansBase.sol`, `repay(...)` function: adding a `require` statement to check `amount > 0` in the for consistency. **Fixed.**

13. `base\LoansBase.sol`, for consistency, use `AddressLib.sol` to check the non-zero addresses `L361-363`. **Fixed.**

14. `base\LoansBase.sol`, `L187`: the comment `// the pays at x% of collateral price` should be `// the liquidator pays x% of the collateral price`. **Fixed.**

15. `base\TokenCollateralLoans.sol`, the function `setLoanTerms` seems to create an entirely new loan (`L84`) instead of just setting the loan terms, it is recommended to give this function a name that better reflects the side effect. **Fixed.**

16. `base\Consensus.sol`, `L51`: `callerAddress != address(0)` should be `callerAddress.requireNotEmpty(...)` (for consistency, use `AddressLib`). **Fixed.**

17. `base\LoansBase.sol`, `L119`: the comment mentions `tokens`, however, `LoansBase` is used for Ether loans too. **Fixed.**

18. `base\LoansBase.sol`, `withdrawCollateral(...)` would succeed in case of the `0` amount, unclear if this is expected. Update: **Not an issue.**

19. A typo: `_processReponse` => `_processResponse` in `InterestConsensus.sol`. **Fixed.**

20. The `LoanRepaid` event in `base/LoansBase.sol`, `L241` is emitted even when the loan is not fully repaid, which could be misleading.
    Acknowledged: the behavior is intentional.

