Quantstamp Security Assessment Certificate

September 20th 2021 — Quantstamp Verified

Decentraland 2

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

Executive Summary

Type	MarketPlace
Reviewers	Souhail Mssassi, Research Engineer Leonardo Passos, Senior Research Engineer
Timeline	2021-08-25 through 2021-09-16
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Collections Architecture Collection V2 Contract
Documentation Quality	Low
Test Quality	Undetermined

Source Code

Repository	Commit
bid-contract	9964f0f
wearables-contracts	c91d81e

Total Issues	**14** (1 Resolved)
High Risk Issues	**1** (0 Resolved)
Medium Risk Issues	**2** (0 Resolved)
Low Risk Issues	**7** (1 Resolved)
Informational Risk Issues	**3** (0 Resolved)
Undetermined Risk Issues	**1** (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

Quantstamp has reviewed the Decentraland and found 14 issues of various severity levels. Quantstamp strongly suggests addressing all issues in this report to the extent that is possible. Many issues are a result of weak documentation and unclear behavior.

Disclaimer

This audit restricts to the following files:

- contracts/bid/ERC721Bid.sol
- contracts/bid/ERC721BidStorage.sol
- contracts/bridge/*
- contracts/validators/*
- contracts/collections/v2/ERC721BridgedCollection.sol

Disclaimer

After the re-audit, except for one issue, none of the issues presented in this report were fixed. The team dubbed the issues as unlikely to cause any harm; by doing that, they take full responsibility in case any of the issues herein reported do happen in practice. Moreover, our team was unable to run the tests for the wearables part. Results of that test suite are hence excluded in this report.

ID	Description	Severity	Status
QSP-1	Potential Market Manipulation	High	Acknowledged
QSP-2	Race Condition in the `ownerCutPerMillion`	Medium	Acknowledged
QSP-3	Integer Overflow / Underflow	Medium	Acknowledged
QSP-4	Ownership Can Be Renounced	Low	Acknowledged
QSP-5	Input Addresses Missing Validation	Low	Acknowledged
QSP-6	Highest Bid Does Not Necessarily Win	Low	Acknowledged
QSP-7	NFT Owners May Lose Their Token When Attempting To Accept A Bid	Low	Acknowledged
QSP-8	Inconsistent Minimum Bid Duration	Low	Fixed
QSP-9	Inconsistent Maximum Bid Duration	Low	Acknowledged
QSP-10	`onERC721Received` does not Ensure ERC721 Compatibility	Low	Acknowledged
QSP-11	Unlocked Pragma	Informational	Acknowledged
QSP-12	Gas Usage / `for` Loop Concerns	Informational	Acknowledged
QSP-13	Block Timestamp Manipulation	Informational	Acknowledged
QSP-14	Bids Can Be Made To Any ERC721 Token	Undetermined	Acknowledged

Quantstamp Review 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 reviewing 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 security review.

Setup

Tool Setup:

- Slither v0.6.6
- Mythril v0.2.7

Steps taken to run the tools:

Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .` Installed the Mythril tool from Pypi: `pip3 install mythril` Ran the Mythril tool on each contract: `myth -x path/to/contract`

Findings

QSP-1 Potential Market Manipulation

Severity: *High Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: The token owner could induce the market towards higher bids by either: surpassing the bid value of the highest bid available; or adding a single bid in the case of no bids at a given time. All the owner has to do is to use a different address from the one of the token owner.

Recommendation: The best approach to overcome this issue is to rely on a commit-reveal scheme; however, due to different pressing requirements, this may not be feasible for a given project. In the latter case, at the very least inform users that market manipulation is possible and that they should be aware of this risk when placing bids.

