Quantstamp Security Assessment Certificate

March 29th 2022 — Quantstamp Verified

EnterDao

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

Executive Summary

Type	Period-Based Renting of Land in the Metaverses
Auditors	Fayçal Lalidji, Security Auditor Jose Ignacio Orlicki, Senior Engineer Roman Rohleder, Research Engineer
Timeline	2022-03-07 through 2022-03-29
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	LandWorks Protocol Whitepaper LandWorks Yiel Farming High Level Description
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
LandWorks-protocol	11fbdb8
LandWorks-protocol	232f59b
LandWorks-YF-Contracts	43edba4
LandWorks-YF-Contracts	19a4e4f

Total Issues	**10** (3 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**2** (2 Resolved)
Low Risk Issues	**5** (1 Resolved)
Informational Risk Issues	**3** (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

Initial audit:
Through reviewing the code, we found **10 potential issues** of various levels of severity. We recommend addressing the findings prior to deploying the smart contracts to the main network.

Re-audit:
All highlighted issues have been either fixed or acknowledged.

ID	Description	Severity	Status
QSP-1	Race Condition For Renting	^ Medium	Fixed
QSP-2	Check-Effects-Interactions Pattern Missing	^ Medium	Fixed
QSP-3	Ether `transfer()` Not Backward Compatible	⌄ Low	Acknowledged
QSP-4	URIs Not Curated	⌄ Low	Acknowledged
QSP-5	Uncapped Fees	⌄ Low	Acknowledged
QSP-6	Privileged Roles and Ownership	⌄ Low	Fixed
QSP-7	One Step Ownership Transfer Can Represent Risks	⌄ Low	Acknowledged
QSP-8	Events Not Emitted on State Change	○ Informational	Acknowledged
QSP-9	NFTs Withdrawal Can Be Complicated in Case of ID Loss	○ Informational	Acknowledged
QSP-10	Public Fee Claim Function	○ 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

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 v0.8.0

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Race Condition For Renting

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `LibRent.sol`

Description: Given that you only specify the period for renting, the renter is effectively competing to rent the next free slot (`rentStart = lastRentEnd`). Then is given a condition where the renting order depends on the on-chain race of transactions, in function `rent()`. The latter can be influenced by transactions fees and miners. A renter that is expecting to be the next one, might be front-ran by other renters or blockchain miners. This can lead to moderate financial loss if users plan to rent on some date (now or next time available) but if they are front-ran they can lose the slot and the opportunity cost of an event in the land. Also, currently, rents cannot be canceled.

Recommendation: Allow the user to input in the same transaction the maximum rent start time that he can allow.

Update: Fixed in https://github.com/EnterDAO/LandWorks-protocol/pull/20

QSP-2 Check-Effects-Interactions Pattern Missing

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `LandWorksDecentralandStaking.sol`

Description: `LandWorksDecentralandStaking.withdraw` modifies important state variables after an external call, thus not respecting " Checks effects interaction pattern".

Recommendation: We recommend resetting the value of `stakedAssets[tokenIds[i]]` to zero before the external call.

Update: Fixed in https://github.com/EnterDAO/LandWorks-YF-Contracts/pull/9

QSP-3 Ether `transfer()` Not Backward Compatible

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `LibTransfer.sol`

Description: Due to future changes in the EVM protocol, and given that the transfer function `transfer()` for Ether only sends 2300 gas per call, using this function can be dangerous or unstable. Future changes in the protocol might send more gas (giving opportunity for dangerous reentrancies) or less (more functions calls will fail).

Recommendation: It is recommended to use `call()` in the form `SOME_ADDRESS.call{value: msg.value}("")` while taking into account the checks effects interaction pattern to avoid any possible re-entrancy.

Update: "We are aware of the potential future changes of the EVM protocol. We have decided to postpone this as we will need to make significant changes in order for this to be resolved"

QSP-4 URIs Not Curated

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `ERC721Facet.sol`

Description: Given that cross-site scripting (XSS) are one of the most common web vulnerabilities. If rogue or mismanaged owner keys are handled by a malicious user, then malformed URIs can be introduced using functions `initERC721()` or `setBaseURI()`. These malformed URIs can be at least mitigated using character filtering. This issue has a low impact because it can only be combined with XSS issues found on the web implementation of the application, and with a rogue owner that updates the URIs.

Recommendation: When updating URIs fragments, consider reverting on the following characters (or more) found: `%`, `<`, `>`, `&`, `;`, `'` and `\0`. Also, read Cross Site Scripting Prevention Cheat Sheet.

Update: "We are aware of the potential vulnerability, which will be looked over with caution".

QSP-5 Uncapped Fees

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `LibFee.sol`, `FeeFacet.sol`

Description: Functions `FeeFacet.setFee()` and correspondingly `LibFee.setFeePercentage()` do not constrain the input `_feePercentage` other than the maximum of 100%. Consequently, users can be subject to high fees. This can subject users to high fees with no prior announcements.

Recommendation: Consider adding a `MAX_FEE` bound and state the max fee in public-facing documentation.

Update: "Given that the protocol is upgradeable through a Governance proposal, it requires the same process to change the fee compared to upgrading the protocol to remove the fee cap if we include it now in the protocol".