Test Results

Test Suite Results

All the current tests pass within a reasonable time.

```
  Contract: ERC20TransferTest
JSON file created at './build/soliditycoverage_1595684956492.json'.
    with _1_basic
       ✓ .transfer => user : Should be able transfer tokens.  (122ms)
    with _1_basic
```

```
      ✓ .transfer => user : Should NOT be able transfer tokens. MustFail . (54ms)

  Contract: AddressLibTest
    with _1_basic
      ✓ .isEmpty => user : Should be able to test whether address is empty or not.
    with _2_empty
      ✓ .isEmpty => user : Should be able to test whether address is empty or not.
    with _1_basic
      ✓ .isNotEmpty => user : Should be able to test whether address is empty or not.
    with _2_empty
      ✓ .isNotEmpty => user : Should be able to test whether address is empty or not.
    with _1_basic
      ✓ .requireNotEmpty => user : Should be able to require address is not empty.
    with _2_empty
      ✓ .requireNotEmpty => user : Should be able to require address is not empty.
    with _1_basic
      ✓ .requireEmpty => user : Should be able to require address is not empty.
    with _2_empty
      ✓ .requireEmpty => user : Should be able to require address is not empty.
    with _1_basic
      ✓ .isEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _2_equalEmpty
      ✓ .isEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _3_emptyAndNotEmpty
      ✓ .isEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _4_equal
      ✓ .isEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _1_basic
      ✓ .isNotEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _2_equalEmpty
      ✓ .isNotEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _3_emptyAndNotEmpty
      ✓ .isNotEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _4_equal
      ✓ .isNotEqualTo => user : Should be able to test whether two addresses are equal or not.
    with _1_basic
      ✓ .requireEqualTo => user : Should be able to require address is not empty.
    with _2_emptyEqual
      ✓ .requireEqualTo => user : Should be able to require address is not empty.
    with _3_notEqual
      ✓ .requireEqualTo => user : Should be able to require address is not empty.
    with _1_basic
      ✓ .requireNotEqualTo => user : Should be able to require address is not empty.
    with _2_basic
      ✓ .requireNotEqualTo => user : Should be able to require address is not empty.
    with _3_emptyEqual
      ✓ .requireNotEqualTo => user : Should be able to require address is not empty.
    with _4_notEmptyEqual
      ✓ .requireNotEqualTo => user : Should be able to require address is not empty.

  Contract: NumbersListTest
    with _1_basic
      ✓ .addValue => user : Should be able to add a new value.  (144ms)
    with _1_basic
      ✓ .getAverage => user : Should be able to get the average.  (154ms)
    with _2_roundDown
      ✓ .getAverage => user : Should be able to get the average.  (161ms)
    with _1_notFinalized
      ✓ .isFinalized => user : Should be able to test if it is finalized.  (156ms)
    with _2_finalized
      ✓ .isFinalized => user : Should be able to test if it is finalized.  (168ms)
    with _1_empty
      ✓ .min/max/sum/count => user : Should be able to get min/max/sum/count.
    with _2_1Item
      ✓ .min/max/sum/count => user : Should be able to get min/max/sum/count.  (86ms)
    with _3_3Items
      ✓ .min/max/sum/count => user : Should be able to get min/max/sum/count.  (175ms)
    with _4_5Items
      ✓ .min/max/sum/count => user : Should be able to get min/max/sum/count.  (144ms)
    with _5_10Items
      ✓ .min/max/sum/count => user : Should be able to get min/max/sum/count.  (422ms)
    with _1_empty
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.
    with _2_10min_10max_10avg_1tol
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (68ms)
    with _3_10min_30max_20avg_10tol
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (159ms)
    with _4_max_too_high
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (196ms)
    with _5_min_too_low
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (153ms)
    with _6_all_within_range
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (165ms)
    with _7_max_too_high_close
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (203ms)
    with _8_min_too_low_close
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (147ms)
    with _9_all_within_range_0_tolerance
      ✓ .isWithinTolerance => user : Should be able to get min/max/sum.  (153ms)

  Contract: BaseConstructorTest
    with _1_basic
      ✓ .new => user : Should be able to create a new instance.  (44ms)

  Contract: BaseInitializeTest
    with _1_valid
      ✓ .initialize => user : Should (or not) be able to initialize the new instance.  (158ms)
    with _2_settings_empty
      ✓ .initialize => user : Should (or not) be able to initialize the new instance. MustFail . (165ms)
    with _3_settings_not_contract
```

```
      ✓ .initialize => user : Should (or not) be able to initialize the new instance. MustFail . (170ms)

  Contract: BaseWhenNotPausedTest
    with _1_notPaused
      ✓ .whenNotPaused => user : Should (or not) be able to call function when it is/isnt paused.  (39ms)
    with _2_paused
      ✓ .whenNotPaused => user : Should (or not) be able to call function when it is/isnt paused. MustFail .
(83ms)
    with _1_notPaused
      ✓ .whenLendingPoolNotPaused => user : Should (or not) be able to call function when a lending address
is/isnt paused.  (43ms)
    with _2_paused
      ✓ .whenLendingPoolNotPaused => user : Should (or not) be able to call function when a lending address
is/isnt paused. MustFail . (87ms)

  Contract: BaseWhenPausedTest
    with _1_notPaused
      ✓ .externalWhenPaused => user : Should (or not) be able to call function when it is/isnt paused.
MustFail . (73ms)
    with _2_paused
      ✓ .externalWhenPaused => user : Should (or not) be able to call function when it is/isnt paused.  (73ms)
    with _1_notPaused
      ✓ .externalWhenNotPaused => user : Should (or not) be able to call function when a lending address
is/isnt paused. MustFail . (46ms)
    with _2_paused
      ✓ .externalWhenNotPaused => user : Should (or not) be able to call function when a lending address
is/isnt paused.  (71ms)

  Contract: ConsensusModifiersTest
    with _1_not_lenders
      ✓ .new => user : Should (or not) be able to call the function MustFail . (216ms)
    with _2_lenders
      ✓ .new => user : Should (or not) be able to call the function  (207ms)

  Contract: ConsensusSignatureValidTest
    with _1_hash1_signer_incorrect
      ✓ .new => user : Should return false if numbers are out of range  (69ms)
    with _2_hash1_signer_correct
      ✓ .new => user : Should return false if numbers are out of range  (108ms)
    with _3_hash1_r_s_swapped
      ✓ .new => user : Should return false if numbers are out of range  (161ms)
    with _4_hash2_signer_incorrect
      ✓ .new => user : Should return false if numbers are out of range  (70ms)
    with _5_hash2_signer_correct
      ✓ .new => user : Should return false if numbers are out of range  (83ms)
    with _6_hash2_send_wrong_hash
      ✓ .new => user : Should return false if numbers are out of range  (84ms)
    with _7_hash2_signer_not_signer_role
      ✓ .new => user : Should return false if numbers are out of range  (38ms)

  Contract: EstimateGasInterestConsensusProcessRequestTest
    with _1_response
      ✓ .new => user : Should accept/not accept a nodes response  (398ms)
    with _2_responses
      ✓ .new => user : Should accept/not accept a nodes response  (472ms)
    with _3_responses
      ✓ .new => user : Should accept/not accept a nodes response  (586ms)
    with _4_responses
      ✓ .new => user : Should accept/not accept a nodes response  (686ms)
    with _5_responses
      ✓ .new => user : Should accept/not accept a nodes response  (795ms)
    with _6_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1204ms)
    with _7_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1020ms)
    with _8_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1082ms)
    with _9_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1207ms)

  Contract: EstimateGasLoanTermsConsensusProcessRequestTest
    with _1_response
      ✓ .new => user : Should accept/not accept a nodes response  (501ms)
    with _2_responses
      ✓ .new => user : Should accept/not accept a nodes response  (633ms)
    with _3_response
      ✓ .new => user : Should accept/not accept a nodes response  (782ms)
    with _4_responses
      ✓ .new => user : Should accept/not accept a nodes response  (928ms)
    with _5_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1062ms)
    with _6_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1263ms)
    with _7_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1390ms)
    with _8_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1534ms)
    with _9_responses
      ✓ .new => user : Should accept/not accept a nodes response  (1731ms)

  Contract: EtherCollateralLoansCreateLoanWithTermsTest
    with _1_no_msg_value
      ✓ .createLoanWithTerms => user : Should able (or not) to set loan terms.  (303ms)
    with _2_with_msg_value
      ✓ .createLoanWithTerms => user : Should able (or not) to set loan terms.  (314ms)
    with _3_msg_value_collateral_param_not_match_1
      ✓ .createLoanWithTerms => user : Should able (or not) to set loan terms. MustFail . (167ms)
    with _4_msg_value_collateral_param_not_match_2
      ✓ .createLoanWithTerms => user : Should able (or not) to set loan terms. MustFail . (215ms)
    with _5_msg_value_collateral_param_not_match_3
```

```
      ✓ .createLoanWithTerms => user : Should able (or not) to set loan terms. MustFail . (159ms)

Contract: EtherCollateralLoansDepositCollateralTest
  with _1_borrower_mismatch
     ✓ .depositCollateral => user : Should able to deposit collateral.  (173ms)
  with _2_deposit_zero
     ✓ .depositCollateral => user : Should able to deposit collateral.  (179ms)
  with _3_deposit_more
     ✓ .depositCollateral => user : Should able to deposit collateral.  (206ms)
  with _4_incorrect_value
     ✓ .depositCollateral => user : Should able to deposit collateral.  (186ms)

Contract: EtherCollateralLoansGetBorrowerLoansTest
  with _1_valid_no_msg_value
     ✓ .getBorrowerLoans => user : Should able to get the borrower loan ids.  (238ms)
  with _2_valid_with_msg_value
     ✓ .getBorrowerLoans => user : Should able to get the borrower loan ids.  (255ms)

Contract: EtherCollateralLoansLiquidateTest
  with _1_loan_expired
     ✓ .liquidate => user : Should able to (or not) liquidate a loan.  (494ms)
  with _2_under_collateralised
     ✓ .liquidate => user : Should able to (or not) liquidate a loan.  (446ms)
  with  3 collateralised and loan duration expired
     ✓ .liquidate => user : Should able to (or not) liquidate a loan.  (485ms)
  with _4_doesnt_need_liquidating
     ✓ .liquidate => user : Should able to (or not) liquidate a loan. MustFail . (432ms)

Contract: EtherCollateralLoansRepayTest
  with _1_to_pay_more_than_owed
     ✓ .repay => user : Should able to repay your loan.  (303ms)
  with _2_to_pay_zero
     ✓ .repay => user : Should able to repay your loan.  (236ms)
  with _3_to_less_than_owed
     ✓ .repay => user : Should able to repay your loan.  (288ms)
  with _4_to_pay_exact_owed
     ✓ .repay => user : Should able to repay your loan.  (300ms)