QSP-2 Race Condition in the `ownerCutPerMillion`

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `contracts/ERC721Bid.sol`

Description: The owner can modify the value of `ownerCutPerMillion` by invoking `setOwnerCutPerMillion(...)`. However, `ownerCutPerMillion` is used in the `onERC721Received(...)` function; if `setOwnerCutPerMillion(...)` is mined first, the user will receive a reduced number of tokens, without any chance of reverting the transaction

Recommendation: In `onERC721Received(...)`, use the data parameter to encode both the `bidId` and the expected `ownerCutPerMillion`. If the latter differs from the current `ownerCutPerMillion` as stored in the contract, revert entirely.

QSP-3 Integer Overflow / Underflow

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `contracts/ERC721Bid.sol`

Description: In the `ERC721Bid` contract, the `_placeBid` function performs an increment using the usual addition operator, which might cause integer overflow and thus harm the business logic of the contract.

Recommendation: Use the `SafeMath` library to perform the mathematical operation in order to avoid integer overflows and underflows.

QSP-4 Ownership Can Be Renounced

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`, `contracts/bridges/CollectionsBridgeBase.sol`, `contracts/bridges/CollectionsBridgeChild.sol`, `contracts/bridges/CollectionsBridgeRoot.sol`, `contracts/validators/ERC721CollectionV2Validator.sol`

Description: All ownable contracts can be left with no owner, as he may renounce ownership. If that happens, any function guarded by the `onlyOwner` modifier will no longer be able to be executed. This is particularly concerning; for instance, if the owner renounces his ownership, one is unable to pause `ERC721Bid` as a temporary reactive measure against hacks.

Recommendation: Override the `renounceOwnership` s.t. it always reverts.

QSP-5 Input Addresses Missing Validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`, `contracts/validators/ERC721CollectionV1Validator.sol`, `contracts/bridges/CollectionsBridgeBase.sol`,

Description: Input addresses passed on to the constructors of `ERC721Bid`, `ERC721CollectionV1Validator`, and `ERC721BridgedCollection` lack proper validation. Settting an address to `0x0` or to an EOA when a contract address is expected can lead to an incorrect state, potentially causing system downtime and malfunctioning. Additionally, the `admin` address in the function `setAmin` is not verified.

Recommendation: Add input verification logic s.t.:

- When an address is expected to refer to a contract, it passes the `Adress.isContract(...)` check;
- Input addresses are always different from `0x0`.

QSP-6 Highest Bid Does Not Necessarily Win

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: The token owner may chose a bid that is not necessarily the highest. This could occur, for instance, when a lower bid is mined faster than a higher valued one; while the latter stays in the mempool, the token owner accepts the lower bid.

Recommendation: There is no recommendation to solve this issue as it stems from the intrinsic characteristics of the Ethereum blockchain. Nonetheless, we recommend publicly documenting the issue as a means to educate users of its existence.

QSP-7 NFT Owners May Lose Their Token When Attempting To Accept A Bid

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: To accept a bid, token owners must transfer their NFT to the bid contract by means of a `safeTransferFrom` call. However, it is not inconceivable that users do so by incorrectly calling `transferFrom`. The latter does not trigger the `onERC721Received` logic, which will cause the owner to lose his ownership of the NFT, while not receiving any payment.

Recommendation: While the documentation in the code is clear, we suggest making it publicly available outside the codebase. Additionally, if possible, consider having a mechanism in place for owners to get their NFTs back in case they mistakenly transfer them to the bid contract by means of calling `transferFrom`.

QSP-8 Inconsistent Minimum Bid Duration

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: Following the code, a bid duration should be at least one minute, but the error message on L118 in `ERC7Bid.sol` contradicts that. It is unclear whether the error message is wrong or whether bid durations should be greater than 1 minute.

Recommendation: Clarify whether bids should last a duration greater than or equal to 1 minute. If so, change the error message on L118 to “The bid should last at least a minute”. Otherwise, change the require statement on L116-118 to:

```
require(
  _duration > MIN_BID_DURATION,
  "The bid should be last longer than a minute"
);
```

QSP-9 Inconsistent Maximum Bid Duration

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: Following the code, a bid duration should not exceed 6 months. The require statement ensuring the maximum duration, however, checks if the duration is at most 182 days, which is incorrect. For instance, an exact 6 month window starting in August has 183 days; however, 183 days would be incorrectly flagged as an invalid duration.