QSP-6 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Fixed

File(s) affected: `LandWorks-protocol/contracts/*`, `Landworks-YF/contracts/LandWorksDecentralandStaking.sol`

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

- `contractOwner`, as initialized during the constructor of `LandWorks.sol`:
  . Transfer ownership to another address, by calling `LandWorks.transferOwnership()`. **Including the ability to renounce ownership, by setting it to a known uncontrolled address, thereby prohibiting any of the followingly listed actions**.
  . Add, replace or remove "diamond facets" and thereby change any functionality, by calling `LandWorks.diamondCut()`.
  . Change the ERC721 base URI, by calling `LandWorks.setBaseURI()`.
  . Change the protocol fees for a token to an arbitrary value between 0-100%, by calling `LandWorks.setFee()` (or `LandWorks.setTokenPayment()`).
  . Change the protocol payment to an arbitrary token and fee, by calling `LandWorks.setTokenPayment()`.
  . Change the metaverse name of a given ID to an arbitrary string, by calling `LandWorks.setMetaverseName()`.
  . Add or remove registries of a given ID, by calling `LandWorks.setRegistry()`.
  . Add metaverses to LandWorks, by calling `LandWorks.addMetaverse()`.
  . Change the consumable adapter of a metaverse registry to an arbitrary address, by calling `LandWorks.setConsumableAdapter()`.
  . Change the administrative consumer of a metaverse registry to an arbitrary address, by calling `LandWorks.setAdministrativeConsumerFor()`.
  . Change the administrative operator of decentraland to an arbitrary address, by calling `LandWorks.updateAdministrativeOperator()`.
  . The receiving address for all protocol fees, when `FeeFacet.claimProtocolFee()` or `FeeFacet.claimProtocolFees()` are called (which are callable by anyone).

The `owner` role of the `LandWorks-YF` repository, as set during deployment to `msg.sender` is able to:

- Appoint a new `owner`, by calling `transferOwnership()`.
- Renounce his role and therefore block subsequent calls to below mentioned functions, by calling `renounceOwnership()`.
- Modify `rewardRate`, by calling `notifyRewardAmount()`.
- Modify `rewardsDuration` once a period finished, by calling `setRewardsDuration()`.
- Pause or unpause the contract, by calling `pause()` and `unpause()` respectively, thereby disabling/enabling calls to `stake()`.

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

Update: "Information about the ownership and its privilege has been added in LandWorks documentation. https://docs.landworks.xyz/what-is-landworks/ownership https://docs.landworks.xyz/yield-farming#ownership https://docs.landworks.xyz/technical-reference/ownership https://docs.landworks.xyz/technical-reference/decentraland-staking#ownership".

QSP-7 One Step Ownership Transfer Can Represent Risks

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `LibOwnership.Sol`

Description: `LibOwnership` implements a one-step ownership transfer function. If the ongoing owner inputs an incorrect address, the ownership will be transferred to the new address that might be compromised or simply incorrect.

Recommendation: We recommend implementing a two-step ownership transfer mechanism with a second function that set the owner only if called by the new address.

Update: "In order for this to be resolved, we will need to add an additional function, which manipulates the supported interfaces, where EIP173 has been added as a supported interface upon Diamond deployment".

QSP-8 Events Not Emitted on State Change

Severity: *Informational*

Status: Acknowledged

File(s) affected: `LibDiamond.sol`, `LibFee.sol`

Description: An event should always be emitted when a state change is performed in order to facilitate smart contract monitoring by other systems which want to integrate with the smart contract.
This is not the case for the functions and the correspondingly modified state variables:

1. `LibDiamond.addFacet()`, after changing `facetFunctionSelectors[]` and `facetAddresses`.
2. `LibDiamond.addFunction()`, after changing `facetFunctionSelectors[]` and `selectorToFacetAndPosition[]`.
3. `LibDiamond.removeFunction()`, after changing `facetFunctionSelectors[]` and `selectorToFacetAndPosition[]`.
4. `LibFee.distributeFees()`, after changing `assetRentFees[]` and `protocolFees[]`.

Recommendation: Emit an event in the aforementioned functions.

Update: "Given that the protocol is already deployed, making changes to functions related to its upgradeability might impose risk. `distributeFees()` accrues the fees, and its result is later logged in an event".

QSP-9 NFTs Withdrawal Can Be Complicated in Case of ID Loss

Severity: *Informational*

Status: Acknowledged

File(s) affected: `LandWorksDecentralandStaking.sol`

Description: NFT withdrawal from yield farming contract can be complicated if the users or the UI does not save the staked ids since they have to be used as input to be able to get the NFTs back to their original owner.

Recommendation: Ids can be saved in a dynamic array adding to the mapping itself allowing easy retrieval of the users' staked Ids.

Update: "Acknowledged, but adding this change would increase the gas cost significantly".

QSP-10 Public Fee Claim Function

Severity: Informational

Status: Acknowledged

File(s) affected: `FacetFee.sol`

Description: `FacetFee.claimProtocolFee` is publicly callable by any address, thus allowing anyone to send the protocol fee to the owner. If the owner EOA or contract has any issue with token withdrawal the fee amount sent can be lost.

Recommendation: We recommend only allowing the owner itself to withdraw the assets.

Update: "Inline documentation specifying the possibility of lost funds has been added. The business decision is to not require the Governance Proposal to pass every time the protocol fee is to be claimed (https://github.com/EnterDAO/LandWorks-protocol/pull/24)".

Automated Analyses

Slither

The reported issues by slither do not represent a risk of the users' assets and are mainly false positives.

Adherence to Specification

1. The `LandWorks` documentation for `Updating lending conditions` states `paymentToken - the token which will be accepted as a form of payment. Provide 0x0 for ETH`. However, for ETH payments `0x0000000000000000000000000000000000000001` must be used, instead of `0x0`.

Code Documentation

1. The NatSpec comment for parameter `_status` in function `FeeFacet.setTokenPayment()` states that it controls `Whether the token will be approved or not`, however, more precisely it controls whether the provided token should be added (`_status == true`) or should be removed (`_status == false`) from the supported token payments list `tokenPayments`.
2. The NatSpec comment for `LibERC721.mint()` states `Usage of this method is discouraged, use {_safeMint} whenever possible`. However a function `_safeMint` does not exist, only two functions `safeMint`, with different parameters.
3. The NatSpec comment for the return value for `MarketplaceFacet.list()` is missing.