Contract: EtherCollateralLoansTakeOutLoanTest
  with _1_max_loan_exceeded
     ✓ .takeOutLoan => user : Should able to take out a loan.  (224ms)
  with _2_loan_terms_expired
     ✓ .takeOutLoan => user : Should able to take out a loan.  (241ms)
  with _3_collateral_deposited_recently
     ✓ .takeOutLoan => user : Should able to take out a loan.  (232ms)
  with _4_more_collateral_needed
     ✓ .takeOutLoan => user : Should able to take out a loan.  (337ms)
  with _5_successful_loan
     ✓ .takeOutLoan => user : Should able to take out a loan.  (328ms)
  with _6_with_recipient
     ✓ .takeOutLoan => user : Should able to take out a loan.  (330ms)

Contract: EtherCollateralLoansWithdrawCollateralTest
  with _1_non_borrower
     ✓ .withdrawCollateral => user : Should able to withdraw collateral.  (313ms)
  with _2_withdraw_zero
     ✓ .withdrawCollateral => user : Should able to withdraw collateral.  (406ms)
  with _3_more_than_allowed
     ✓ .withdrawCollateral => user : Should able to withdraw collateral.  (397ms)
  with _4_less_than_allowed
     ✓ .withdrawCollateral => user : Should able to withdraw collateral.  (402ms)
  with _5_none_allowed
     ✓ .withdrawCollateral => user : Should able to withdraw collateral.  (363ms)

Contract: InitializeModifiersTest
  with _1_notInitialized
     ✓ .isNotInitialized => user : Should (or not) be able to create a new instance.  (39ms)
  with _2_initialized
     ✓ .isNotInitialized => user : Should (or not) be able to create a new instance. MustFail . (81ms)
  with _1_initialized
     ✓ .isInitialized => user : Should (or not) be able to create a new instance.  (81ms)
  with _2_notInitialized
     ✓ .isInitialized => user : Should (or not) be able to create a new instance. MustFail . (44ms)

Contract: InterestConsensus hashInterestRequest and hashReponse
  with _1_first_test_hashInterestRequest
     ✓ .new => user : Should correctly calculate the hash for a request  (62ms)
  with _2_first_test_hashInterestRequest
     ✓ .new => user : Should correctly calculate the hash for a request  (63ms)
  with _3_second_test_hashInterestRequest
     ✓ .new => user : Should correctly calculate the hash for a request  (63ms)
  with _1_first_test_hashInterestResponse
     ✓ .new => user : Should correctly calculate the hash for a response  (60ms)
  with _2_first_test_hashInterestResponse
     ✓ .new => user : Should correctly calculate the hash for a response  (66ms)
  with _3_second_test_hashInterestResponse
     ✓ .new => user : Should correctly calculate the hash for a response  (67ms)

Contract: InterestConsensusInitializeTest
  with _1_no_settings
     ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (258ms)
  with _2_no_lenders
     ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (274ms)
  with _3_all_provided
     ✓ .new => user : Should (or not) be able to create a new instance.  (235ms)

Contract: InterestConsensusProcessRequestTest
  with _1_insufficient_responses
     ✓ .new => user : Should accept/not accept a nodes response  (381ms)
  with _2_one_response_successful
```

```
      ✓ .new => user : Should accept/not accept a nodes response  (414ms)
  with _3_responses_just_over_tolerance
      ✓ .new => user : Should accept/not accept a nodes response  (762ms)
  with _4_responses_just_within_tolerance
      ✓ .new => user : Should accept/not accept a nodes response  (505ms)
  with _5_zero_tolerance
      ✓ .new => user : Should accept/not accept a nodes response  (440ms)
  with _6_two_responses_same_signer
      ✓ .new => user : Should accept/not accept a nodes response  (579ms)
  with _7_expired_response
      ✓ .new => user : Should accept/not accept a nodes response  (568ms)
  with _8_responses_invalid_response_chainid
      ✓ .new => user : Should accept/not accept a nodes response  (622ms)

Contract: InterestConsensusProcessResponseTest
  with _1_signer_already_submitted
      ✓ .new => user : Should accept/not accept a nodes response  (479ms)
  with _2_signer_nonce_taken
      ✓ .new => user : Should accept/not accept a nodes response  (521ms)
  with _3_response_expired
      ✓ .new => user : Should accept/not accept a nodes response  (500ms)
  with _4_signature_invalid
      ✓ .new => user : Should accept/not accept a nodes response  (422ms)
  with _5_first_valid_submission
      ✓ .new => user : Should accept/not accept a nodes response  (500ms)
  with _6_later_valid_submission
      ✓ .new => user : Should accept/not accept a nodes response  (506ms)
  with _7_first_valid_submission_ropsten
      ✓ .new => user : Should accept/not accept a nodes response  (500ms)

Contract: LendersInitializeTest
  with _1_basic
      ✓ .new => user : Should (or not) be able to create a new instance.  (50ms)
  with _2_notzTokenInstance
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (56ms)
  with _3_notLendingPoolInstance
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (51ms)
  with _4_notConsensusInstance
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (53ms)
  with _5_notzTokenInstance_notLendingPoolInstance
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (69ms)
  with _6_notSettingsInstance
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (57ms)

Contract: LendersModifiersTest
  with _1_zTokenSender
      ✓ .isZToken => user : Should able (or not) to call function with modifier isZToken.  (223ms)
  with _2_lendingPoolSender
      ✓ .isZToken => user : Should able (or not) to call function with modifier isZToken. MustFail . (238ms)
  with _3_consensusSender
      ✓ .isZToken => user : Should able (or not) to call function with modifier isZToken. MustFail . (250ms)
  with _4_notValidSender
      ✓ .isZToken => user : Should able (or not) to call function with modifier isZToken. MustFail . (221ms)
  with _1_zTokenSender
      ✓ .isLendingPool => user : Should able (or not) to call function with modifier isLendingPool. MustFail .
(232ms)
  with _2_lendingPoolSender
      ✓ .isLendingPool => user : Should able (or not) to call function with modifier isLendingPool.  (221ms)
  with _3_consensusSender
      ✓ .isLendingPool => user : Should able (or not) to call function with modifier isLendingPool. MustFail .
(236ms)
  with _4_notValidSender
      ✓ .isLendingPool => user : Should able (or not) to call function with modifier isLendingPool. MustFail .
(238ms)
  with _1_validAddress
      ✓ .isValid => user : Should able (or not) to call function with modifier isValid.  (229ms)
  with _2_invalidAddress
      ✓ .isValid => user : Should able (or not) to call function with modifier isValid. MustFail . (240ms)

Contract: LendersSetAccruedInterestTest
  with _1_gap_in_accrual
      ✓ .setAccruedInterest => user : Should able to set accrued interest.  (187ms)
  with _2_invalid_interval
      ✓ .setAccruedInterest => user : Should able to set accrued interest.  (187ms)
  with _3_invalid_request
      ✓ .setAccruedInterest => user : Should able to set accrued interest.  (254ms)
  with _4_valid_request
      ✓ .setAccruedInterest => user : Should able to set accrued interest.  (214ms)

Contract: LendersWithdrawInterestTest
  with _1_0Available_10Requested
      ✓ .withdrawInterest => user : Should able to withdraw interest.  (131ms)
  with _2_50Available_10Requested
      ✓ .withdrawInterest => user : Should able to withdraw interest.  (144ms)
  with _3_50Available_100Requested
      ✓ .withdrawInterest => user : Should able to withdraw interest.  (131ms)
  with _5_50Available_0Requested
      ✓ .withdrawInterest => user : Should able to withdraw interest.  (144ms)
  with _6_50Available_0Requested_noPermissions
      ✓ .withdrawInterest => user : Should able to withdraw interest.  (118ms)

Contract: LendingPoolCreateLoanTest
  with _1_basic
      ✓ .createLoan => user : Should able (or not) to create loan.  (163ms)
  with _2_notLoanSender
      ✓ .createLoan => user : Should able (or not) to create loan. MustFail . (137ms)
  with _3_transferFail
      ✓ .createLoan => user : Should able (or not) to create loan. MustFail . (285ms)
  with _4_compoundFails
      ✓ .createLoan => user : Should able (or not) to create loan. MustFail . (193ms)
```

```
Contract: LendingPoolDepositTest
  with _1_basic
     ✓ .deposit => user : Should able (or not) to deposit DAIs.  (258ms)
  with _2_notTransferFromEnoughBalance
     ✓ .deposit => user : Should able (or not) to deposit DAIs. MustFail . (272ms)
  with _3_notDepositIntoCompound
     ✓ .deposit => user : Should able (or not) to deposit DAIs. MustFail . (313ms)
  with _4_notMint
     ✓ .deposit => user : Should able (or not) to deposit DAIs. MustFail . (349ms)

Contract: LendingPoolInitializeTest
  with _1_basic
     ✓ .initialize => user : Should (or not) be able to create a new instance.  (118ms)
  with _2_notZdai
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (111ms)
  with _3_notDai
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (111ms)
  with _4_notLenderInfo
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (121ms)
  with _5_notLoanInfo
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (123ms)
  with _6_notCToken
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (118ms)
  with _7_notZdai_notLoanInfo
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (114ms)
  with _8_notDai_notLenderInfo
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (107ms)
  with _9_notSettings
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (125ms)

Contract: LendingPoolLiquidationPaymentTest
  with _1_basic
     ✓ .liquidationPayment => user : Should able (or not) to liquidate payment.  (186ms)
  with _2_transferFromFail
     ✓ .liquidationPayment => user : Should able (or not) to liquidate payment. MustFail . (211ms)
  with _3_notLoansSender
     ✓ .liquidationPayment => user : Should able (or not) to liquidate payment. MustFail . (135ms)
  with _4_compoundFail
     ✓ .liquidationPayment => user : Should able (or not) to liquidate payment. MustFail . (249ms)

Contract: LendingPoolRepayTest
  with _1_basic
     ✓ .repay => user : Should able (or not) to repay loan.  (209ms)
  with _2_notLoan
     ✓ .repay => user : Should able (or not) to repay loan. MustFail . (140ms)
  with _3_transferFail
     ✓ .repay => user : Should able (or not) to repay loan. MustFail . (195ms)
  with _4_compoundFail
     ✓ .repay => user : Should able (or not) to repay loan. MustFail . (266ms)

Contract: LendingPoolWithdrawInterestTest
  with _1_basic
     ✓ .withdrawInterest => user : Should able (or not) to withdraw the interest.  (319ms)
  with _2_basic
     ✓ .withdrawInterest => user : Should able (or not) to withdraw the interest. MustFail . (317ms)
  with _3_basic
     ✓ .withdrawInterest => user : Should able (or not) to withdraw the interest.  (321ms)
  with _4_transferFail
     ✓ .withdrawInterest => user : Should able (or not) to withdraw the interest. MustFail . (398ms)
  with _5_notEnoughBalance
     ✓ .withdrawInterest => user : Should able (or not) to withdraw the interest. MustFail . (353ms)
  with _6_notAddressEqual
     ✓ .withdrawInterest => user : Should able (or not) to withdraw the interest. MustFail . (306ms)

Contract: LendingPoolWithdrawTest
  with _1_basic
     ✓ .withdraw => user : Should able (or not) to withdraw DAIs.  (494ms)
  with _2_transferFail
     ✓ .withdraw => user : Should able (or not) to withdraw DAIs. MustFail . (562ms)
  with _3_compoundFail
     ✓ .withdraw => user : Should able (or not) to withdraw DAIs. MustFail . (574ms)
  with _1_basic
