Quantstamp Security Assessment Certificate

June 16th 2022 — Quantstamp Verified

Arcade.xyz

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

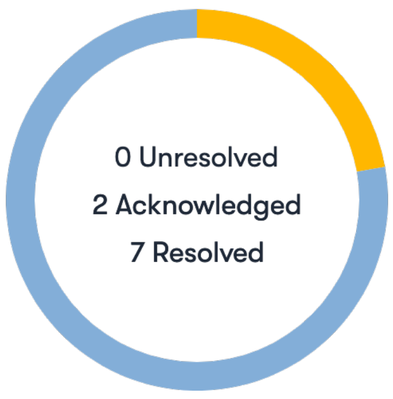
Executive Summary

Type	NFT Lending Protocol
Auditors	Rabib Islam, Research Engineer Danny Aksenov, Security Auditor Hisham Galal, Research Engineer
Timeline	2022-05-23 through 2022-06-16
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	High
Test Quality	Medium

Source Code

Repository	Commit
v2-contracts (initial audit)	f1c9c43
v2-contracts (re-audit)	bd10288

Total Issues	**9**	(7 Resolved)
High Risk Issues	**1**	(1 Resolved)
Medium Risk Issues	**2**	(2 Resolved)
Low Risk Issues	**2**	(2 Resolved)
Informational Risk Issues	**2**	(1 Resolved)
Undetermined Risk Issues	**2**	(1 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

After conducting the initial audit, we have discussed security issues ranging across all levels of severity. We have also identified a number of issues concerning documentation and best practices. We recommend that all these issues are addressed before the code is deployed to production.
Update: All security issues have been addressed.

ID	Description	Severity	Status
QSP-1	Incorrect Rollover Logic	High	Fixed
QSP-2	Loss of Precision Due to Division before Multiplication	Medium	Fixed
QSP-3	Fees can be arbitrarily high	Medium	Fixed
QSP-4	Missing Input Validation	Low	Fixed
QSP-5	Checks-Effects-Interactions Pattern Violation	Low	Fixed
QSP-6	Unlocked Pragma	Informational	Acknowledged
QSP-7	Privileged Roles and Ownership	Informational	Fixed
QSP-8	`initializeLoanWithItems` and `rolloverLoanWithItems` do not check whether the `itemPredicates` array is empty	Undetermined	Fixed
QSP-9	`verifyPredicates` returns `true` on empty arrays	Undetermined	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

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 (modified)

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 Incorrect Rollover Logic

Severity: *High Risk*

Status: Fixed

File(s) affected: `contracts/LoanCore.sol`

Description: On L311 of `LoanCore`, `data.balance` is added to `data.balancePaid` after the former has already been zeroed out. This results in an incorrect calculation.

Recommendation: Do not zero out `data.balance` before adding it to `data.balancePaid`.

QSP-2 Loss of Precision Due to Division before Multiplication

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts/InstallmentsCalc.sol`

Description: Division before multiplication may result in a loss of precision when the operations are carried over integer numbers. This occurs on L46 of `InstallmentsCalc.sol`.

Recommendation: Rewrite equations so that division happens after multiplication. In particular, in this case, the following would be appropriate as the return value:
`principal + ((principal * interestRate) / INTEREST_RATE_DENOMINATOR) / BASIS_POINTS_DENOMINATOR)`

QSP-3 Fees can be arbitrarily high

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `contracts/FeeController`

Description: Fees can be set using `setOriginationFee` and `setRollerFee`, even up to 100%.

Recommendation: Test to make sure that the fees do not go too high.

Update: Limitations on the fees have been implemented.

QSP-4 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/LoanCore.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Specifically, in the following function, it is worthwhile validating a certain condition:

- `LoanCore.initialize`
  . `_borrowerNote` should not be equal to `_lenderNote`.

Recommendation: We recommend adding the relevant checks.

QSP-5 Checks-Effects-Interactions Pattern Violation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/LoanCore.sol`

Related Issue(s): SWC-107

Description: The Checks-Effects-Interactions coding pattern is meant to mitigate any chance of other contracts manipulating the state of the blockchain in unexpected and possibly malicious ways before control is returned to the original contract. As the name implied, only after checking whether appropriate conditions are met and acting internally on those conditions should any external calls to, or interactions with, other contracts be done.
In `LoanCore`, we see on L205-206 the state being modified after a token transfer on L197. This also occurs on L382-383 after a token transfer on L371.

Recommendation: We recommend refactoring the code so that it conforms to the Checks-Effects-Interactions pattern.

QSP-6 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/*`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.11`. 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: *Project Team*: "Arcade team is electing to leave pragmas for all contracts set to `^0.8.11`. The team does not believe there is a danger of non-deterministic deployment since our `hardhat.config.ts` is only configured for the compiler version of `0.8.11`. Needing to keep this configuration unchanged is an operational risk that we are willing to manage.
"The benefits of keeping the pragma unlocked outweigh the costs, since a flexible pragma makes it easier to compose with and build on top of our contracts, both internally and by other protocols and communities. Locking the pragma enforces a downstream lock on any contracts which may inherit or import our code, which is undesirable."

QSP-7 Privileged Roles and Ownership

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/LoanCore.sol`, `contracts/vault/VaultFactory.sol`, `contracts/PromissoryNote.sol`, `contracts/ERC721PermitUpgradeable.sol`

Description: Smart contracts will often have `owner` or role variables to designate the person with special privileges to make modifications to the smart contract.
In the case of `LoanCore`, the contract has `ORIGINATOR_ROLE`, `REPAYER_ROLE`, and `FEE_CLAIMER_ROLE`.
`VaultFactory` has `DEFAULT_ADMIN_ROLE`.
`PromissoryNote` has `ADMIN_ROLE`.
`ERC721PermitUpgradeable` has `DEFAULT_ADMIN_ROLE`.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update: The README was updated.
Project Team: "We will update the user facing docs as we get closer to UI implementation and launch. Considering that the data is already in the README, and the repo is public, we consider the privileged roles and access to be sufficiently transparent at this point in either case."