Adherence to Best Practices

1. To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the address parameter(s) in `LibDiamond.DiamondCut()`/`IDiamondCut.DiamondCut()`, `IMetaverseAdditionFacet.SetRegistry()`, `IMarketplaceFacet.SetRegistry()`.
2. According to best practices unused or duplicate code should be removed for readability and maintainability.
    1. In L47 of contract `LibDiamond.sol` the event `DiamondCut()` is a re-declaration from `IDiamondCut.sol` and should therefore be removed.
    2. The code `LibMarketplace.Asset memory asset = ms.assets[_assetId];` in L171 of `MarketplaceFacet.sol` is unused and should therefore be removed.

Test Results

Test Suite Results

```
### LandWorks-YF-Contracts

LandWorks Decentraland Staking
     ✓ Should initialize properly with correct configuration (54ms)
     ✓ Should update rewards duration accordingly
     ✓ Should emit event on update rewards duration
     ✓ Should not update rewards duration if previous reward period as not finished (54ms)
     ✓ Should notifyRewardsAmount accordingly when period finished (66ms)
     ✓ Should notifyRewardsAmount accordingly when period has not finished (79ms)
    Owner
       ✓ Should set owner to deployer
       ✓ Should allow owner to pause
       ✓ Should allow owner to unpause
       ✓ Should not allow nonOnwer to pause
       ✓ Should not allow nonOnwer to unpause
       ✓ Should not allow nonOwner to change rewardsDuration
       ✓ Should not allow nonOwner to notifyRewardAmount

       ✓ Should set reward rate properly
       ✓ Should set reward for duration properly
       ✓ Should revert if reward is too high
       ✓ Should be able to exit (264ms)
       ✓ Should emit correct events when exit (189ms)
      Staking
         ✓ Should stake LandWorks NFTs successfully (149ms)
         ✓ Should update fields correctly on second time staking (233ms)
         ✓ Should emit events correctly (110ms)
```

```
        ✓ Should not allow staking of unsupported metaverse Id (72ms)
        ✓ Should not allow staking of unsupported registry (60ms)
        ✓ Should revert on staking non-existing tokens
        ✓ Should revert on staking non-owned tokens (44ms)
        ✓ Should not allow staking of no tokens
        ✓ Should not allow staking when paused
     Withdrawal
        ✓ Should withdraw staked LandWorks NFTs successfully (131ms)
        ✓ Should withdraw when paused (77ms)
        ✓ Should use the same amount even if estate size changes (99ms)
        ✓ Should emit events correctly on Withdraw (44ms)
        ✓ Should not be able to withdraw LandWorks NFTs staked by other person
        ✓ Should not allow staking of no tokens
     Rewards
        ✓ Should not emit and send if no rewards have been accrued
        ✓ Should accrue correct amount for one holder per second (86ms)
        ✓ Should accrue correct amount for balance > 1 per second (301ms)
        ✓ Should accrue correct amount for multiple users per second (196ms)
        ✓ Should accrue correct amount for multiple users proportionally to their balance per second (270ms)
        ✓ Should emit correct events on Claim (61ms)


  39 passing (4s)

## LandWorks-protocol

Network Info
============
> HardhatEVM: v2.8.2
> network:    hardhat

Creating Typechain artifacts in directory typechain for target ethers-v5
Successfully generated Typechain artifacts!


  Consumable Adapter V1
     ✓ should support IERC721Consumable interface (53ms)
     ✓ should set properties correctly on deploy
     ✓ should update consumer correctly (69ms)
     ✓ should emit event correctly (53ms)
     ✓ should accept only landworks as sender (43ms)
     ✓ should fail when token is not existing
     ✓ should fail when sender is not owner

  LandWorks
Duplicate definition of Transfer (Transfer(address,address,uint256,address,bytes,bytes), Transfer(address,address,uint256,address,bytes), Transfer(address,address,uint256))
Duplicate definition of Transfer (Transfer(address,address,uint256,address,bytes,bytes), Transfer(address,address,uint256,address,bytes), Transfer(address,address,uint256))
    General Diamond Tests
       ✓ should revert if owner is zero address (44ms)
       ✓ should be deployed
       ✓ should have 7 facets
       ✓ has correct function selectors linked to facet (115ms)
       ✓ associates selectors correctly to facets (514ms)
       ✓ returns correct response when facets() is called (84ms)
    DiamondCut Facet
       ✓ should fail if not called by contract owner
       ✓ should allow adding new functions (247ms)
       ✓ should allow replacing functions (207ms)
       ✓ should allow removing functions (202ms)
       ✓ should support all declared interfaces (128ms)
    Ownership Facet
       ✓ should return owner
       ✓ should revert if transferOwnership not called by owner
       ✓ should revert if transferOwnership called with same address
       ✓ should allow transferOwnership if called by owner
    MarketplaceFacet
      setMetaverseName
         ✓ should set metaverse name
         ✓ should emit event with args
         ✓ should revert when caller is not owner
         ✓ should properly set a list of metaverse names (129ms)
      setRegistry
         ✓ should add registry (43ms)
         ✓ should emit event with args
         ✓ should remove registry (65ms)
         ✓ should revert when registry is 0x0
         ✓ should revert when caller is not owner
         ✓ should revert when registry is already added
         ✓ should revert when registry is already removed/never added

    list
       ✓ should list successfully (122ms)
       ✓ should emit event with args (78ms)
       ✓ should list successfully with a payment token (127ms)
       ✓ should revert when metaverse registry is 0x0
       ✓ should revert when min period is 0
       ✓ should revert when max period is 0
       ✓ should revert when min period exceeds max period
       ✓ should revert when max period exceeds max future time
       ✓ should revert when registry is not supported
       ✓ should revert when payment token is not supported
       ✓ should revert when trying to list a non-existing metaverse token id (85ms)
       ✓ should revert when trying to list to a non-contract metaverse registry (58ms)
       ✓ should revert when caller is not owner of the to-be-listed asset (39ms)
       ✓ withdrawing and listing again should not get the old token id for the latest asset (253ms)