     ✓ .withdraw => user : Should able (or not) to withdraw DAIs.  (564ms)
  with _2_recipientNotEnoughZDaiBalance
     ✓ .withdraw => user : Should able (or not) to withdraw DAIs. MustFail . (588ms)
  with _3_recipientNotZDaiBalance
     ✓ .withdraw => user : Should able (or not) to withdraw DAIs. MustFail . (578ms)

Contract: LoanTermsConsensus hashRequest and hashReponse
  with _1_mainnet_first_test_hashRequest
     ✓ .hashRequest => user : Should correctly calculate the hash for a request  (60ms)
  with _2_ropsten_test_hashRequest
     ✓ .hashRequest => user : Should correctly calculate the hash for a request  (60ms)
  with _3_second_test_hashRequest
     ✓ .hashRequest => user : Should correctly calculate the hash for a request  (59ms)
  with _1_first_test_hashResponse
     ✓ .hashResponse => user : Should correctly calculate the hash for a response  (60ms)
  with _2_second_test_hashResponse
     ✓ .hashResponse => user : Should correctly calculate the hash for a response  (61ms)

Contract: LoanTermsConsensusProcessRequestTest
  with _1_insufficient_responses
     ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (388ms)
  with _2_one_response_successful
     ✓ .processRequest => user : Should accept/not accept node request/responses  (485ms)
  with _3_responses_just_over_tolerance
     ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (918ms)
  with _4_responses_just_within_tolerance
     ✓ .processRequest => user : Should accept/not accept node request/responses  (654ms)
  with _5_zero_tolerance
```

```
      ✓ .processRequest => user : Should accept/not accept node request/responses  (528ms)
  with _6_two_responses_same_signer
      ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (738ms)
  with _7_expired_response
      ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (901ms)
  with _8_signer_nonce_taken
      ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (608ms)
  with _9_responses_invalid_sig_chainid
      ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (774ms)
  with _10_borrower_nonce_taken
      ✓ .processRequest => user : Should accept/not accept node request/responses MustFail . (445ms)

Contract: LoanTermsConsensusProcessResponseTest
  with _1_signer_already_submitted
      ✓ .new => user : Should accept/not accept a nodes response  (587ms)
  with _2_signer_nonce_taken
      ✓ .new => user : Should accept/not accept a nodes response  (592ms)
  with _3_response_expired
      ✓ .new => user : Should accept/not accept a nodes response  (578ms)
  with _4_signature_invalid
      ✓ .new => user : Should accept/not accept a nodes response  (554ms)
  with _5_first_valid_submission
      ✓ .new => user : Should accept/not accept a nodes response  (478ms)
  with _6_later_valid_submission
      ✓ .new => user : Should accept/not accept a nodes response  (710ms)

Contract: LoansBaseConvertTokenToWeiTest
  with _1_usdc_price
      ✓ .convertTokenToWei => user : Should able to convert wei to token.  (149ms)
  with _1_dai_price
      ✓ .convertTokenToWei => user : Should able to convert wei to token.  (152ms)
  with _1_btc_price
      ✓ .convertTokenToWei => user : Should able to convert wei to token.  (161ms)
  with _1_mkr_price
      ✓ .convertTokenToWei => user : Should able to convert wei to token.  (161ms)

Contract: LoansBaseConvertWeiToTokenTest
  with _1_usdc_price
      ✓ .convertWeiToToken => user : Should able to convert wei to token.  (354ms)
  with _1_dai_price
      ✓ .convertWeiToToken => user : Should able to convert wei to token.  (145ms)
  with _1_btc_price
      ✓ .convertWeiToToken => user : Should able to convert wei to token.  (157ms)
  with _1_mkr_price
      ✓ .convertWeiToToken => user : Should able to convert wei to token.  (157ms)

Contract: LoansBaseGetCollateralInfoTest
  with _1_link_usd_response_8_token_18_inverse_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (629ms)
  with _2_link_usd_response_8_token_18_inverse_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (715ms)
  with _3_link_usd_response_8_token_18_inverse_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (751ms)
  with _4_usdc_eth_response_18_token_6_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (519ms)
  with _5_usdc_eth_response_18_token_6_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (673ms)
  with _6_usdc_eth_response_18_token_6_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (634ms)
  with _7_dai_eth_response_18_token_18_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (515ms)
  with _8_usdc_slink_response_12_token_6_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (622ms)
  with _9_usdc_slink_response_12_token_6_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (650ms)
  with _10_usdc_slink_response_12_token_6_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (967ms)
  with _11_usdc_tlink_response_12_token_6_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (620ms)
  with _12_usdc_tlink_response_12_token_6_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (762ms)
  with _13_usdc_tlink_response_12_token_6_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (804ms)
  with _14_usdc_tlink_response_12_token_10_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (661ms)
  with _15_usdc_tlink_response_12_token_11_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (626ms)
  with _16_usdc_tlink_response_12_token_11_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (621ms)
  with _17_usdc_tlink_response_12_token_11_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (739ms)
  with _18_usdc_tlink_response_12_token_11_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (767ms)
  with _19_usdc_tlink_response_8_token_10_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (639ms)
  with _20_usdc_tlink_response_8_token_10_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (726ms)
  with _21_usdc_tlink_response_8_token_10_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (824ms)
  with _22_usdc_tlink_response_8_token_10_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (638ms)
  with _23_usdc_tlink_response_8_token_10_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (731ms)
  with _24_usdc_tlink_response_8_token_10_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (765ms)
  with _25_usdc_tlink_response_8_token_10_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (875ms)
  with _26_usdc_tlink_response_8_token_10_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (747ms)
  with _27_usdc_tlink_response_8_token_10_repay_100
```

```
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (799ms)
    with _28_usdc_tlink_response_8_token_10_repay_0
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (646ms)
    with _29_usdc_tlink_response_8_token_10_repay_50
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (725ms)
    with _30_usdc_tlink_response_8_token_10_repay_100
      ✓ .getCollateralInfo => user : Should able to get collateral info from a loan.  (778ms)

  Contract: LoansBaseLendingTokenTest
    with _1_basic
      ✓ .lendingToken => user : Should able to get the lending token address.  (74ms)
    with _2_empty_address
      ✓ .lendingToken => user : Should able to get the lending token address.  (78ms)

  Contract: LoansBaseModifiersTest
    with _1_loanNonExistent
      ✓ .loanActive(loanID) => user : Should able (or not) to call function with modifier loanActive. MustFail
. (478ms)
    with _2_loanTermsSet
      ✓ .loanActive(loanID) => user : Should able (or not) to call function with modifier loanActive. MustFail
. (396ms)
    with _3_loanActive
      ✓ .loanActive(loanID) => user : Should able (or not) to call function with modifier loanActive.  (389ms)
    with  4 loanClosed
      ✓ .loanActive(loanID) => user : Should able (or not) to call function with modifier loanActive. MustFail
. (404ms)
    with _1_loanNonExistent
      ✓ .loanTermsSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
MustFail . (397ms)
    with _2_loanTermsSet
      ✓ .loanTermsSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
(399ms)
    with _3_loanActive
      ✓ .loanTermsSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
MustFail . (550ms)
    with _4_loanClosed
      ✓ .loanTermsSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
MustFail . (456ms)
    with _1_loanNonExistent
      ✓ .loanActiveOrSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
MustFail . (424ms)
    with _2_loanTermsSet
      ✓ .loanActiveOrSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
(398ms)
    with _3_loanActive
      ✓ .loanActiveOrSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
(408ms)
    with _4_loanClosed
      ✓ .loanActiveOrSet(loanID) => user : Should able (or not) to call function with modifier loanActive.
MustFail . (407ms)
    with _1_basic
      ✓ .isBorrower => user : Should able (or not) to call function with modifier isBorrower.  (368ms)
    with _2_not_borrower
      ✓ .isBorrower => user : Should able (or not) to call function with modifier isBorrower. MustFail .
(362ms)

  Contract: LoansBasePayLoanTest
    with _1_less_than_principal
      ✓ .payLoan => user : Should able to pay loan correctly.  (115ms)
    with _2_more_than_principal
      ✓ .payLoan => user : Should able to pay loan correctly.  (112ms)
    with _3_no_principal_left
      ✓ .payLoan => user : Should able to pay loan correctly.  (107ms)
    with _4_full_amount
      ✓ .payLoan => user : Should able to pay loan correctly.  (116ms)

  Contract: OwnerSignersRoleTest
    with _1_basic
      ✓ .addSigner => user : Should be able (or not) to add a new signer.  (41ms)
    with _2_empty_address
      ✓ .addSigner => user : Should be able (or not) to add a new signer. MustFail . (49ms)
    with _3_not_owner
      ✓ .addSigner => user : Should be able (or not) to add a new signer. MustFail . (54ms)
    with _4_signer_add_new_signer
      ✓ .addSigner => user : Should be able (or not) to add a new signer. MustFail . (264ms)
    with _1_basic
      ✓ .removeSigner => user : Should be able (or not) to remove a new signer.  (144ms)
    with _2_not_exist
      ✓ .removeSigner => user : Should be able (or not) to remove a new signer. MustFail . (54ms)
    with _3_empty_address
      ✓ .removeSigner => user : Should be able (or not) to remove a new signer. MustFail . (140ms)
    with _4_not_owner
      ✓ .removeSigner => user : Should be able (or not) to remove a new signer. MustFail . (132ms)
    with _5_signer_remove_signer
      ✓ .removeSigner => user : Should be able (or not) to remove a new signer. MustFail . (164ms)
    with _1_isSigner
      ✓ .isSigner => user : Should be able (or not) to remove a new signer.  (98ms)
    with _2_isNotSigner
      ✓ .isSigner => user : Should be able (or not) to remove a new signer.  (150ms)

  Contract: SettingsConstructorTest
    with _1_basic
      ✓ .new => user : Should (or not) be able to create a new instance.  (88ms)
    with _2_not_required_submissions
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (79ms)
    with _3_not_response_expiry_length
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (82ms)
    with _4_not_safety_interval
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (84ms)
    with _5_not_terms_expiry
```

```
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (80ms)
    with _6_not_eth_price
      ✓ .new => user : Should (or not) be able to create a new instance. MustFail . (83ms)

  Contract: SettingsPauseLendingPoolTest