QSP-8 `initializeLoanWithItems` and `rolloverLoanWithItems` do not check whether the `itemPredicates` array is empty

Severity: *Undetermined*

Status: Fixed

File(s) affected: `contracts/OriginationController.sol`

Description: Both `initializeLoanWithItems` and `rolloverLoanWithItems` of `OriginationController` do not check whether `itemPredicates` is empty. Thus, the verification of predicates is skipped.

Recommendation: Check whether `itemPredicates` is empty.

QSP-9 `verifyPredicates` returns `true` on empty arrays

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `contracts/verifiers/ItemsVerifier.sol`

Description: `verifyPredicates` returns `true` when provided with an empty array.

Recommendation: Determine whether the above functionality is intended, and if not, have the function check whether `items` is empty.

Update: *Project Team*: "An empty-array check in ItemsVerifier.sol is not appropriate, since the verifier should not be coupled to loan rules or signatures, but should just be a “dumb verifier”. If there are no predicates to check, then verification should succeed as a base case.
"The check was added to OriginationController in the resolution for QSP-8, which is coupled to loan rules. This is appropriate, since a signature with an empty predicates array could be abused to fill any loan, and this protection should be enforced as part of the lending protocol (not the verification scheme)."

Automated Analyses

Slither

Slither stops at the following issue: `Type not found enum LoanLibrary.LoanState`. This is likely due to a reported issue.

Update: Applied the fix mentioned by 0xdomrom in an issue. Slither produced 472 results, none of which present threats to security.

Code Documentation

The code is well-documented and adheres to NatSpec.

1. `LoanCore.sol::L85`, `tehe` should be spelled `the`
    - **Update**: Fixed.
1. `LoanCore.sol::L184`, `he` should be spelled `the`
    - **Update**: Fixed.
1. `LoanCore.sol::L221-222`, `validate` should be spelled `validates` and `distribute` should be spelled `distributes`
    - **Update**: Fixed.
1. `LoanCore.sol::L637`, `eiaser` should be spelled `easier`

- **Update**: Fixed.

1. `OriginationController.sol::L615`, there is a comment explaining that the numInstallments must be between 2 and 1000, however the comment fails to mention that the number of installments can also be 0.

   - **Update**: Fixed.

1. `PunkRouter.sol::L46`, `depositPUnk` should be `depositPunk`.

   - **Update**: `PunkRouter` is being removed entirely.

1. `RepaymentController.sol::L53`, implementation of grace period is not finished.

   - **Update**: Removed.

1. `RepaymentController.sol::L197`, the documentation states that `repayPart` is called when paying back an installment loan with an amount greater than the minimum amount due, however the logic on `L216` allows the repayment amount to be equal to the minimum amount due.

   - **Update**: Fixed.

1. `AssetVault.sol::L32`, `exampple` should be `example`.

   - **Update**: Fixed.

Adherence to Best Practices

1. Version `4.6` of OpenZeppelin's `Initializable.sol` contract introduces some new functionality, including a new pattern for preventing unauthorized implementation contract initialization. The new pattern is mentioned in their documentation here.

   - **Update**: *Project Team*: "Won't fix - upgrading `@openzeppelin/contracts-upgradeable` to `4.6.0` (we are currently on `4.5.2`) breaks type exporting in a way that would take too much complexity to fix."

1. At `L447` of `LoanCore`, it would be more efficient to directly pass `_msgSender()` in place of `user`.

   - **Update**: Fixed.

1. Rename `ISignatureVerifier.sol` to `IArcadeSignatureVerifier` or vice versa.

   - **Update**: Fixed.

1. * `RepaymentController.sol`, `getInstallmentMinPayment` does not check whether the `LoanState` of the `Loan` is `Repaid`.

   - **Update**: Fixed.

1. The constructor of `RepaymentController` should reference the `borrowerNote` and `lenderNote` of `LoanCore`.

   - **Update**: *Project Team*: "Fixed. The reference to `borrowerNote` was removed, since it was not used. Additionally AccessControl was removed as a dependency and replaced with Context."

1. Checking the state of a loan should be consolidated into either the Controller contracts or `LoanCore`. Currently, `OriginationController` checks the state at `rolloverLoan`.

   - **Update**: *Project Team*: "Added a check in `getInstallmentMinPayment`, but we are going to keep the redundant checks in both controller and core in place (also relevant to 4). We see benefits to having both: The checks in the controller "fail early" and save gas for users, and the checks in `LoanCore` provide redundancy in case we use a different contract in the `REPAYER_ROLE` that might have bugs."

Test Results

Test Suite Results

Three tests are currently failing.