    updateConditions
       ✓ should successfully update conditions (43ms)
       ✓ should emit events with args
       ✓ should successfully update conditions when caller is approved (66ms)
       ✓ should successfully update conditions when caller is operator (60ms)
       ✓ should successfully update conditions when caller is consumer (70ms)
       ✓ should revert when asset does not exist
       ✓ should revert when caller is not approved
       ✓ should revert when revert when min period is 0
       ✓ should revert when max period is 0
       ✓ should revert when min period exceeds max period
       ✓ should revert when max period exceeds max future time
       ✓ should revert when payment token is not supported
       ✓ should also claim rent fee on update (71ms)
       ✓ should also claim rent fee to owner on update when caller is not owner, but approved for the asset (78ms)
       ✓ should also claim rent fee to owner on update when caller is not owner, but operator for the asset (88ms)
       ✓ should also claim rent fee to consumer on update when there is consumer set (78ms)
       ✓ should allow consumer to update conditions (87ms)

     delist
        ✓ should successfully delist (115ms)
        ✓ should emit events with args (65ms)
        ✓ should not claim, transfer, burn and clear storage when an active rent exists (78ms)
        ✓ should claim successfully (109ms)
        ✓ should revert when caller is neither owner, nor approved
     withdraw
        ✓ should withdraw successfully (127ms)
        ✓ should emit events with args (96ms)
        ✓ should revert when asset does not exist
        ✓ should revert when caller is not approved
        ✓ should revert when asset is not delisted
        ✓ should revert when an active rent exists (74ms)
     rent
        ✓ should successfully rent (83ms)
        ✓ should emit event with args
        ✓ should calculate new rent from latest and accrue fees (94ms)
        ✓ should revert when asset is not found
        ✓ should revert when trying to rent a delisted asset (61ms)
        ✓ should revert when period is less than asset min period
        ✓ should revert when period is more than asset max period
        ✓ should revert when current rents are more than asset maxFutureTime (50ms)
        ✓ should revert when msg.value is invalid
        ✓ should revert when rent start exceeds max rent start provided
        ✓ should revert when payment token mismatches actual payment token for the asset
        ✓ should revert when provided amount and actual payment amount mismatch
        ✓ should revert when payment token is recently updated (55ms)
        ✓ should revert when price per second is recently updated (42ms)
      using token as payment
         ✓ should rent with ERC20 (104ms)
         ✓ should revert when token value is not approved/invalid (39ms)
         ✓ should revert when payment token mismatches actual payment token for the asset
         ✓ should revert when provided amount and actual payment mismatch (46ms)
         ✓ should revert when payment token is ERC-20 and msg.value is provided
  FeeFacet
    feePrecision
       ✓ should get fee precision
    setTokenPayment
       ✓ should add token payment (51ms)
       ✓ should emit event with args
       ✓ should remove token payment (61ms)
       ✓ should revert when token payment is 0x0
       ✓ should revert when caller is not owner
```

```
    ✓ should revert when token payment is already added
    ✓ should revert when token payment is already removed/never added
    ✓ should revert when fee percentage equal to precision
    ✓ should revert when fee percentage exceeds precision
  setFee
    ✓ should set fee
    ✓ should emit event with args
    ✓ should revert caller is not owner
  claim
    claimProtocolFee
      ✓ should claim ETH protocol fee
      ✓ should claim ETH protocol fee with approved nonOwner
      ✓ should claim token protocol fee (42ms)
      ✓ should claim token protocol fee with nonOwner (42ms)
      ✓ should emit event with args
      ✓ should claim fee even if payment token is removed (64ms)
      ✓ should not emit event with args if protocol fee is zero
    claimProtocolFees
      ✓ should claim protocol fees (58ms)
      ✓ should claim protocol fees with nonOwner (57ms)
      ✓ should emit events with args
      ✓ should claim fees even if payment token is removed (68ms)
    claimRentFee
      ✓ should claim ETH rent fee
      ✓ should claim token rent fee (131ms)
      ✓ should emit event with args (112ms)
      ✓ should revert when caller is not approved
      ✓ should revert when asset is nonexistent
      ✓ should successfully claim rent fees to owner when caller is approved and there is no consumer (48ms)
      ✓ should successfully claim rent fees to owner when caller is operator and there is no consumer (45ms)
      ✓ should successfully claim rent fees to consumer if there is consumer (45ms)
      ✓ should successfully claim rent fees when caller is consumer (49ms)
    claimMultipleRentFees
      ✓ should claim multiple rent fees successfully (75ms)
      ✓ should emit events with args (40ms)
      ✓ should revert when one of assets is not found (62ms)
      ✓ should revert when caller is not approved
      ✓ should successfully claim rent fees to owner when caller is approved (104ms)
      ✓ should successfully claim rent fees to owner when caller is operator (86ms)
Decentraland Facet
  rentDecentraland
    ✓ should successfully rent decentraland (122ms)
    ✓ should emit event with args (52ms)
    ✓ should not update state when rent does not begin in execution block timestamp (83ms)
    ✓ should revert when operator is 0x0
    ✓ should revert when asset is not found
    ✓ should revert when trying to rent a delisted asset (59ms)
    ✓ should revert when period is less than asset min period
    ✓ should revert when period is more than asset max period
    ✓ should revert when current rents are more than asset maxFutureTime (51ms)
    ✓ should revert when msg.value is invalid
    ✓ should revert when rent start exceeds max rent start provided
    ✓ should revert when payment token mismatches actual payment token for the asset
    ✓ should revert when provided amount and actual payment amount mismatch
    ✓ should revert when trying to set operator during rent and contract does not implement setUpdateOperator (176ms)