    with _1_basic
      ✓ .pauseLendingPool => user : Should (or not) be able to pause a lending pool.  (54ms)
    with _2_notOwner
      ✓ .pauseLendingPool => user : Should (or not) be able to pause a lending pool. MustFail . (41ms)
    with _3_emptyLendingPool
      ✓ .pauseLendingPool => user : Should (or not) be able to pause a lending pool. MustFail . (41ms)
    with _1_basic
      ✓ .pauseLendingPool => user : Should NOT be able to pause a lending pool twice. MustFail . (79ms)

  Contract: SettingsSetSettingTest
    with _1_basic
      ✓ .setRequiredSubmissions => user : Should (or not) be able to set a new required submissions value.
(53ms)
    with _2_zero
      ✓ .setRequiredSubmissions => user : Should (or not) be able to set a new required submissions value.
MustFail . (54ms)
    with _3_new_value_required
      ✓ .setRequiredSubmissions => user : Should (or not) be able to set a new required submissions value.
MustFail . (51ms)
    with _1_basic
      ✓ .setMaximumTolerance => user : Should (or not) be able to set a new maximum tolerance value.  (50ms)
    with _2_zero
      ✓ .setMaximumTolerance => user : Should (or not) be able to set a new maximum tolerance value.  (50ms)
    with _3_new_value_required
      ✓ .setMaximumTolerance => user : Should (or not) be able to set a new maximum tolerance value. MustFail
. (59ms)
    with _4_basic_2
      ✓ .setMaximumTolerance => user : Should (or not) be able to set a new maximum tolerance value.  (55ms)
    with _5_max_exceeded
      ✓ .setMaximumTolerance => user : Should (or not) be able to set a new maximum tolerance value. MustFail
. (141ms)
    with _6_max_limit
      ✓ .setMaximumTolerance => user : Should (or not) be able to set a new maximum tolerance value.  (55ms)
    with _1_basic
      ✓ .setResponseExpiryLength => user : Should (or not) be able to set a new response expiry length value.
(54ms)
    with _2_zero
      ✓ .setResponseExpiryLength => user : Should (or not) be able to set a new response expiry length value.
MustFail . (56ms)
    with _3_new_value_required
      ✓ .setResponseExpiryLength => user : Should (or not) be able to set a new response expiry length value.
MustFail . (59ms)
    with _1_basic
      ✓ .setSafetyInterval => user : Should (or not) be able to set a new safety interval value.  (54ms)
    with _2_zero
      ✓ .setSafetyInterval => user : Should (or not) be able to set a new safety interval value. MustFail .
(56ms)
    with _3_new_value_required
      ✓ .setSafetyInterval => user : Should (or not) be able to set a new safety interval value. MustFail .
(55ms)
    with _1_basic
      ✓ .setTermsExpiryTime => user : Should (or not) be able to set a new terms expiry time value.  (53ms)
    with _2_zero
      ✓ .setTermsExpiryTime => user : Should (or not) be able to set a new terms expiry time value. MustFail .
(54ms)
    with _3_new_value_required
      ✓ .setTermsExpiryTime => user : Should (or not) be able to set a new terms expiry time value. MustFail .
(60ms)
    with _1_basic
      ✓ .setLiquidateEthPrice => user : Should (or not) be able to set a new liquidate ETH price value.
(55ms)
    with _2_zero
      ✓ .setLiquidateEthPrice => user : Should (or not) be able to set a new liquidate ETH price value.
MustFail . (58ms)
    with _3_new_value_required
      ✓ .setLiquidateEthPrice => user : Should (or not) be able to set a new liquidate ETH price value.
MustFail . (64ms)

  Contract: SettingsUnpauseLendingPoolTest
    with _1_basic
      ✓ .unpauseLendingPool => user : Should (or not) be able to pause a lending pool.  (87ms)
    with _2_notOwner
      ✓ .unpauseLendingPool => user : Should (or not) be able to pause a lending pool. MustFail . (74ms)
    with _3_emptyLendingPool
      ✓ .unpauseLendingPool => user : Should (or not) be able to pause a lending pool. MustFail . (67ms)
    with _1_notPausedTwice
      ✓ .unpauseLendingPool => user : Should NOT be able to unpause a lending pool twice. MustFail . (121ms)

  Contract: TokenCollateralLoansCreateLoanWithTermsTest
    with _1_without_collateral
      ✓ .createLoanWithTerms => user : Should able to set loan terms.  (389ms)
    with _2_basic_collateral
      ✓ .createLoanWithTerms => user : Should able to set loan terms.  (422ms)
    with _3_not_enough_balance
      ✓ .createLoanWithTerms => user : Should able to set loan terms. MustFail . (432ms)
    with _4_not_enough_allowance
      ✓ .createLoanWithTerms => user : Should able to set loan terms. MustFail . (2464ms)

  Contract: TokenCollateralLoansDepositCollateralTest
    with _1_deposit_basic
      ✓ .depositCollateral => user : Should able (or not) to deposit tokens as collateral.  (491ms)
    with _2_borrower_loan_mismatch
      ✓ .depositCollateral => user : Should able (or not) to deposit tokens as collateral.  (408ms)
    with _3_deposit_zero
      ✓ .depositCollateral => user : Should able (or not) to deposit tokens as collateral.  (428ms)
```

```
  with _4_deposit_more_value_not_zero
     ✓ .depositCollateral => user : Should able (or not) to deposit tokens as collateral.  (360ms)
  with _5_value_not_zero
     ✓ .depositCollateral => user : Should able (or not) to deposit tokens as collateral.  (368ms)
  with _1_deposit_basic
     ✓ .depositCollateral#2 => user : Should able (or not) to deposit tokens as collateral.  (467ms)
  with _2_not_enough_allowance
     ✓ .depositCollateral#2 => user : Should able (or not) to deposit tokens as collateral.  (415ms)
  with _3_transferFrom_failed
     ✓ .depositCollateral#2 => user : Should able (or not) to deposit tokens as collateral.  (497ms)

Contract: TokenCollateralLoansInitializeTest
  with _1_basic
     ✓ .initialize => user : Should (or not) be able to create a new instance.  (169ms)
  with _2_not_oracle
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (168ms)
  with _3_not_lendingpool
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (169ms)
  with _4_not_loanTerms
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (176ms)
  with _5_not_settings
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (175ms)
  with _6_not_collateralToken
     ✓ .initialize => user : Should (or not) be able to create a new instance. MustFail . (164ms)

  Contract: TokenCollateralLoansRequireExpectedBalanceTest
    with _1_valid_transferFrom
       ✓ .requireExpectedBalance => user : Should able to check balance after transferring/from.  (76ms)
    with _2_valid_transfer
       ✓ .requireExpectedBalance => user : Should able to check balance after transferring/from.  (81ms)
    with _3_invalid_transferFrom
       ✓ .requireExpectedBalance => user : Should able to check balance after transferring/from. MustFail .
(85ms)
    with _4_invalid_transfer
       ✓ .requireExpectedBalance => user : Should able to check balance after transferring/from. MustFail .
(78ms)
    with _5_overflow_transferFrom
       ✓ .requireExpectedBalance => user : Should able to check balance after transferring/from. MustFail .
(86ms)
    with _6_invalid_transfer
       ✓ .requireExpectedBalance => user : Should able to check balance after transferring/from. MustFail .
(82ms)

  Contract: TokenCollateralLoansWithdrawCollateralTest
    with _1_more_than_allowed
       ✓ .withdrawCollateral => user : Should able to withdraw collateral (tokens).  (595ms)
    with _2_non_borrower
       ✓ .withdrawCollateral => user : Should able to withdraw collateral (tokens). MustFail . (452ms)
    with _3_withdraw_zero
       ✓ .withdrawCollateral => user : Should able to withdraw collateral (tokens). MustFail . (513ms)
    with _1_valid
       ✓ .withdrawCollateral#2 => user : Should able (or not) to withdraw collateral (tokens).  (558ms)
    with _2_not_enough_balance
       ✓ .withdrawCollateral#2 => user : Should able (or not) to withdraw collateral (tokens).  (623ms)
    with _3_transfer_fail
       ✓ .withdrawCollateral#2 => user : Should able (or not) to withdraw collateral (tokens).  (654ms)

Contract: ChainlinkPairAggregatorConstructorTest
  with _1_basic
     ✓ .new => user : Should (or not) able to create a new instance.  (64ms)
  with _2_emptyChainlink
     ✓ .new => user : Should (or not) able to create a new instance. MustFail . (61ms)
  with _3_zeroResponseDecimals
     ✓ .new => user : Should (or not) able to create a new instance.  (60ms)
  with _4_zeroCollateralDecimals
     ✓ .new => user : Should (or not) able to create a new instance.  (53ms)
  with _5_zeroDecimals
     ✓ .new => user : Should (or not) able to create a new instance.  (55ms)
  with _6_big_diff
     ✓ .new => user : Should (or not) able to create a new instance. MustFail . (66ms)

Contract: ChainlinkPairAggregatorGetLatestAnswerTest
  with _1_coll18_response18
     ✓ .getLatestAnswer => user : Should able to get the last price.  (126ms)
  with _2_coll18_response10
     ✓ .getLatestAnswer => user : Should able to get the last price.  (124ms)
  with _3_coll18_response18
     ✓ .getLatestAnswer => user : Should able to get the last price.  (130ms)
  with _4_coll10_response18
     ✓ .getLatestAnswer => user : Should able to get the last price.  (128ms)

Contract: ChainlinkPairAggregatorGetLatestRoundTest
  with _1_basic
     ✓ .getLatestRound => user : Should able to get the last round.  (70ms)

Contract: ChainlinkPairAggregatorGetLatestTimestampTest
  with _1_basic
     ✓ .getLatestTimestamp => user : Should able to get the last timestamp.  (70ms)
  with _2_now
     ✓ .getLatestTimestamp => user : Should able to get the last timestamp.  (72ms)

Contract: ChainlinkPairAggregatorGetPreviousAnswerTest
  with _1_basic
     ✓ .getPreviousAnswer => user : Should able (or not) to get the previous price.  (182ms)
  with _2_zeroDays
     ✓ .getPreviousAnswer => user : Should able (or not) to get the previous price.  (126ms)
  with _3_notHistory
     ✓ .getPreviousAnswer => user : Should able (or not) to get the previous price. MustFail . (120ms)

Contract: ChainlinkPairAggregatorGetPreviousTimestampTest
```

```
  with _1_basic
     ✓ .getPreviousTimestamp => user : Should able (or not) to get the previous timestamp.  (127ms)
  with _2_zeroDays
     ✓ .getPreviousTimestamp => user : Should able (or not) to get the previous timestamp.  (135ms)
  with _2_notHistory
     ✓ .getPreviousTimestamp => user : Should able (or not) to get the previous timestamp. MustFail . (123ms)

Contract: InverseChainlinkPairAggregatorGetLatestAnswerTest
  with _1_link_usd
     ✓ .getLatestAnswer => user : Should able to get the last price.  (131ms)
  with _2_usd_tokenB
     ✓ .getLatestAnswer => user : Should able to get the last price.  (124ms)
  with _3_usd_tokenB
     ✓ .getLatestAnswer => user : Should able to get the last price.  (134ms)
  with _4_usd_tokenC
     ✓ .getLatestAnswer => user : Should able to get the last price.  (128ms)
  with _5_usd_tokenD
     ✓ .getLatestAnswer => user : Should able to get the last price.  (132ms)

Contract: InverseChainlinkPairAggregatorGetPreviousAnswerTest
  with _1_link_usd
     ✓ .getPreviousAnswer => user : Should able (or not) to get a previous price.  (269ms)
  with _2_usd_tokenB
     ✓ .getPreviousAnswer => user : Should able (or not) to get a previous price. MustFail . (211ms)
  with _3_usd_tokenB
     ✓ .getPreviousAnswer => user : Should able (or not) to get a previous price. MustFail . (164ms)
  with _4_usd_tokenC
     ✓ .getPreviousAnswer => user : Should able (or not) to get a previous price.  (177ms)


 386 passing (5m)
```