Recommendation: Set the error message in L123 to “The bid can not last longer than 182 days”.

QSP-10 `onERC721Received` does not Ensure ERC721 Compatibility

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: The function `ERC721Bid.onERC721Received(...)` does not check if `msg.sender` is ERC721 compliant, i.e., if it passes `_requireERC721(...)`. Since `onERC721Received(...)` is public, any contract can invoke it.

Recommendation: Add the `_requireERC721(msg.sender)` check as part of `ERC721Bid.onERC721Received(...)`.

QSP-11 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/*`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.7.6`. 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, it is recommended to remove the caret to lock the file onto a specific Solidity version.

QSP-12 Gas Usage / `for` Loop Concerns

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/ERC721Bid.sol`, `contracts/CollectionsBridgeChild.sol`, `contracts/CollectionsBridgeRoot.sol`, `contracts/CollectionsBridgeRoot.sol`, `contracts/ERC721CollectionV1Validator.sol`, `contracts/ERC721CollectionV2Validator.sol`, `contracts/ERC721BridgedCollection.sol`

Description: Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, 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. It is best to break such loops into individual functions as possible. Lines Affected :

- ERC721Bid.removeExpiredBids (L292);
- CollectionsBridgeChild.withdrawFor (L43)
- CollectionsBridgeRoot.depositFor (L53)
- CollectionsBridgeRoot._processMessageFromChild (L101)
- ERC721CollectionV1Validator.constructor (L17)
- ERC721CollectionV2Validator.setFactories (L35)
- ERC721BridgedCollection.mint (L75)
- ERC721BridgedCollection.burn (L91)
- ERC721BridgedCollection.batchTransferFrom (L124)
- ERC721BridgedCollection.safeBatchTransferFrom (L142)

Recommendation: Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

QSP-13 Block Timestamp Manipulation

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

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. Affected Lines :

- ERC721Bid._placeBid (L133);
- ERC721Bid.onERC721Received (L224);
- ERC721Bid._removeExpiredBid (L312);
- ERC721Bid.onERC721Received (L224);
- ERC721Bid._removeExpiredBid (L318).

Recommendation: There is no action to be done except from informing users that bid durations can be slightly off due to the nature of the Ethereum blochchain.

QSP-14 Bids Can Be Made To Any ERC721 Token

Severity: Undetermined

Status: Acknowledged

File(s) affected: `contracts/bid/ERC721Bid.sol`

Description: Bid placement does not rely on a set of configurable token addresses; instead, a bid can be placed for any ERC721 token. It is unclear whether this behavior is intentional or not.

Recommendation: Clarify the issue by providing better code documentation. If not intentional, add setup logic for the owner to register specific token addresses that the platform shall support. In the latter case, upon having a bid, make sure the token has been previously registered.

Automated Analyses

Slither

The results of Slither were inspected manually and were determined to be either false positive or not relevant.

Mythril

Myth did not report any issues.

Code Documentation

- Unclear how the bridge implementation fits the overall picture, as documentation is lacking.
- Highlevel architectural documentation of the system and how different components communicate with one another is currently missing. This is particularly concerning for the bridge code base.