    ✓ should rent using decentraland facet even if asset is not from a decentraland registry (225ms)
    ✓ should revert when price per second is recently updated
    using token as payment
      ✓ should rentDecentraland with ERC20 (126ms)
      ✓ should revert when token value is not approved/invalid
      ✓ should revert when payment token mismatches actual payment token for the asset
      ✓ should revert when provided amount and actual payment mismatch (42ms)
      ✓ should revert when payment token is ERC-20 and msg.value is provided
  updateState
    ✓ should successfully update state (84ms)
    ✓ should emit event with args (74ms)
    ✓ should revert if asset does not exist
    ✓ should revert if it is not this rent's period  (89ms)
    ✓ should revert if the rent has expired (51ms)
  updateAdministrativeState
    ✓ should successfully update state
    ✓ should emit event with args
    ✓ should revert if asset does not exist
    ✓ should revert if there is an active rent (53ms)
    ✓ should revert if registry does not support setUpdateOperator (141ms)
  updateOperator
    ✓ should successfully update operator (62ms)
    ✓ should emit event with args (56ms)
    ✓ should revert when operator is 0x0
    ✓ should revert if asset does not exist
    ✓ should revert when caller is not renter
  updateAdministrativeOperator
    ✓ should successfully update administrative operator
    ✓ should emit event with args
    ✓ should revert when new administrative operator is 0x0
    ✓ should revert when caller is not owner
  using EstateRegistry
    ✓ should rent estate (117ms)
Diamond cut MetaverseConsumableAdapterFacet
   ✓ should have 8 facets
   ✓ should have correct function selectors linked to facet (107ms)
   ✓ should associate selectors correctly to facets (505ms)
   ✓ should return correct response when facets() is called (68ms)
  setConsumableAdapter
    ✓ should successfully set metaverse registry adapter
    ✓ should emit event with args
    ✓ should revert when metaverse registry is 0x0
    ✓ should revert when adapter is 0x0
    ✓ should revert when caller is not owner
  setAdministrativeConsumerFor
    ✓ should successfully set metaverse registry administrative consumer
    ✓ should emit event with args
    ✓ should revert when metaverse registry is 0x0
    ✓ should revert when administrative consumer is 0x0
    ✓ should revert when caller is not owner

    rentWithConsumer
      ✓ should successfully rent with consumer (102ms)
      ✓ should emit event with args (50ms)
      ✓ should not update adapter when rent does not begin in execution (83ms)
      ✓ should revert when consumer is 0x0
      ✓ should revert when asset is not found
      ✓ should revert when trying to rent a delisted asset (59ms)
      ✓ should revert when period is less than asset min period
      ✓ should revert when period is more than asset max period
      ✓ should revert when current rents are more than asset maxFutureTime (48ms)
      ✓ should revert when msg.value is invalid
      ✓ should revert when rent start exceeds max rent start provided
      ✓ should revert when payment token mismatches actual payment token for the asset
      ✓ should revert when provided amount and actual payment amount mismatch
      ✓ should revert when trying to update adapter which does not implement setConsumer (61ms)
      ✓ should revert when adapter is not set for landworks (87ms)
      ✓ should revert when adapter is not set for the metaverse registry (87ms)
    updateAdapterState
      ✓ should successfully update adapter for rent (90ms)
      ✓ should emit events with args (82ms)
      ✓ should revert if asset does not exist
      ✓ should revert if it is not rent's period (75ms)
      ✓ should revert if the rent has expired (39ms)
      ✓ should revert if adapter does not implement setConsumer (57ms)
      ✓ should revert when adapter is not set for landworks (101ms)
      ✓ should revert when adapter is not set for the metaverse registry (100ms)
    updateConsumer
      ✓ should successfully update consumer (62ms)
      ✓ should emit event with args (68ms)
      ✓ should revert when consumer is 0x0
      ✓ should revert if asset does not exist
      ✓ should revert when caller is not renter
    updateAdapterAdministrativeState
      ✓ should successfully update adapter with administrative consumer
      ✓ should emit events with args
      ✓ should revert if asset does not exist
      ✓ should revert if there is an active rent (55ms)
      ✓ should revert if adapter does not implement setConsumer
      ✓ should revert when adapter is not set for landworks (71ms)
      ✓ should revert when adapter is not set for the metaverse registry (70ms)
    delist
      ✓ should successfully delist with withdraw (117ms)
      ✓ should emit events with args (71ms)
      ✓ should revert when adapter does not implement setConsumer (50ms)
    withdraw
      ✓ should withdraw successfully (132ms)
      ✓ should emit events with args (90ms)
      ✓ should revert when adapter does not implement setConsumer (64ms)
Diamond cut MetaverseAdditionFacet
   ✓ should have 9 facets
   ✓ should have correct function selectors linked to facet (118ms)
```

```
        ✓ should associate selectors correctly to facets (518ms)
        ✓ should return correct response when facets() is called (75ms)
      addMetaverseWithAdapters
         ✓ should successfully add metaverse through MetaverseAdditionFacet (166ms)
         ✓ should emit events with args (48ms)
         ✓ should revert when caller is not owner
         ✓ should revert when registry is 0x0 (48ms)
         ✓ should revert when registries and administrative consumers mismatch
         ✓ should revert when metaverse name is already set
         ✓ should revert when metaverse registries already exist
      addMetaverseWithoutAdapters
         ✓ should successfully add metaverse without adapters through MetaverseAdditionFacet (96ms)
         ✓ should emit events with args (43ms)
         ✓ should revert when caller is not owner
         ✓ should revert when registry is 0x0
         ✓ should revert when registries and administrative consumers mismatch
         ✓ should revert when metaverse name is already set
         ✓ should revert when metaverse registries already exist
         ✓ should be able to change the consumer of a metaverse registry which implements ERC721Consumable (199ms)