Update: All tests are passing.

```
AssetVault
   Deployment
      ✓ should fail to initialize if deployed as a standalone (not by factory) (525ms)
   Initialize Bundle
      ✓ should successfully initialize a bundle (68ms)
      ✓ should initialize multiple bundles with unique ids (878ms)
   Deposit
     ERC20
        ✓ should accept deposit from an ERC20 token (41ms)
        ✓ should accept multiple deposits from an ERC20 token (320ms)
        ✓ should accept deposits from multiple ERC20 tokens (518ms)
     ERC721
        ✓ should accept deposit from an ERC721 token (68ms)
        ✓ should accept multiple deposits from an ERC721 token (619ms)
        ✓ should accept deposits from multiple ERC721 tokens (851ms)
     ERC1155
        ✓ should accept deposit from an ERC1155 NFT (66ms)
        ✓ should accept deposit from an ERC1155 fungible token (71ms)
        ✓ should accept multiple deposits from an ERC1155 token (572ms)
        ✓ should accept deposits from multiple ERC1155 tokens (806ms)
     ETH
        ✓ should accept deposit of ETH
        ✓ should accept multiple deposits of ETH (59ms)
   enableWithdraw
      ✓ should close the vault
      ✓ should fail to close the vault by non-owner
   call
      ✓ succeeds if current owner and on whitelist
      ✓ succeeds if delegated and on whitelist (53ms)
```

```
      ✓ fails if withdraw enabled on vault (43ms)
      ✓ fails if delegator disallows (63ms)
      ✓ fails if delegator is EOA (54ms)
      ✓ fails if delegator is contract which doesn't support interface
      ✓ fails from current owner if not whitelisted
      ✓ fails if delegated and not whitelisted (49ms)
      ✓ fails if on global blacklist
      ✓ fails if on global blacklist even after whitelisting
      ✓ fails if address is on the whitelist but selector is not
      ✓ fails if selector is on the whitelist but address is not
  Withdraw
    ERC20
       ✓ should withdraw single deposit from a bundle (51ms)
       ✓ should withdraw single deposit from a bundle after transfer (65ms)
       ✓ should withdraw multiple deposits of the same token from a bundle (79ms)
       ✓ should withdraw deposits of multiple tokens from a bundle (598ms)
       ✓ should fail to withdraw when withdraws disabled
       ✓ should fail to withdraw from non-owner (43ms)
       ✓ should throw when withdraw called by non-owner
       ✓ should fail when non-owner calls with approval
    ERC721
       ✓ should withdraw single deposit from a bundle (97ms)
       ✓ should throw when already withdrawn (105ms)
       ✓ should throw when withdraw called by non-owner (89ms)
       ✓ should fail to withdraw when withdraws disabled (80ms)
    ERC1155
       ✓ should withdraw single deposit from a bundle (77ms)
       ✓ should withdraw fungible deposit from a bundle (74ms)
       ✓ should fail to withdraw when withdraws disabled (64ms)
       ✓ should throw when withdraw called by non-owner (68ms)
    ETH
       ✓ should withdraw single deposit from a bundle
       ✓ should fail to withdraw when withdraws disabled
       ✓ should throw when withdraw called by non-owner
    Introspection
       ✓ should return true for declaring support for eip165 interface contract

CallWhitelist
  Access control
    add
       ✓ should succeed from owner (228ms)
       ✓ should fail from non-owner
       ✓ should succeed after ownership transferred (206ms)
       ✓ should fail from old address after ownership transferred (85ms)
    remove
       ✓ should succeed from owner (60ms)
       ✓ should fail from non-owner
       ✓ should succeed after ownership transferred (50ms)
       ✓ should fail from old address after ownership transferred
  Global blacklist
     ✓ erc20 transfer
     ✓ erc20 approve
     ✓ erc20 transferFrom
     ✓ erc721 transferFrom
     ✓ erc721 safeTransferFrom
     ✓ erc721 safeTransferFrom with data
     ✓ erc721 setApprovalForAll
     ✓ erc1155 setApprovalForAll
     ✓ erc1155 safeTransferFrom
     ✓ erc1155 safeBatchTransferFrom
  Whitelist
     ✓ doesn't override global blacklist
     ✓ passes after adding to whitelist
     ✓ fails after removing to whitelist (38ms)
     ✓ adding twice is a noop (38ms)
     ✓ removing twice is a noop (65ms)
     ✓ add again after removing (48ms)

FeeController
  constructor
     ✓ creates Fee Controller
    setOriginationFee
       ✓ reverts if sender does not have admin role
       ✓ reverts if new fee is over the maximum
       ✓ sets origination fee
    getOriginationFee
       ✓ initially returns 0.5%
       ✓ returns updated origination fee after set
    setRolloverFee
       ✓ reverts if sender does not have admin role
       ✓ reverts if new fee is over the maximum
       ✓ sets origination fee
    getRolloverFee
       ✓ initially returns 0.1%
       ✓ returns updated rollover fee after set

Installments
  getInstallmentMinPayment
     ✓ reverts if the loan ID is invalid (936ms)
     ✓ reverts for an already closed loan (1180ms)
  Installment Period
     ✓ Create a loan with 1 installment period, should revert. (66ms)
     ✓ Create a loan with 1001 installment periods, should revert. (55ms)
     ✓ Create a loan with interest rate greater than 1e18 and less than 1e26. (58ms)
     ✓ Create a loan with invalid signature deadline. (60ms)
     ✓ Verify missed payments equals zero. (118ms)
     ✓ reverts if trying to repay part on a closed loan (775ms)
     ✓ Fast Forward to 2nd period. Verify missed payments equals one. (119ms)
     ✓ Fast Forward to 5th period. Verify missed payments equals four. (118ms)
     ✓ Fast Forward to 6th period. Verify missed payments equals five. (119ms)
     ✓ Pay first installment then miss 2 payments. Verify missed payments equals two. (522ms)
     ✓ Miss first installment, pay second, then miss 4 payments. Verify missed payments equals four. (795ms)
  Installment Repayments