Test Results

Test Suite Results

All tests for the bid contracts (51 test cases) are passing using `npx hardhat test`. For the wearables part, tests currently fail with the following output:

```
Error HH411: The library fx-portal-contracts, imported from contracts/bridges/CollectionsBridgeChild.sol, is not installed. Try installing it using npm.
```

```
Compilation finished successfully
  Contract: Bid
    Place bids
      ✓ should bid an erc721 token (77ms)
      ✓ should bid an erc721 token :: Relayed EIP721 (103ms)
      ✓ should bid a composable erc721 token (58ms)
      ✓ should increment bid counter (114ms)
      ✓ should re-use bid slot when bidder bid and previously has an active bid (444ms)
      ✓ should clean old bid reference when reusing bid slot (192ms)
      ✓ should bid an erc721 token with fingerprint (45ms)
      ✓ reverts when bidding a composable erc721 token whithout fingerprint (85ms)
      ✓ reverts when bidding a composable erc721 token with changed fingerprint (60ms)
      ✓ reverts when bidding an erc721 token with different interface
      ✓ reverts when bidding an address
      ✓ reverts when bidder has not funds
      ✓ reverts when bidder did not authorize bid contract on his behalf (109ms)
      ✓ reverts when placing a bid with 0 as price
      ✓ reverts when bid expires in less than a minute (46ms)
      ✓ reverts when bid expires in more than 6 months (42ms)
      ✓ reverts when bidding an unowned token (71ms)
      ✓ reverts when bidding an owned token (84ms)
    Cancel Bids
      ✓ should cancel a bid (79ms)
      ✓ should cancel a bid :: Relayed EIP721 (140ms)
      ✓ should cancel a bid in a different order from placed (654ms)
      ✓ reverts when cancelling invalid bid (67ms)
      ✓ reverts when cancelling by hacker (58ms)
    Accept Bids
      ✓ should accept a bid for an ERC721 (223ms)
      ✓ should accept a bid for an ERC721 :: Relayed EIP721 (187ms)
      ✓ should accept a bid for a composable ERC721 (152ms)
      ✓ should accept a bid and invalidate the others for the same token (568ms)
      ✓ should simulate an end-2-end (1039ms)
      ✓ reverts when accepting invalid tokenId (143ms)
      ✓ reverts when accepting invalid bidId (191ms)
      ✓ reverts when accepting with insufficient funds (163ms)
      ✓ reverts when accepting without approved contract (300ms)
      ✓ reverts when accepting bid for another token with the same index and id (140ms)
      ✓ reverts when accepting an expired bid (190ms)
      ✓ reverts when accepting with fingerprint changed (72ms)
    Share sale
      ✓ should share sale (171ms)
      ✓ should set to 0 (79ms)
      ✓ reverts when calling by hacker (58ms)
      ✓ reverts when set bigger than 1000000
    Pausable
      ✓ should be paused by the owner (337ms)
      ✓ should be paused by the owner :: Relayed EIP721 (444ms)
      ✓ reverts when pausing by hacker
    Remove Bids
      ✓ should remove an expired bid by bidder (85ms)
      ✓ should remove an expired bid by bidder :: Relayed EIP721 (102ms)
      ✓ should remove an expired bid by anyone (64ms)
      ✓ should remove an expired bid in the middle (374ms)
      ✓ should remove expired bids (578ms)
```

```
      ✓ reverts when removing invalid bid
      ✓ reverts when cancelling a not expired bid (53ms)
      ✓ reverts when calling with different sized arrays (385ms)
    End-2-End
----- Place bids for tokenOne & tokenTwo -----
----- Cancel first tokenOne bid -----
----- Accept third bid placed for tokenOne -----
----- Check counter for tokenOne -----
----- Cancel second tokenTwo bid -----
----- Accept third bid placed for tokenTwo -----
----- Check counter for tokenOne -----
----- Return tokenOne to holder -----
----- Place new bids for tokenOne -----
----- Cancel second and third tokenOne bid -----
----- Accept first bid placed for tokenOne -----
----- Return tokenOne to holder -----
----- Place new bids for tokenOne -----
----- Cancel second and first tokenOne bid -----
----- Accept third bid placed for tokenOne -----
----- Return tokenOne to holder -----
----- Place new bids for tokenOne -----
----- Cancel first and third tokenOne bid -----
----- Accept second bid placed for tokenOne -----
      ✓ should simulate a real case (3456ms)

  51 passing (44s)
```