         ✓ should clear the consumer of a metaverse registry which implements ERC721Consumable upon withdraw (272ms)
  Cache Bug
     ✓ should not exhibit the cache bug (178ms)

ERC721Facet
   ✓ should have initialised symbols successfully
   ✓ should have initialised name successfully
   ✓ should have initialised base URI successfully
   ✓ should have 1 total supply
   ✓ should revert when already initialized
  balanceOf
     ✓ should have proper balance
     ✓ should revert when owner is 0x0
  ownerOf
     ✓ should return the proper owner of the tokenID
     ✓ should revert when token ID is nonexistent
  getApproved
     ✓ should return approved
     ✓ should revert when tokenID is nonexistent
transfers
  transferFrom
     ✓ should successfully transferFrom when called by owner (97ms)
     ✓ should successfully transferFrom when called by approved (98ms)
     ✓ should successfully transferFrom when called by operator (99ms)
     ✓ should successfully transferFrom when called by owner without approval (105ms)
     ✓ should successfully transferFrom from owner to owner (69ms)
     ✓ should clear consumer when transferred (84ms)
     ✓ should revert when sender is not owner
     ✓ should revert when receiver is 0x0
     ✓ should revert when caller is not authorized
     ✓ should revert when tokenID is nonexistent
  safeTransferFrom(3)
     ✓ should successfully safeTransferFrom when called by owner (84ms)
     ✓ should successfully safeTransferFrom when called by approved (88ms)
     ✓ should successfully safeTransferFrom when called by operator (91ms)
     ✓ should successfully safeTransferFrom when called by owner without approval (101ms)
     ✓ should successfully safeTransferFrom from owner to owner (59ms)
     ✓ should clear consumer when safeTransferFrom (83ms)
     ✓ should revert when sender is not owner
     ✓ should revert when receiver is 0x0
     ✓ should revert when caller is not authorized
     ✓ should revert when tokenID is nonexistent
    to a valid contract
       ✓ should successfully safeTransferFrom when called by owner (94ms)
       ✓ should successfully safeTransferFrom when called by approved (92ms)
       ✓ should successfully safeTransferFrom when called by operator (96ms)
       ✓ should successfully safeTransferFrom when called by owner without approval (107ms)
       ✓ should revert when receiver contract returns unexpected value (131ms)
       ✓ should revert when receiver contract reverts with message (80ms)
       ✓ should revert when receiver contract reverts without message (84ms)
       ✓ should revert when receiver contract reverts without message (80ms)
       ✓ should revert when receiver contract does not implement function
  safeTransferFrom(4)
     ✓ should successfully safeTransferFrom when called by owner (83ms)
     ✓ should successfully safeTransferFrom when called by approved (88ms)
     ✓ should successfully safeTransferFrom when called by operator (87ms)
     ✓ should successfully safeTransferFrom when called by owner without approval (95ms)
     ✓ should successfully safeTransferFrom from owner to owner (62ms)
     ✓ should clear consumer when safeTransferFrom (89ms)
     ✓ should revert when sender is not owner
     ✓ should revert when receiver is 0x0
     ✓ should revert when caller is not authorized
     ✓ should revert when tokenID is nonexistent
    to a valid contract
       ✓ should successfully safeTransferFrom when called by owner (93ms)
       ✓ should successfully safeTransferFrom when called by approved (96ms)
       ✓ should successfully safeTransferFrom when called by operator (95ms)
       ✓ should successfully safeTransferFrom when called by owner without approval (106ms)
       ✓ should revert when receiver contract returns unexpected value (79ms)
       ✓ should revert when receiver contract reverts with message (73ms)
       ✓ should revert when receiver contract reverts without message (82ms)
       ✓ should revert when receiver contract reverts without message (89ms)
       ✓ should revert when receiver contract does not implement function
approve
   ✓ should successfully approve
   ✓ should clear previous approval
   ✓ should re-approve previous
   ✓ should successfully approve when caller is operator
   ✓ should revert when approval receiver is the actual owner
   ✓ should revert when caller is not the owner
   ✓ should revert when caller is approved for the token
   ✓ should revert when tokenID is nonexistent
setApprovalForAll
   ✓ should successfully set operator
   ✓ should successfully unset operator
   ✓ should revert when operator is the owner
setBaseURI
   ✓ should set base URI successfully
   ✓ should emit event with args
   ✓ should get tokenURI properly
   ✓ should get tokenURI properly
tokenURI
   ✓ should revert when tokenID is nonexistent
consumers
   ✓ should successfully change consumer
   ✓ should emit event with args
   ✓ should successfully change consumer when caller is approved (42ms)
   ✓ should successfully change consumer when caller is operator
   ✓ should revert when caller is not owner, not approved
   ✓ should revert when caller is approved for the token
   ✓ should revert when tokenID is nonexistent
   ✓ should revert when calling consumerOf with nonexistent tokenID
  payouts
    transferFrom
       ✓ should payout rent on transferFrom when called from owner (41ms)
       ✓ should payout rent on transferFrom when called by approved (43ms)
       ✓ should payout rent on transferFrom when called by operator (46ms)
    safeTransferFrom
       ✓ should payout rent on safeTransferFrom when called from owner (41ms)
       ✓ should payout rent on safeTransferFrom when called by approved (43ms)
       ✓ should payout rent on safeTransferFrom when called by operator (44ms)
  tokenOfOwnerByIndex
     ✓ should sucessfully return the token ID placed at the given index
     ✓ should revert when index is equal to the total tokens owned by the given address
     ✓ should revert when index is greater than the total tokens owned by the given address
     ✓ should revert when the given address does not own any tokens
     ✓ should switch indexes if one is burnt (353ms)
  tokenByIndex
     ✓ should successfully return the token by index
     ✓ should revert when index is equal to total supply
     ✓ should revert when index is more than total supply
     ✓ should switch indexes if one is burnt (348ms)

349 passing (1m)
```