     ✓ Scenario: numInstallments: 0. Tries to use legacy loan with installment repay functions. (128ms)
     ✓ Scenario: numInstallments: 0. Tries to use installment loan with legacy repay functions. (130ms)
     ✓ Scenario: numInstallments: 8, durationSecs: 36000, principal: 100, interest: 10%. Repay minimum on first payment. (548ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Repay minimum on first payment. (526ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Pay the minimum with insufficient allowance, should revert. (135ms)
    Late Payments
       ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Make repayment after one skipped payment.  (525ms)
       ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Make repayment after two skipped payments. (546ms)
       ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Pay the minimum with insufficient allowance, should revert. (143ms)
       ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Make repayment after three skipped payments. (564ms)
      After Loan Duration Payments
         ✓ Repay minimum after missing loan duration and 1 extra period. (593ms)
         ✓ Repay minimum after missing loan duration and 1 extra period, with insufficient allowance, should revert. (155ms)
         ✓ Repay minimum after skipping whole loan duration x 2. (623ms)
         ✓ Repay minimum after skipping whole loan duration x 10. (1169ms)
    Multiple Repayments
       ✓ Scenario: numInstallments: 8, durationSecs: 36000, principal: 100, interest: 10%. Repay minimum on first two repayments. (1621ms)
       ✓ Scenario: numInstallments: 4, durationSecs: 36000, principal: 100, interest: 10%. Repay the minimum interest plus 1/4 the principal for four consecutive payments. (2035ms)
       ✓ Scenario: numInstallments: 8, durationSecs: 72000, principal: 100, interest: 10%. Repay the minimum plus 1/4 the principal for four payments every other period. (2216ms)
       ✓ Scenario: numInstallments: 12, durationSecs: 1y, principal: 1000, interest: 6.25%. Repay minimum on 12 payments, verify the principal has not changed. (5615ms)
       ✓ Scenario: numInstallments: 12, durationSecs: 1y, principal: 100000, interest: 10.00%. Repay min interest and monthly principal x 12. (5912ms)
       ✓ Scenario: numInstallments: 4, durationSecs: 1y, principal: 100, interest: 10.00%. Repay min interest and monthly principal x 4. (2074ms)
       ✓ Repay everything with repayPart after paying 2 minimum payments (in second installment period). (1917ms)
       ✓ Scenario: numInstallments: 4, durationSecs: 1y, principal: 100, interest: 10.00%. Repay min interest x 1 and 1/4 principal, then pay off rest of loan. (1175ms)
       ✓ Scenario: numInstallments: 24, durationSecs: 2y, principal: 1000, interest: 0.75%. Repay minimum on 24 payments, verify the principal has changed. (11281ms)
       ✓ Send zero as amount for repayPart call. (143ms)
    Close Loan
       ✓ Close loan in first installment period. (1162ms)
       ✓ Close loan in first installment period, but set allowance to less than required. Should revert. (140ms)
       ✓ Close loan in last installment period. (808ms)
       ✓ Close loan after paying 2 min payments (3rd installment period). (1575ms)
       ✓ Close loan after paying 2 min payments (2nd installment period). (1550ms)
       ✓ Close loan after paying 1 minimum payment, 1 repayPart for half the principal (2nd installment period). (1553ms)
  Defaults
     ✓ Scenario: numInstallments: 2, durationSecs: 36000. Claim after first missed installment. (805ms)
     ✓ Scenario: numInstallments: 2, durationSecs: 36000. Borrower calls claim after first missed installment, should revert. (106ms)
     ✓ Scenario: numInstallments: 2, durationSecs: 36000. Claim in first installment period should revert. (143ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000. Claim after first missed installment, should revert. (124ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000. Claim after 40% the loan duration, should revert still second installment period. (124ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000. Borrower repays minimum. Lender tries to claim in same installment, should revert (518ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000. Borrower repays minimum. Lender tries to claim in second installment, should revert (530ms)
     ✓ Scenario: numInstallments: 4, durationSecs: 36000. Borrower repays minimum. Lender tries to claim various times in loan duration. (983ms)
     ✓ Scenario: numInstallments: 10, durationSecs: 36000. Borrower repays minimum. Lender tries to claim various times in loan duration. (1017ms)
     ✓ Scenario: numInstallments: 24, durationSecs: 2y. Borrower repays minimum. Lender tries to claim various times in loan duration. (1081ms)

Integration
  Originate Loan
     ✓ should successfully create a loan (549ms)
     ✓ should fail to start loan if wNFT has withdraws enabled (132ms)
     ✓ should fail to create a loan with nonexistent collateral (58ms)
     ✓ should fail to create a loan with passed due date (60ms)
  Repay Loan
     ✓ should successfully repay loan (1142ms)
     ✓ should allow the collateral to be reused after repay (787ms)
     ✓ fails if payable currency is not approved (117ms)
     ✓ fails with invalid note ID (122ms)
  Claim loan
     ✓ should successfully claim loan (489ms)
     ✓ should allow the collateral to be reused after claim (569ms)
     ✓ fails if not past durationSecs (108ms)
     ✓ fails for invalid noteId
     ✓ fails if not called by lender (100ms)
```

```
ItemsVerifier
  verifyPredicates
    ✓ fails for an invalid collateral type (100ms)
    ✓ fails if a signature item is missing an address (86ms)
    ✓ fails if an ERC1155 item has a non-positive required amount (84ms)
    ✓ fails if an ERC1155 item has a an invalid token ID (78ms)
    ✓ fails if an ERC20 item has a non-positive required amount (45ms)