Code Coverage

The code coverage can and should be pushed to a much higher percentage, ideally at 100%, given the relatively brief nature of the codebase. The testsuite for wearables contracts is present, but currently depend on a dependency that cannot be installed.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
bid/	7.78	2.08	10.53	6.93	
ERC721Bid.sol	7.78	2.08	10.53	6.93	… 552,566,570
ERC721BidStorage.sol	100	100	100	100	
commons/	39.47	18.75	42.11	39.53	
ContextMixin.sol	60	50	100	50	13,14,15
EIP712Base.sol	75	100	66.67	80	61
NativeMetaTransaction.sol	0	0	0	0	… 73,85,95,96
Ownable.sol	80	50	80	81.82	56,57
Pausable.sol	12.5	0	16.67	10	… 75,76,87,88
mocks/	7.37	0	14.29	7.22	
ComposableToken.sol	0	0	12.5	0	… 44,47,51,63
ERC721.sol	6.49	0	6.45	6.41	… 453,457,458
FakeBid.sol	0	100	50	0	21
FakeERC20.sol	0	100	50	0	12
Token.sol	50	100	33.33	50	16,27
TokenWithoutInterface.sol	0	100	33.33	0	12,16
All files	**13**	**3.77**	**19.54**	**12.86**	

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

```
d494a2ee33eb006e50cb9eec903dfc5ea07afc9a6a711aefad29b5806a3024e6 ./contracts/CollectionsBridgeBase.sol
5cf3c0ca394ec4841c0ea5279f30bf3c527aad39ebfec5c94c78139664c1e292 ./contracts/CollectionsBridgeChild.sol
ad1681e4f06c013c89befeef172241c995b9d99f57f7a0d66e0b7fbd74803580 ./contracts/BaseCollectionValidator.sol
2591bdff402412f0d22c293be49611103fe35df96b99cb2c5d88b410b6d42684 ./contracts/ERC721CollectionV2Validator.sol
23877b15218097f61255e7aae168e46649cca43ab6476df219727cb93d57f094 ./contracts/ERC721BidStorage.sol
9f7b1c6776ce226749c912e69a3b1e775fc676f013c9f2e6f59bed2069891620 ./contracts/ERC721BridgedCollection.sol
04d7cf5d25cfbc2912a9bb737043390000f7dc778e8ebd5865fa0d85e59845eb ./contracts/ERC721Bid.sol
3ab6f43405bd91924103181d2dde38fa8f37a0cb59e74b990592982db5eb92f2 ./contracts/ERC721CollectionV1Validator.sol
9dc0908126a8c07182577416acc105b541c3eb3c212c54be82296b6f8d938116 ./contracts/CollectionsBridgeRoot.sol
```