Code Coverage

Quantstamp usually recommends developers to increase the branch coverage to `90%` and above before a project goes live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase. For branch code coverage, the current targeted files by the audit achieve an acceptable score that can be improved further, `LandWorks-protocol` especially.

LandWorks-YF-Contracts

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
LandWorksDecentralandStaking.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IDecentralandEstateRegistry.sol	100	100	100	100	
IERC721Consumable.sol	100	100	100	100	
ILandWorks.sol	100	100	100	100	
contracts/mocks/	97.14	66.67	100	97.14	
EstateRegistryMock.sol	75	50	100	75	21
LandRegistryMock.sol	100	100	100	100	
MockENTR.sol	100	100	100	100	
MockLandWorksNFT.sol	100	75	100	100	
All files	**99.07**	**93.33**	**100**	**99.07**	

LandWorks-protocol

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
ILandWorks.sol	100	100	100	100	
LandWorks.sol	100	100	100	100	
contracts/adapters/	100	100	100	100	
ConsumableAdapterV1.sol	100	100	100	100	
contracts/facets/	100	100	100	100	
DiamondCutFacet.sol	100	100	100	100	
DiamondLoupeFacet.sol	100	100	100	100	
ERC721Facet.sol	100	100	100	100	
FeeFacet.sol	100	100	100	100	
MarketplaceFacet.sol	100	100	100	100	
MetaverseAdditionFacet.sol	100	100	100	100	
MetaverseConsumableAdapterFacet.sol	100	100	100	100	
OwnershipFacet.sol	100	100	100	100	
contracts/facets/decentraland/	100	100	100	100	
DecentralandFacet.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IDiamondCut.sol	100	100	100	100	
IDiamondLoupe.sol	100	100	100	100	
IERC173.sol	100	100	100	100	
IERC721Consumable.sol	100	100	100	100	
IERC721Facet.sol	100	100	100	100	
IFeeFacet.sol	100	100	100	100	
IMarketplaceFacet.sol	100	100	100	100	
IMetaverseAdditionFacet.sol	**100**	**100**	**100**	**100**	
IMetaverseConsumableAdapterFacet.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IRentPayout.sol	100	100	100	100	
IRentable.sol	100	100	100	100	
contracts/interfaces/decentraland/	100	100	100	100	
IDecentralandFacet.sol	100	100	100	100	
IDecentralandRegistry.sol	100	100	100	100	
contracts/libraries/	92.27	69	100	92.13	
LibDiamond.sol	79.22	44	100	79.01	… 175,177,179
LibERC721.sol	100	91.67	100	100	
LibFee.sol	100	100	100	100	
LibOwnership.sol	100	100	100	100	
LibTransfer.sol	100	100	100	100	
contracts/libraries/marketplace/	100	100	100	100	
LibDecentraland.sol	100	100	100	100	
LibMarketplace.sol	100	100	100	100	
LibMetaverseConsumableAdapter.sol	100	100	100	100	
LibRent.sol	100	100	100	100	
contracts/mocks/	100	100	9.52	100	
Test1Facet.sol	100	100	5	100	
Test2Facet.sol	100	100	100	100	
contracts/mocks/decentraland/	100	100	100	100	
EstateRegistryMock.sol	100	100	100	100	
LANDProxyMock.sol	100	100	100	100	
LANDRegistryMock.sol	100	100	100	100	
contracts/mocks/token/	96.3	80	93.75	92.31	
ERC20Mock.sol	100	100	100	100	
ERC721Consumable.sol	92.86	50	85.71	92.86	76
ERC721Mock.sol	100	100	100	100	
ERC721ReceiverMock.sol	100	100	100	88.89	36
ERC721WithSetUpdateOperatorMock.sol	100	100	100	100	
contracts/shared/	100	100	100	100	
RentPayout.sol	100	100	100	100	
All files	**96.99**	**88.21**	**89.36**	**96.71**	