    ✓ verifies a specific ERC721 and token id (160ms)
    ✓ verifies a specific ERC721 for any token id (176ms)
    ✓ verifies a specific ERC1155 and token id with a minimum amount (184ms)
    ✓ verifies a minimum ERC20 amount (79ms)
    ✓ verifies a combination of multiple items (433ms)

LoanCore
  Deployment
    ✓ should not allow initialization with an invalid fee controller (368ms)
    ✓ should not allow initialization with an invalid borrower note
    ✓ should not allow initialization with an invalid lender note
  Start Loan
    ✓ should successfully start a loan (104ms)
    ✓ should successfully set fee controller and use new fee (133ms)
    ✓ should successfully start two loans back to back (179ms)
    ✓ should fail to start two loans where principal for both is paid at once (152ms)
    ✓ rejects calls from non-originator (38ms)
    ✓ should fail to start a loan that is already started (95ms)
    ✓ should fail to start a loan that is repaid (153ms)
    ✓ should fail to start a loan that is already claimed (141ms)
    ✓ should fail to start a loan if collateral has not been sent
    ✓ should fail to start a loan if lender did not deposit (53ms)
    ✓ should fail to start a loan if lender did not deposit enough (71ms)
    ✓ should fail when paused (42ms)
  Repay Loan
    ✓ should successfully repay loan (137ms)
    ✓ rejects calls from non-repayer (119ms)
    ✓ should update repayer address and work with new one (154ms)
    ✓ should fail if the loan does not exist (97ms)
    ✓ should fail if the loan is not active (108ms)
    ✓ should fail if the loan is already repaid (138ms)
    ✓ should fail if the loan is already claimed (129ms)
    ✓ should fail if the debt was not repaid (106ms)
    ✓ should fail if the debt was not repaid in full (121ms)
    ✓ should fail if the interest was not paid in full (227ms)
    ✓ should still work when paused (485ms)
  Claim loan (no installments)
    ✓ should successfully claim loan (118ms)
    ✓ Rejects calls from non-repayer (118ms)
    ✓ should fail if loan doesnt exist (98ms)
    ✓ should fail if the loan is not active (105ms)
    ✓ should fail if the loan is already repaid (136ms)
    ✓ should fail if the loan is already claimed (129ms)
    ✓ should fail if the loan is not expired (100ms)
    ✓ should fail when paused (106ms)
    ✓ pause, unPause, make tx (135ms)
  Claim fees
    ✓ should successfully claim fees (101ms)
    ✓ should fail for anyone other than the admin (110ms)
    ✓ only fee claimer should be able to change fee claimer (125ms)
  canCallOn
    ✓ should return true for borrower on vault in use as collateral (121ms)
    ✓ should return true for any vaults if borrower has several (167ms)
    ✓ should return false for irrelevant user and vault
    ✓ should return false for irrelevant user on vault in use as collateral (121ms)
    ✓ should return false for lender user on vault in use as collateral (134ms)
  setFeeController
    ✓ should revert if the caller is not the owner
    ✓ should revert if provided an invalid fee controller address
  Upgradeable
    ✓ maintains state: confirms that v1 loans still exist after upgrading to v2 (190ms)
  Nonce management
    ✓ does not let a nonce be consumed by a non-originator
    ✓ consumes a nonce
    ✓ reverts if attempting to use a nonce that has already been consumed
    ✓ cancels a nonce
    ✓ reverts if attempting to use a nonce that has already been cancelled

OriginationController
  initializer
    ✓ Reverts if _loanCore address is not provided
    ✓ Instantiates the OriginationController (89ms)
  Upgradeability
    ✓ v1 functionality can be upgraded in v2 (61ms)
    ✓ v1 functionality cannot be upgraded to v2 by non authorized user (56ms)
  initializeLoan
    ✓ Reverts if msg.sender is not either lender or borrower (63ms)
    ✓ Reverts if wNFT not approved (53ms)
    ✓ Reverts if principal not approved (45ms)
    ✓ Reverts if principal below minimum allowable amount (38ms)
    ✓ Reverts if approving own loan (54ms)
    ✓ Reverts if signer is not a participant (47ms)
    ✓ Reverts for an invalid nonce (51ms)
    ✓ Reverts on an expired signature (44ms)
    ✓ Reverts if the nonce does not match the signature (49ms)
    ✓ Initializes a loan signed by the borrower (89ms)
    ✓ Initializes a loan signed by the lender (87ms)
    ✓ Initializes a loan with unbundled collateral (90ms)
    ✓ does not allow a nonce to be re-used (104ms)
  initializeLoanWithCollateralPermit
    ✓ Reverts if the collateral does not support permit
    ✓ Reverts if vaultFactory.permit is invalid
    ✓ Initializes a loan with permit (100ms)
initializeLoanWithItems
    ✓ Reverts if the collateralAddress does not fit the vault factory interface
    ✓ Reverts if the required predicates fail (78ms)
    ✓ Reverts if the required predicates array is empty (41ms)
    ✓ Reverts for an invalid nonce (225ms)
    ✓ Reverts if the nonce does not match the signature (121ms)
    ✓ Reverts if the verifier contract is not approved (120ms)
    ✓ Initalizes a loan signed by the borrower (153ms)
    ✓ Initalizes a loan signed by the lender (152ms)
    ✓ does not allow a nonce to be re-used (248ms)
    ✓ Initializes a loan with permit and items (161ms)
  initializeLoanWithCollateralPermitAndItems
    ✓ Reverts if vaultFactory.permit is invalid (99ms)
    ✓ Trigger ERC721Permit not token owner (89ms)
verification whitelist
    ✓ does not allow a non-owner to update the whitelist
    ✓ Try to set 0x0000 as address, should revert.
    ✓ allows the contract owner to update the whitelist
    ✓ does not allow a non-contract owner to perform a batch update
    ✓ reverts if a batch update's arguments have mismatched length
    ✓ allows the contract owner to perform a batch update
approvals
    ✓ reverts if trying to approve oneself
    ✓ allows the borrower to approve another signer (99ms)
    ✓ allows the lender to approve another signer (155ms)