Tests

```
d5395ed2c6c0ae8178aa2e25b283755bf71805b11f060497e01156015b7027db ./CollectionManager.spec.js
84d05b751cb92a0bfb2d55bc40b9d94cf1ad1984ca877a7eb93b3eaa65ee43bb ./CollectionsBridgeChild.spec.js
b9dca169332c87cb2c4e4cc7ebec05ddad123bd635ca95b752f9302cbd4f2c40 ./CollectionsBridgeEnd2End.spec.js
a2061bc7634fd668a7cf5abb29d48c35cf866a249febf73ddeb582fec1359744 ./CollectionsBridgeRoot.spec.js
82f065a8704cbbe0b5c8176659f6b6f20b0a7ef5a591ac1e6274bac337cb9326 ./CollectionStore.spec.js
12823d43fc40601e1c5fa650e948e65b0924772a59c85cc2a600897a3720cb24 ./CollectionsV2End2End.spec.js
4ee22b8e9e76ec58a8cfb8a0e3ad8c357719cd31d58874b5f5de002a6d261a73 ./Committee.spec.js
cb9850ce7a347acd87116f0405cba9939abb6d73350c0f1c8c52f9d675afd2cc ./ERC721BridgedCollection.spec.js
6a1b3f0b4d99d1c032419933e9b18a3d99c4925220912d6639ce82d8e4d93cf7 ./ERC721CollectionFactoryV2.spec.js
8e941c4f4fd0e8a506ba70037dfe114047adf7e9db4bb61f92599a8838af1369 ./ERC721CollectionV1Validator.spec.js
fecab76c0ab78279f5b3035dc877717fdbd0da1dce52304aacd72c5df21c6634 ./ERC721CollectionV2.spec.js
23b8490192f0052a809b197fbe669425213ba968308fb24cf1697749aa9d88ea ./ERC721CollectionV2Validator.spec.js
3c47e56a6625e773133d1eece08ff90f10cdf6026a70e9eba8e63338f49cddf8 ./Forwarder.spec.js
02d4e8790fbe65a0159b77efad06701c1c4d307aef390de26341496a2d96c074 ./Rarities.spec.js
7c4bc6d7134f85862d06f6876bffc969ec15747bd1bb4f4b59eab421cd12362a ./SimpleForwarder.spec.js
ab35b4e4f8a60cce06c48860a94079c71d3a5467dd222576f3bd1203e6e2301b ./BaseCollection.spec.js
568f28cfa8229bc61424e60a84bd2b586f2542460a6c24171bcbadea18f0cc06 ./BurningStore.spec.js
8f9708f7693cc64d75455fffc2ea242a8056296b9a4df38eb5cf236bc70041c6 ./Collection.spec.js
c677c70b55fd4b4a6d6c3780a93f6b54c5a0cf986bbc8f4ccba18d5ae8c23935 ./DeterministicCollection.spec.js
92829bb977ac317d7f8acafcf19beb69298d270766a53a5e22dc4421f4f3e95b ./Donation.spec.js
6cbf8ff1e9513a7fac399b05b06731b235d40a58cab8eb172bfdf24fb69bbe5f ./Factory.spec.js
32ad4cd69bfc37fba7d1523955d07ddbeb49d90c7c186d9f0945333000cd53d1 ./assertRevert.js
be7473de4f606ec5141b43dbaaa106c2e5c0da905902ffdb9eeff7c962437d7e ./balanceSnap.js
fbf054dcae68312745b9d82070124850e09cde6d586211d1ad9136106227db74 ./baseCollection.js
4f44dc609a44aeac0c3c2a98908704617a441125d0bd484310c0bc20d60487f1 ./baseCollectionV2.js
b7c4db51968a96dac9cd71bf4e7fd757cd22865020a60425961a0a0540887d37 ./bridgedCollection.js
df8d2b648e5c8cb107f218df2a7e615cf9092d361270b5bd799f6afbaa1ac999 ./collection.js
da8f1502899cdbf816fe6d62130b72a2a18904f5b11ea6c8fac72991b698fe31 ./collectionV2.js
719c2661cf9430e47c418aa10af24b3231a6109519dc0bcc7e1cea376ec02413 ./increase.js
5ff9d8912feb50fc5f22783e8bba38bdbe964147fb55cab8c53c09051016ffd6 ./metaTx.js
56eaa234eb5ca465d0bc027f4e5f2f79f5de861d9d4fa30edfc7e8582f967386 ./test/Bid.js
ba564232e7dce9e24dc30b2c9139157d659ca6b18a66ca92143d7e163f41c646 ./test/helpers/assertRevert.js
b1da982c66b08e3424ace444945c13f56476dc6c1bb612fa5c22cb8fe9bbe7f3 ./test/helpers/increaseTime.js
d0ccbd8c149f04512ca0b09f5e18ee5b82fc77ccc5f85dec6bfc547b7d941279 ./test/helpers/metaTx.js
```

Changelog

- 2021-08-30 - Initial report
- 2021-09-17 - Re-audit

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.