Code Coverage

Test coverage appears to be very high as measured by the coverage tools. However, we have the following suggestions:

1. Adding coverage for the lines `96` and `100` in `LoansBase.sol`. The uncovered methods are `external`, and therefore, it is important to ensure their correctness.

2. Bringing the branch coverage of `LoanTermsConcensus` to `100%` by covering the "else" branch of `REQUEST_NONCE_TAKEN` at `L43`.

3. Bringing the branch coverage of `EtherCollateralLoans` to `100%` by covering the "else" branch of `INCORRECT_ETH_AMOUNT` at `L58`.

Additionally, we recommend adding tests that cover the edge cases, including access control logic (`QSP-1`).

Update: Test coverage has been improved as of the commit `dd7d2ac`.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
base/	100	100	100	100	
Base.sol	100	100	100	100	
Consensus.sol	100	100	100	100	
EtherCollateralLoans.sol	100	100	100	100	
Initializable.sol	100	100	100	100	
InterestConsensus.sol	100	100	100	100	
Lenders.sol	100	100	100	100	
LendingPool.sol	100	100	100	100	
LoanTermsConsensus.sol	100	100	100	100	
LoansBase.sol	100	100	100	100	
OwnerSignersRole.sol	100	100	100	100	
Settings.sol	100	100	100	100	
TokenCollateralLoans.sol	100	100	100	100	
ZDai.sol	100	100	100	100	
ZToken.sol	100	100	100	100	
ZUSDC.sol	100	100	100	100	
interfaces/	100	100	100	100	
InterestConsensusInterface.sol	**100**	**100**	**100**	**100**	
LendersInterface.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
LendingPoolInterface.sol	100	100	100	100	
LoanTermsConsensusInterface.sol	100	100	100	100	
LoansInterface.sol	100	100	100	100	
PairAggregatorInterface.sol	100	100	100	100	
SettingsInterface.sol	100	100	100	100	
ZTokenInterface.sol	100	100	100	100	
providers/chainlink/	100	100	100	100	
ChainlinkPairAggregator.sol	100	100	100	100	
InverseChainlinkPairAggregator.sol	100	100	100	100	
providers/compound/	100	100	100	100	
CErc20Interface.sol	100	100	100	100	
util/	100	100	100	100	
AddressLib.sol	100	100	100	100	
NumbersList.sol	100	100	100	100	
ZeroCollateralCommon.sol	100	100	100	100	
All files	**100**	**100**	**100**	**100**	

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

```
adbfaf8714e4372a20e8460a48b8d10b48c120842145a4cdc869e93157528e02 ./contracts/Migrations.sol

9f70cabd5bc33ea4b600fba0fe60f622bd8af6a647a05eba24789c1219b5db36 ./contracts/interfaces/ZTokenInterface.sol

a98b382279b70c1e3d6a4cd9800ee908031fe7f34574f5e86b16a8a65bfbe103 ./contracts/interfaces/LoansInterface.sol

3a142240d99a184ce553827c3f0aab133081f5886f18f36d0df6238735dc949a
./contracts/interfaces/PairAggregatorInterface.sol

4ec1f07b8ceb14a581e0311fcf56cc4d4454341515a4360b9b48897a9defa7e3
./contracts/interfaces/InterestConsensusInterface.sol

77c84736526497e371674d4f5ed878b029ed58e60a42207e99783dfc65854941
./contracts/interfaces/LoanTermsConsensusInterface.sol

8da3b3de2b6d093210804f8153d1caa3b11053b68980a3a0273caf7381fd54ed ./contracts/interfaces/SettingsInterface.sol

0ce0b3ac35f10f1312f8748ff7f416c7c6e7f5dc5e0e83ae61a6255ed869d8db ./contracts/interfaces/LendersInterface.sol

dcf0dd5cfedbe036f63beb0adbb26b89c3f741e72c14f5c874829209bd9f07c6 ./contracts/interfaces/LendingPoolInterface.sol

21bd49f28399ffd05ccef339c8c0765433f483813a1d3179c2fb38cb8876537b ./contracts/base/Initializable.sol

b10b6c7d93b4bce9a2c36c357b4473134a2865b0f0445337bd2d43d43c5768c8 ./contracts/base/Settings.sol

d56a6a878f933d7c5cdb7b62c4a556e46c99777376a1e412886947882b62eb2a ./contracts/base/LoansBase.sol

37c1c6d4ee51a0f2ae8e1b5abd90b8ac970a1ebf6b9eeb6b806ecc7ceb0d4c94 ./contracts/base/LendingPool.sol

3c10e8e25d51eb3a8660a2cce1eb7688272448d3b8aa86c3f9d2dca9ea45ad15 ./contracts/base/ZUSDC.sol

b1af44b97b5d84f39fd82c3ce99dcac3168d752cc2be2657f012af212f403a83 ./contracts/base/Lenders.sol

073d0a38252d425892c1c59cd16748e74467322e3ec0556e1574628327ef0c13 ./contracts/base/OwnerSignersRole.sol

35782474a7c94b67327e65e2e8b8767fe118dc34002b7896b4b6092caa6d18b9 ./contracts/base/ZToken.sol

d1e9e5900ff7af5a8d3ab8965cad6041726d613e026027d19914eb4f30c98df7 ./contracts/base/LoanTermsConsensus.sol

9539d466e6280a02dd6990c7efb0f91ac8e7acb7ab0de458e03a2c2770a805a2 ./contracts/base/ZDai.sol

e43e5901d304aaef68eff6468d378deff66c76917c90ade6b67f4b0bee2ee0ba ./contracts/base/Base.sol

41c1986657880e490ee6d5afc6e50865d144cc1777e136eb20e5b5d0f7bf3819 ./contracts/base/TokenCollateralLoans.sol

704e2f3b1e0850d7665eed4f1bd4830905b5ef72010cd2f8ff0aec7a45b34c37 ./contracts/base/InterestConsensus.sol

2c50c0dadde447c439f91455ce821d81d842787c3e555d29dce23e4dcca138d8 ./contracts/base/EtherCollateralLoans.sol
```

abaa99cb3345188f3d1ca76cb2eda8d71bfb4c22e68789cad7116e4abb1de5c2 ./contracts/base/Consensus.sol

923bc84f8c33f194828974829e515c99b6387ab2b82e3e8e1b6f0a65c5766b57 ./contracts/mock/base/BaseMock.sol

86c7a854974a2e6120dc9bece4871d8d4c87ef10622ac62171c8492b88c04399 ./contracts/mock/base/LendersMock.sol

1f03bb3e2f4c965ed02d72be69045f42793c6c4ab18fadcc16317975d9432804
./contracts/mock/base/InitializableModifiersMock.sol

00c185eedbf6fb3bfbc252482fac58eb267df04ae07a4e16535bf10a6cb59e08 ./contracts/mock/base/LoansBaseMock.sol

0b549d608754113bb3b790a15b5c5e5e32550669a2a384a7f93c96d8f0aafcbc
./contracts/mock/base/LoanTermsConsensusMock.sol