    ✓ allows the borrower to approve another originator (109ms)
    ✓ allows the lender to approve another originator (106ms)
    ✓ honors an ERC-1271 approval (176ms)
    ✓ does not allow unilateral borrower origination even if the lender approves (59ms)
    ✓ does not allow unilateral lender origination even if the borrower approves (55ms)

PromissoryNote
  constructor
    ✓ Creates a PromissoryNote
    ✓ fails to initialize if not called by the deployer (394ms)
    ✓ fails to initialize if already initialized (83ms)
  mint
    ✓ Reverts if sender is not an assigned minter
    ✓ Assigns a PromissoryNote NFT to the recipient
  burn
    ✓ Reverts if sender does not own the note
    ✓ Burns a PromissoryNote NFT (78ms)
  pause
    ✓ does not allow a non-admin to pause (73ms)
    ✓ does not allow a non-admin to unpause (68ms)
    ✓ allows an admin to pause (68ms)
    ✓ allows an admin to unpause (87ms)
    ✓ transfers revert on pause (167ms)
  Permit
    ✓ should accept owner signature
    ✓ rejects if given owner is not real owner
    ✓ rejects if promissoryNoteId is not valid (44ms)
    ✓ rejects reused signature
    ✓ rejects other signature
    ✓ rejects expired signature
  Introspection
    ✓ should return true for declaring support for eip165 interface contract

PunkRouter
  Deposit CryptoPunk
    ✓ should successfully deposit a cryptopunk into bundle (1087ms)
    ✓ should fail if not approved (812ms)
    ✓ should fail if not owner (479ms)
  Withdraw CryptoPunk held by PunkRouter
    ✓ should successfully withdraw punk (496ms)
    ✓ should fail if not designated admin (461ms)
```

```
RepaymentController
   ✓ Legacy loan type (no installments), repay interest and principal. 100 ETH principal, 10% interest rate. (982ms)
   ✓ Legacy loan type (no installments), repay interest and principal. 10 ETH principal, 7.5% interest rate. (598ms)
   ✓ Legacy loan type (no installments), repay interest and principal. 25 ETH principal, 2.5% interest rate. (783ms)
   ✓ Legacy loan type (no installments), repay interest and principal. 25 ETH principal, 2.5% interest rate. Borrower tries to repay with insufficient balance. Should revert. (115ms)
   ✓ Legacy loan type (no installments), repay interest and principal. 25 ETH principal, 2.5% interest rate. Borrower tries to repay with insufficient allowance. Should revert. (107ms)
   ✓ Legacy loan type (no installments), repay interest and principal. 9999 Wei principal, 2.5% interest rate. Should revert on initialization. (43ms)
   ✓ Legacy loan type (no installments), repay interest and principal. 1000 Wei principal, 2.5% interest rate. (612ms)

RepaymentController revert branches
   ✓ Get full interest with invaild rate, should revert.

Rollovers
  Rollover Loan
     ✓ should not allow a rollover if the collateral doesn't match
     ✓ should not allow a rollover if the loan currencies don't match
     ✓ should not allow a rollover on an already closed loan (501ms)
     ✓ should not allow a rollover if called by a third party
     ✓ should not allow a rollover if signed by the old lender
     ✓ should not allow a rollover if called by the old lender
     ✓ should roll over to the same lender (97ms)
     ✓ should fail to roll over an already closed loan (111ms)
     ✓ should roll over to a different lender (128ms)
     ✓ should roll over to a different lender, called by the lender (123ms)
     ✓ should roll over to a different lender using an items signature (185ms)
     ✓ should roll over to the same lender using an items signature (167ms)
     ✓ should roll over a loan with for the borrower and the same lender where no funds need to move (102ms)
     ✓ should roll over a loan with extra principal for the borrower and the same lender (107ms)
     ✓ should roll over a loan with extra principal for the borrower and a different lender (104ms)
     ✓ should roll over an installment loan to the same lender (198ms)
     ✓ should roll over an installment loan to a different lender (219ms)

VaultFactory
   ✓ should fail to initialize if passed an invalid template (116ms)
   ✓ should return template address
  isInstance
     ✓ Should return false for non-instance address
     ✓ Should return true for instance address
  instanceCount
     ✓ Should return 0 at first
     ✓ Should increment with bundles (39ms)
  instanceAt
     ✓ Should revert if no vault at index
     ✓ Should return vaults at index (96ms)
  Permit
     ✓ should accept owner signature (49ms)
     ✓ rejects if given owner is not real owner
     ✓ rejects if bundleId is not valid
     ✓ rejects reused signature (47ms)
     ✓ rejects other signature (58ms)
     ✓ rejects expired signature (122ms)
  ERC721
    with minted tokens
      balanceOf
        when the given address owns some tokens
           ✓ returns the amount of tokens owned by the given address (259ms)
        when the given address does not own any tokens
           ✓ returns 0
        when querying the zero address
           ✓ throws
      ownerOf
        when the given token ID was tracked by this token
           ✓ returns the owner of the given token ID
        when the given token ID was not tracked by this token
           ✓ reverts
      transfers
        transferFrom
           ✓ succeeds when called by owner (58ms)
           ✓ succeeds when called by approved user (54ms)
           ✓ succeeds when called by an operator (57ms)
           ✓ fails when the owner address is incorrect
           ✓ fails when the sender is not authorized
           ✓ fails when the token id does not exist
           ✓ fails when the recipient is the zero address
          properly performs a self-send
             ✓ keeps ownership of the token
             ✓ clears the approval for the token ID
             ✓ keeps the owner balance
  Upgradeable
     ✓ Upgrades to v2 (52ms)


335 passing (2m)
```