6c9b2db8addbc3af2c3236b1232efb6d2677e726999a1a93a1fffd5321797559 ./contracts/mock/base/InterestConsensusMock.sol

e04f5bdfd11cf75eb209f2f4acce52f5febc336091ae911a7a5f83b9d63531fe
./contracts/mock/base/ConsensusModifiersMock.sol

cb4f0ad722267861ced6cf3b65928e59525cec402b0dd85b972764930b75610f
./contracts/mock/base/EtherCollateralLoansMock.sol

67d3e81521221f9dabc23f81e42edc97ff9946fbe5e854abd923cc2ce3368354
./contracts/mock/base/LoansBaseModifiersMock.sol

4c8d49a413d7e71bb85451f18244d7f728b161c0a9bf22193340c3275b1359e5
./contracts/mock/base/TokenCollateralLoansMock.sol

45906d964fcb2f24543ec77c6607692c97d1b4abbaf2ac49e69fdfdfad7eb4f9 ./contracts/mock/base/ConsensusMock.sol

b992ecd8fe653513aa830657dace8f8dd9108a6d73739825171ba2d2b6f5a5b1 ./contracts/mock/base/LendingPoolMock.sol

36fa0e71f982f46abadabac77112cdd072ec1016c0258bdf9ee7fd89babbf96e ./contracts/mock/base/LendersModifiersMock.sol

2a3d4817f9bae7f701fa6291a798b2f82ec4d88a18b0e3a92f2622c2e7e86910
./contracts/mock/providers/compound/CUSDCMock.sol

26b130c296effc659bafa5652930405515c59887dc193f19e9ee97c1df2311f7
./contracts/mock/providers/compound/CERC20Mock.sol

47ac176fb886a1411520faf3aa1884167e06812e1c5da2f9bf5ea23df5df1ecf
./contracts/mock/providers/compound/CDAIMock.sol

05d8bab01f6e97d9cf8c83bf372b8fcb1ff8ab3dff060438568b7571f89f1037
./contracts/mock/providers/chainlink/PairAggregatorMock.sol

e1357b0c45e3a9c0d66ad52d9b9956637b4f8db509cb39fa3fae20e3be4b5b7c ./contracts/mock/util/AddressLibMock.sol

6d625c1418459d7409d1412cf13fb5a4e458e752d42e8ce28ebf2f42f89bdd47 ./contracts/mock/util/NumbersListMock.sol

1b167ceed84907baf3a6c7c820fb0604efda92c6540a51597dd478674e021a1f ./contracts/mock/util/Mock.sol

bc14e1c00b13e5e2482bfdaf6e1845a5587949bd6fe16cfe91212ee66cbc62aa ./contracts/mock/token/LINKMock.sol

62097eed10fec9062a6e7498709027f864ebcde885876136410c4dfe8121b130 ./contracts/mock/token/ERC20Mock.sol

54c8bb35727984a6ee93c045f4c3432994caa749a88e13bab68912bb82b9e9eb ./contracts/mock/token/USDCMock.sol

6ba5dcc4037e85887f5cffffbb1659e4ecd9fbd1de865b924d64cc3371a655c1 ./contracts/mock/token/DAIMock.sol

ed680d1b6ef57dbc58bf73dcdca1542c59b0bf68ffcfdee10bc5462aca20460a
./contracts/providers/openzeppelin/SignedSafeMath.sol

a20f393c83bdf50cb774dcab411cdc558a20323bf92ccfe8e920b850da626bac
./contracts/providers/compound/CErc20Interface.sol

1713e9e1ca65a86ab6fbc41708f57cfb34b4e207d7e8d7c15869ea6ba4282b24
./contracts/providers/chainlink/ChainlinkPairAggregator.sol

170aa5a0fd5401c6b64e69666ef0d42618c69449b26a4b99e3ac2d432e1d99da
./contracts/providers/chainlink/InverseChainlinkPairAggregator.sol

54d15dc7699e0947f7030dbad1c236ee0d1483760050f4e43e6307d3b7dba8c1 ./contracts/util/NumbersList.sol

0d139145bafc1efa7f6a200bc90884c7b0f3002e567e88cbbd3e58cc6cc476b7 ./contracts/util/AddressLib.sol

fb39b324a3eff012f5d548de7bc016ec0a1e420d045df5b4d8827a56f7386c32 ./contracts/util/ZeroCollateralCommon.sol

Tests

ba632aa94ecf94ffa6705ecd757a9685e8594726f8eb0fee055c4ed26516a0ca ./test/ERC20TransferTest.js

da83a3c5720334204634b8fad0817f59eb84a4383c09c4d3af961daf4d9d08da ./test/base/LendersModifiersTest.js

ded2b00396c4de37a93a691956065d38c7e90a69c03bef2ea1b63bed72831dba ./test/base/InterestConsensusHashesTest.js

ecb051e372bb3012f0d2bee124083c8db8cec2ace206641fb2f7dbdb133ed16d ./test/base/LendingPoolRepayTest.js

1cee3c3dfcbb294dda21b6386fa810b2a2f63384348abccde4f940c9cf02ffad ./test/base/LoansBaseGetCollateralInfoTest.js

7c5e13f7a0fa876b65961cc42e382e2214df846fb38dcd5fd5bdce3e3c1c3b12
./test/base/EtherCollateralLoansCreateLoanWithTermsTest.js

befd460a75a0729ff07814ee5acd2d6d5d2f525533612b72f6b335f92c6690a6 ./test/base/ConsensusSignatureValidTest.js

65ed95826b813cbb30d3e11f9f4da065b771d7349c394f956373b3128980c398

./test/base/TokenCollateralLoansInitializeTest.js

f9cb458ba04edf8b745f2a65f7b954a0d575abe084375330b5effb14c85d4362
./test/base/EtherCollateralLoansLiquidateTest.js

c8f5f03e58c859bbf5c11aaab2ba0a3a6cf0ecc07753371883c0fae6aec21ad8 ./test/base/ConsensusModifiersTest.js

b890a69febaef49b6df2b5f639b2830b570236352e174f926652058f90201000 ./test/base/SettingsSetSettingTest.js

f150411e2c03a2df3be43c03e3bd4fdaa73129f37a65cf108d756b9dd8511e06 ./test/base/LoansBasePayLoanTest.js

56a63aef3dba9ea1f8f23228f76f211d1e478e2868245a3b11e84d9f24f08bfc
./test/base/LoanTermsConsensusProcessResponseTest.js

7e271424b253b957cd7bdf95916fc77cb923b9d92a022bd97c7bf830d9df0af5 ./test/base/LendingPoolCreateLoanTest.js

c9c22b64395ae242034c94e9331a3c54d251470d0d907a6d656c345eebeb1fdb ./test/base/SettingsPauseLendingPoolTest.js

2919ac7a6645a14dfc70c7596f8f9a0a916fee9517f999082c2dea099ecb0af6
./test/base/LoanTermsConsensusProcessRequestTest.js

943db234a9390d69bcf7bea923047684917500506c1103a4823addf8b2ec97cd
./test/base/LendingPoolLiquidationPaymentTest.js

5f93ce7cd29f0d4481eb602a2463f005cf17c0031ef166df3f32a80b38c1d340 ./test/base/LoansBaseModifiersTest.js

6d017a5a3f3b79af6eec519d5f597afd1591be318a3037966487f49dd1e49170 ./test/base/LendersSetAccruedInterestTest.js

ee22af272035acece5bcfb5674d14bca59a46fd5838b4fec1217fbbe500e7346
./test/base/EtherCollateralLoansWithdrawCollateralTest.js

93aaa8cbf2f6ab559cd46f0aae47c83eacd1b371fb30b6da62532158f18fa362 ./test/base/BaseWhenNotPausedTest.js

94646b99f7a73410e42c068c6d164198795eb41edb71151af81eced898836a56 ./test/base/OwnerSignersRoleTest.js

23b4c153e55597bbd74f90cf1c6a50344a0455fb48bca67b7813a5958bd1ed35
./test/base/TokenCollateralLoansWithdrawCollateralTest.js

a0303c185979be023c498dfac3ae7ab7b1b46b900884010f0b223f7593c1ea18
./test/base/EtherCollateralLoansTakeOutLoanTest.js

c6c128ba03ab5422485ce7ed78fdfba35079f560e96c7796e19308d477c053a9
./test/base/EtherCollateralLoansDepositCollateralTest.js

5b2cd73889055864990f8c5fd6f717d2413578a0d6425ae26b58bc128d66ccd7
./test/base/TokenCollateralLoansDepositCollateralTest.js

c404da20c8bb59d392c2b706180bc039ab36b644522133bfde42b6c0fe39816c ./test/base/BaseWhenPausedTest.js

bf06d742e6c9297ca17cdc40ff22a14245417254defdc7ede60650fb9669a6d6
./test/base/TokenCollateralLoansCreateLoanWithTermsTest.js

30bdb4a0513aa7b2be6cfe16822e05a9b79801495b23deefe450481d01b0b689
./test/base/EtherCollateralLoansGetBorrowerLoansTest.js

434fecb4fef9af687962d6685e917ad841e7522f8bdc3f48112b954a3cf0bfed ./test/base/LendingPoolInitializeTest.js

c97cc81987daa5f6767d8d3718841181313667e3a3093ef5d14d9be1c284b1d6 ./test/base/InitializeModifiersTest.js

a51be2104281c089d0dc8c46674dee1b533df608c78aa29e1b611f9000bd8fe9 ./test/base/SettingsConstructorTest.js

24a6918a673d44824cfafb1240c3c2c09b045441ecc9eeb9a6d19cd9116fdc06 ./test/base/LoansBaseConvertWeiToTokenTest.js

21bda5d358db6e600cd61ba812705c86d64bbcea02502ad75a77383cc670d5ee ./test/base/LendingPoolWithdrawInterestTest.js

26439cb3b7db54fdb1a4f70534eb6da7f11ce74753108e833d0ed5a141b978b5 ./test/base/LendingPoolDepositTest.js

a7f55ab2c5482aae3125f1ceda44069e478e5b8e986ae96e6726dd735dd698c5 ./test/base/BaseInitializeTest.js

bff32b096da1fd9010e97fffa7e5c15074e926629abf0653777a2bc55f065cd3 ./test/base/SettingsUnpauseLendingPoolTest.js

933d9a00ea7c2d15a50211869d5c63d301caecd75e298e4b1590873c723393d3
./test/base/InterestConsensusProcessResponseTest.js

a1111a828a190058da41e4d6f5091ebdbea25c995a59baa3a19ffa3b68aff636
./test/base/EstimateGasLoanTermsConsensusProcessRequestTest.js

3f950bf6c7a2185183dbeb21559449ad9d34e1001726657a82cdf1ffd3df60ee
./test/base/TokenCollateralLoansRequireExpectedBalanceTest.js

b60449d73d1d38ec967063f137a281b8a66dac897ffd10d3acb34f61276d4f50 ./test/base/EtherCollateralLoansRepayTest.js

108cb1018bfcf21d4ab91355c567b1df2aebb739bab0a51ec9f0d86b3602b948
./test/base/EstimateGasInterestConsensusProcessRequestTest.js

27a9551b8c8579c6bc3e1e6127f4c3fb5fb4c2b80f3e8b830bcb60905a405309
./test/base/InterestConsensusSubmitInterestTest.js

281c4ac3b0490736b25d96fa873f38a101e892233f7ef0bd10e48d7fd589b3d2 ./test/base/LendingPoolWithdrawTest.js

169820785763188ed12e94ba104a832b67d92326e343e340af4756674336aa8b ./test/base/InterestConsensusInitializeTest.js

efabc34ff3cc27fb2ec524730bd1d1bb2d7c20a21eb355662858542757407076 ./test/base/LoansBaseLendingTokenTest.js

5a0e48a692929db185732b8b30ff75e6333b16af723b195cbff0582ded3f1f45 ./test/base/LendersInitializeTest.js

3dbde6e1f64eea2e85f569b627f8355742b2d11514ec80bf97614e02e59b2a89 ./test/base/LoansBaseConvertTokenToWeiTest.js

```
3003c5beeee40cb91462740bdee7c02a197c2aebe60a429629de072203ab2529 ./test/base/BaseConstructorTest.js

ed00e45c86d00b67c8126cd150bb38f632b884f0e9b5f1ea67ee644d079985ea ./test/base/LendersWithdrawInterestTest.js

3977998e148884304bb42c9668e5f4cf88bb1bc90a2a3032fef6093b051315b2 ./test/base/LoanTermsConsensusHashesTest.js

773d4217f095a4cfd7ae1d85f503d70441cc8dbc5fd1380d569d935c4d90a1f2
./test/base/InterestConsensusProcessRequestTest.js

30114fb8b23c6fd1d270e997ed2fc64eab252721e365a19e0cc3e0bc3190b861 ./test/utils/events.js

74f618d06b94ef6b01d31ab15e53e999cd0248975a98ec9025dcbff4adf415cd ./test/utils/consts.js

acdb4592ace14c9e8c9811a5fe7b722c71c535f5dc3dd3dcb85e546748801066 ./test/utils/assertions.js

ff07ffac1f69e538a9144f1dd0d09625ed2944ed46581cac747f94cd9358f4a4 ./test/utils/chains.js

be7a6d340bc54250d311b7232f6cf025a22a9e054207d3426e32b1e89eafd796 ./test/utils/loanStatus.js

5134aa0b684c36f037762ddf752572be2f5fe16e71677352bb5ebc576f798024 ./test/utils/hashes.js

baec1c37c5d2b6453c024c8a5a585db02772735eede043a91a5ecd01d0eb03c8 ./test/utils/printer.js

384a2acc67f2336a2b317b4bd97b4a3953e694856a939a5f7185d6de1a600cbc ./test/utils/structs.js

9fa1089a088ceb6ceb114f0ced244655785fd48b3467883989366a57fbbd5cd0 ./test/utils/collateral-helper.js

3d9df8947ecd5962e2b4f6f048d69c78540d8f3df65514cdbfcd8ef4372caab3 ./test/utils/contracts.js

32110f2c868bf964df1f94071eef9d56f14325e3e96a8c970e5911d757d8f896 ./test/utils/loan-terms-helper.js

8b2399ce16c214be966f3d5d4b5c94d5d8b5137f4ef16d37b8285740748c6fb3
./test/utils/encoders/AggregatorInterfaceEncoder.js

ef97b029cc6b8bd894f5c2f15d028d0d528cc78fe5ee3a3b4569ab2067bce564 ./test/utils/encoders/PairAggregatorEncoder.js

4c9d8890eee4bddfc4a0641f4f9d4d024a51457ccba3eba6b9520b3827b7209b
./test/utils/encoders/LendingPoolInterfaceEncoder.js

965540d78e0457bf4eabbf5470e0e3a3ea3626f08fd2c00b2e84573d24c1db14 ./test/utils/encoders/ERC20InterfaceEncoder.js

a9a945feef479551629b3bfe06f467ae830d061cb7b979f62d62c2b39eebf372
./test/utils/encoders/LendersInterfaceEncoder.js

99dcfbda7cd259505f00f274b311db42c60d560f30c11f0ac26f035f4e09458d
./test/utils/encoders/BurnableInterfaceEncoder.js

6804e2b47fcc294041b5006b37bb3b002f250ddec2c1950d7b8113a44689837f
./test/utils/encoders/MintableInterfaceEncoder.js

dd6182eddd69d5e04b2d1e58a1e64f3489562635c88bdd884eabb5a7f6a0054f
./test/utils/encoders/LoanTermsConsensusEncoder.js

038834cc9b3a7d0af8a99bbb79da374816865d21e99089a17b6582ed0a0a0aad
./test/utils/encoders/CompoundInterfaceEncoder.js

ca9631f90e2bc4177c457d70afc9ac0a2264c64e57eb7b7f915b7d54d67ed1c5 ./test/utils/printers/LoanInfoPrinter.js

febb39714f8b7bf25cdcbe16a7c7cf7a65a9fd5f610e34635712bb2da68a5c02
./test/providers/chainlink/ChainlinkPairAggregatorGetLatestAnswerTest.js

7c202bb2d2857deb05017b02be9a31f842fe5a99e432d78f04154fe3ec56c2f8
./test/providers/chainlink/ChainlinkPairAggregatorGetPreviousAnswerTest.js

b63e8eb6a54c648389dc775fcf20f6deca36e1806e162c8d9b116024da745c30
./test/providers/chainlink/ChainlinkPairAggregatorConstructorTest.js

ec993b814228f4cfc3d59ba37171c6be005577aed4e430bc4b433b3a560d16de
./test/providers/chainlink/InverseChainlinkPairAggregatorGetPreviousAnswerTest.js

fddca3da15f3dc03b7f283c86bfa2a3b2584c1dbd0b514529ba113ae74934d60
./test/providers/chainlink/ChainlinkPairAggregatorGetLatestRoundTest.js

5269c689780dacd38021360359f28c6648d1db5846d8d7b2729dbdc6eb7e4f89
./test/providers/chainlink/ChainlinkPairAggregatorGetLatestTimestampTest.js

179ed3443628607e4f2ea0c3866c1f55758784eaaeff4e1ae2157ae938353a1e
./test/providers/chainlink/ChainlinkPairAggregatorGetPreviousTimestampTest.js

5093ce5b682210c46128465262e110ed6ce5fecfc49dfccbe23c81c7a87cc10d
./test/providers/chainlink/InverseChainlinkPairAggregatorGetLatestAnswerTest.js

f7edc9ddee6fb72e11985423c246fae8f528b83783ba7ebfaa6009b69986af15 ./test/util/NumbersListTest.js

01ca9407e27322812741f25a584aca08a08877f22ceef1eb07cf1b94d27e4571 ./test/util/AddressLibTest.js
```

Changelog

- 2020-07-08 - Initial report
- 2020-07-27 - Diff audited (commit `dd7d2ac`)

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.