Code Coverage

Due to some tests failing, it is not possible to output coverage data.

Update: Coverage is good, though missing some branches.

File	Statements	Branches	Functions	Lines
contracts/	100%	96.3%	98.82%	99.52%
contracts/errors/	100%	100%	100%	100%
contracts/interfaces/	100%	100%	100%	100%
contracts/libraries/	100%	100%	100%	100%
contracts/vault/	**100%**	**100%**	**100%**	**100%**
contracts/verifiers/	100%	95.45%	100%	95%

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

```
b177e136f73c82b24c33ec00aa846291aa0bd9855aaad3f8258d702ed410dc1c ./contracts/OriginationController.sol

a23e8c646e27cd07861996c74483b06ef96f6fe99d6fec137a71f7baf4379207 ./contracts/LoanCore.sol

da639c60e30297868ab9d9279f19cbab7d017b90731cc61ab9ec6841dd4c526f ./contracts/FeeController.sol

189064c5f5b171f5f7f70e84820b090b5850234070732faf561701508d71ab31 ./contracts/RepaymentController.sol

0654e248d42cbf2068b47c22a111c861ce09109855398608 9d8160e1efdf4d74 ./contracts/ERC721Permit.sol

387f90c01da89f1c2870d727c50658f355d16108f00b52b2f1f8703ddb4f5920 ./contracts/ERC721PermitUpgradeable.sol
```

7e739285bfd49710dc2296e656680ba6360ef38839c99913c1b267e1deae9d5c ./contracts/PromissoryNote.sol

2e5e605f1403263ca361846f3f4f87a610771e87eeb17a68cf763dc577b262cd ./contracts/InstallmentsCalc.sol

a9305aad0b4866a7e97be7944038f4db600769bfb9685045f78d615a34a4738b ./contracts/PunkRouter.sol

1a8b6ba70c8f38f8cbb93f0b0fddea36284a269793142a6feb05cd288b40a934 ./contracts/interfaces/ICallWhitelist.sol

a15115d00cf8c57e4958c1fd92ba6b91415272a2965ee805a759403e8138a0b7 ./contracts/interfaces/IFeeController.sol

d64f8ab0d4d9b43d697e244016b954c8acc3e17c9b129e2d8f211b063bf3f048 ./contracts/interfaces/IERC721Permit.sol

6affc9d9bdb7b58401dd45b633e3f31fe4b3baba3857c0a24daf279bbb560f20 ./contracts/interfaces/ICallDelegator.sol

2365bf2d286666df0b526d3b3b746acfbeb4e9b697065fcfa47088a97c12fd4a ./contracts/interfaces/ISignatureVerifier.sol

fd154294e8632f5e65e28b057e971c3b0cc4d319966666d921b373bcc7db517a ./contracts/interfaces/IInstallmentsCalc.sol

e4f5339725be739268c2b6fd3c16e58fdf525f3c4d34f0e98016d0b88b6cbaa9 ./contracts/interfaces/IFlashRollover.sol

84e3933204f5f522deca3bd6a4b9618ca419b5c0e32a1e679d02a2487cbe000a ./contracts/interfaces/IVaultFactory.sol

89c12ee40df1a13a7e33bfbdb711e1522d94b45a739ec1fa9597febd8172d9bc ./contracts/interfaces/IPromissoryNote.sol

38b5cb01dd55280166796057093d9e282a14fbf86bd241ddd1883d8f28bae5ea ./contracts/interfaces/ILoanCore.sol

aa8a0fbc79549644d0cfc51cf3d54c44bfeae8da2d3065de934f5707bb05081a ./contracts/interfaces/IERC721PermitUpgradeable.sol

a4189a4d6586b886c42bd982d2f191406bb5e566eea87aa6c16ff3e2ff80df44 ./contracts/interfaces/IAssetVault.sol

c45d03153e3ab780b5734b2757584a485245cb93b02340ea186a6a95349ed416 ./contracts/interfaces/IOriginationController.sol

a8befdb589388e4ebc5b29f9500e9f6ee714480abedfdfa8a70e0b15fcd4905b ./contracts/interfaces/IRepaymentController.sol

9919ba4881e8d864a2ab5d4fb88310b39d7f9da55833f6a5a59dd864a25df280 ./contracts/vault/VaultFactory.sol

6453b5565cc74c135048a7ac46325242cc847719c8b991d283f0ceeca8ef6db9 ./contracts/vault/CallWhitelist.sol

53def0bbfe573d8d7cccc0e52df0593bceb2b1160b3e77a6fa236c3b8d1ce54b ./contracts/vault/AssetVault.sol

3f8529466b73248e12f09c0052bed60e71f7b7a7155193ec6ad158935d40c158 ./contracts/vault/OwnableERC721.sol

065753a3fbb6dbf867a002e98a8f5cff28522b193fe870813e9fe3473aef59b1 ./contracts/errors/LendingUtils.sol

72190cbcb94dcf1158e892d1cb22af46403404c81b1cd81d072586cec30e5d4c ./contracts/errors/Lending.sol

8033d0cea95809d1c7f55071cc2969709ac2031219c8e35febd9513dbceb698f ./contracts/errors/Vault.sol

2054e1816f7e93ef4dc3268f66f55e9da0aaf36d14797013f97848a4022be4a8 ./contracts/external/interfaces/ILendingPool.sol

9260c5e733c2afe95172639b2c22c9fef473ae045123d110ea19fa755aa96165 ./contracts/external/interfaces/IWrappedPunks.sol

1d2c0077e71e912ced3cf1086b3ac66c9d75d8133bb6681249ee6c6db8961393 ./contracts/external/interfaces/IPunks.sol

64ce829a964d65335618240ab1df24e6d507aa94a0b746a16c2a0a5142233b20 ./contracts/libraries/LoanLibrary.sol

bd1eba5f65371498e60f4383b59c098ceaed7a83021fd4c1c32ed13265127a1e ./contracts/verifiers/ItemsVerifier.sol

a3c5f31a933ca380540e8f2a40da305509b43c6a842af20af2c684cb9abc0048 ./contracts/test/MockERC1155.sol

4c893ebac56c3ce5753fb6671686f9ad0745f2cc10386803b555f1ad4409f6d1 ./contracts/test/MockLendingPool.sol

b3dbf60def7f16af0beba03ef584784418b06560d68c0f17acc7b6506b7ac57c ./contracts/test/Templates.sol

52f0968e9c1211bad9175ab8419990906cff027bc20dfdf9562760c673fa80c7 ./contracts/test/MockERC1271Lender.sol

cf580bc68c04363f148741a427077459495ce0bb8374e6a581d268062cf61d26 ./contracts/test/MockOpenVault.sol

3cb16f55ad31b7ca873843f9544e47e721aa633c1ed936f404a7d1a3e474b818 ./contracts/test/MockERC20.sol

eeac66a28fb8e1f338f201686d9e84eeba4a7bf8a209cc6397c3742049ca9980 ./contracts/test/VaultFactoryV2.sol

18110d427ccd34e10dd1f74caaac45abc4b7b3c0b9b9884fbf615ba2e27fe4c5 ./contracts/test/LoanCoreV2Mock.sol

aded9d250ab375de97e39554bb525fb832bf17874c5c031b555ca4fb04f49540 ./contracts/test/UserProxy.sol

34b6752faedb7dbf3e05d533ea837d3ec807413927be34740c121e2206e673d8 ./contracts/test/MockOriginationController.sol

01e94d305909fe171f69b4f40566bcc823b59859398ad2162e2582ef5ea34353 ./contracts/test/MockCallDelegator.sol

3417c2a00568e639edd46629f17caa9d7197809cf41cc8ed62270a503c090c39 ./contracts/test/CryptoPunks.sol

420ef541e0761e31f5a1ec9f879158397181ed223e39627c3344826f80cfa5da ./contracts/test/WrappedPunks.sol

8f00f1202d18db5856b88cd4e8931e66d8efc8dfcb423f77044d9121726415d4 ./contracts/test/MockERC721.sol

70939ee2d6d357770af7d14f5a0aabec4e11de28087da4fce664bd3d1c8344b6 ./contracts/test/ERC721ReceiverMock.sol

Tests

3814c11a3ec4a9a1567a2e20406c1f8cf56b68d5f682b55fa10ed2c9b690e826 ./test/Rollovers.ts

f4424e7d313bd3b157a6ad13fc954ee54b76aa9f833c0fb6492febff58b6c6f9 ./test/OriginationController.ts

ff62d0ac0896ee7f876f8f370889e9dd3e4d33ae27e240b4b0cedb4c722c605a ./test/PunkRouter.ts

bc640382f1acf3571a10f1f5dee621cc18bbee78440b030d10e4d4a9b3a25e35 ./test/Integration.ts

3b62668f3df862afad1c476639e5438672758cd7f1166f604b6977851273bf13 ./test/CallWhitelist.ts

cb34e45db39c1670f8947214b1bf004e9c87ea093279de0670160dda4af01493 ./test/ItemsVerifier.ts

88ad6d2c6ee9d20b3defb6644ffcb67b336d43ce477fb4f5b31f7235afaed4ab ./test/LoanCore.ts

f9ec81bda27cd2916cbc18b600bbd2a8341589dec3caee14aaa6868ef5b98897 ./test/VaultFactory.ts

d3c0b698190304e7f934bd99fe1d5b4baa082c4a713453273a05396580afbf11 ./test/AssetVault.ts

c9777a68242234907c2e91c88be7991c5698e76b12e4f91a98abd13dd8a3c62b ./test/Installments.ts

cf4c15c1416c2a44c01bb7edb7232861fdf28b6f44e4d3364efa885943945a03 ./test/PromissoryNote.ts

a33ec755cbc097f0e5c1657ff90de2b77ebe8d7966cc7aa2aaee5915de13683a ./test/FeeController.ts

092e8587a07b164bf31cb231dc8759f363323da7f25e931782dbca38e068a8e4 ./test/RepaymentController.ts

76665fed96653693acf515a8c2fc4976b932ddef82ec2c805c5849a5d39d6c97 ./test/utils/erc20.ts

b7a6fcc5dc7b6d5301ca6e2281ab6c64e831059b593e668e2e8417e15c557763 ./test/utils/eip712.ts

```
2773f1711436cc38ccb90138037f55ee4106365aac47ee615a43d60973926c7a ./test/utils/contracts.ts
4d59a960d85d1a6e4eb8ad239007ea43a25fe15cf3eb1abc334a6e91e14db373 ./test/utils/loans.ts
7923037d4c54e73aa0cfe89feaab5b357f5f28696210e5d784ba16b33bb06825 ./test/utils/types.ts
b36246806c07952a00acb2b78f91e7db5257bbe5e24764392b92ee6d617cdd7c ./test/utils/erc721.ts
673a0095f8619e476bd32f7c5958c34f15eb1afcfb03320c08cd768fce042c74 ./test/utils/erc1155.ts
8cc07be5c055af50c0a598267baf50de8e54f40d580b84e94ebd942a26e16cc3 ./test/utils/time.ts
```

Changelog

- 2022-06-06 - Initial report
- 2022-06-16 - Final report

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.