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

```
0596969908061bd4bdbc84bc82b884ceb2374c63fbc373fe95343747e5169402 ./LandWorksDecentralandStaking.sol
1fa4415164caa743392dc3cc0627c69ab198910c40297f73489ae31652ac0e73 ./IERC721Consumable.sol
d509d6582f0bdafbc01ecb15ed69fa40e6a5e731e46d41456bc3db1f95acdebd ./ILandWorks.sol
f38bb36e879447cdb19027f6f30d7d0381ecc45d6e6cb5f2c19530217f1cafcf ./IDecentralandEstateRegistry.sol
7d347b9fee099dc92c2ed2a4e78ca1c952dad79125656cdc63c311a944252a2e ./contracts/LandWorks.sol
d1e07e1079dce664dc1b6d4bc24b6646b3f0e067fc2d5ad1c44dd6675904b978 ./contracts/ILandWorks.sol
06e31472467c5d64b1dc00887c7dfaf16c3bb42eab9c8a3c60228f2f4d87b857 ./contracts/shared/RentPayout.sol
7a7b9ef94389c6c6b1e76018b672408716b8d3491a423e5cccd5a98d9c4a0264 ./contracts/adapters/IConsumableAdapterV1.sol
8517be9b1cb14392fe5da37b95ad7c41db723a16e9bfe85a7d13ea755ce913fb ./contracts/adapters/ConsumableAdapterV1.sol
ea4c61d904a534c7731460452c05a4049dc3cf47cf6d04ea5e55f7b420d7b480 ./contracts/facets/MarketplaceFacet.sol
2bfa908c7fb20368a95a016313ed11e0eaeedb9334600390ee935e387c2825bc ./contracts/facets/OwnershipFacet.sol
10de98461051652dcd46447eca1f94255e89098be6d129160cccf46cc8296689 ./contracts/facets/FeeFacet.sol
b8b9da82abdc791a272de4d0beb9643261ecbe0faec15ac94ace744fefc81e43 ./contracts/facets/DiamondLoupeFacet.sol
cb1ba98c2836525b76f20666be1c73ac373da7c879f0146632db0bd9b2fc96a0 ./contracts/facets/MetaverseAdditionFacet.sol
1d40eeb1a8976da8ee1863947dfca519dd10594387ed2ba0e4aed946354edd42 ./contracts/facets/DiamondCutFacet.sol
8f8ad2f438857578a2dd23b0e02d50a2cf969c7d063513041da074541c3db9ed ./contracts/facets/MetaverseConsumableAdapterFacet.sol
f9faca21885928b28052d4d11883c79997675b77e07c30c298e9b2504f3b81f7 ./contracts/facets/ERC721Facet.sol
a072f00d3ce7d9775e723ad28844333039919d1a15e09a872a65b518978d8a6f ./contracts/facets/decentraland/DecentralandFacet.sol
1b2135ed2a4fe7c248727a080ade8537d7cf04ac3fe1bb97edf0412d51043b64 ./contracts/interfaces/IFeeFacet.sol
a4a8ba8f6c7f2dd16d8f64a0f8596125c7d5f80c02c1f00833fdd7998e686026 ./contracts/interfaces/IDiamondLoupe.sol
82faea8557ab6806faf2622fdabafb0571ed601cc399085d450fd24ae255f099 ./contracts/interfaces/IDiamondCut.sol
3812d12ed43e5ace0cda36f75d2e69887ff2ae06eae0d951bed85a969d48dcae ./contracts/interfaces/IERC721Facet.sol
ca05de29903a80aac2963f58e118964ec7af77f1670eeb3ca00779447b1c06e9 ./contracts/interfaces/IRentable.sol
0f2d0227df0b5057b4027f99a7bd4ee738a3e2ddaeab3f4a0f28003f6febe334 ./contracts/interfaces/IERC173.sol
2ca21543b44ee62694e13957ced45e14d29a00c10ffbc5bef64085f72850d9d2 ./contracts/interfaces/IMetaverseConsumableAdapterFacet.sol
679587a22bcbc0f04af7e91a36dd10a507b92effe62aea0372bd004c67c61337 ./contracts/interfaces/IERC721Consumable.sol
bc58615b059df2360c7fa03c889598452a085d1d2ab11561adca27b4e79c46f5 ./contracts/interfaces/IMetaverseAdditionFacet.sol
a2c0af71b1ea62ef6ee6e568ab6bc034e52810a98087ef50f4921875bedc6f5f ./contracts/interfaces/IRentPayout.sol
5b62bba7e5c125231bb57b6e714f67d1a695977dc6afbc9eab9445152f11eaaa ./contracts/interfaces/IMarketplaceFacet.sol
fd4abaf3752739ebaf7238944cd751a8b81339b1d8f8d842fd8f9f7d3342963d ./contracts/interfaces/decentraland/IDecentralandFacet.sol
62bde64193328fb32e09d65de0703dde9cb3605a253fe11657647b09c7cb2cc6 ./contracts/interfaces/decentraland/IDecentralandRegistry.sol
1b5c2d2dd6b726b50049795272c0ee2084afebe65d5a69f01df50e72c243d0ef ./contracts/libraries/LibTransfer.sol
41dc093306bf83a035036cdfed234b9f72a5bdc4e746b2682e49cf2f1c55e0b7 ./contracts/libraries/LibOwnership.sol
7f1b9f5302296aa716c25176352cfb59cf78286bb93631e7000283e458a1a320 ./contracts/libraries/LibFee.sol
21348def3be2f39716f1a0f270e84fa6f88cc36553bc709c5860a7b1d7b6e4a4 ./contracts/libraries/LibERC721.sol
f52d6b52a2592ce7b9af0a7b0d6b66a0a9f925aa417b25f26b9601a9abd047c8 ./contracts/libraries/LibDiamond.sol
4637217bb2ed073b3f59bce0eabfdabd8f3330a4cbf0b4c163e715d472549504 ./contracts/libraries/marketplace/LibMetaverseConsumableAdapter.sol
bff30fd8b26c2baf302469721e496d3c3ec36ee94b08bd11c394abacd407970d ./contracts/libraries/marketplace/LibMarketplace.sol
b54d4584b0cb651f7ef0a25c7427123315d89f001d1a9c46ae841ced78f345a4 ./contracts/libraries/marketplace/LibRent.sol
0bad5c3b0bd01e156a494a627488bead265afa30065680083c3738acdbd0d8cd ./contracts/libraries/marketplace/LibDecentraland.sol
```

Tests

```
949142ec673a328c8adfbe6dcb91bc9c31b447d7c06c50a604d6f1f12e7b2793 ./diamond.test.ts
a2f21b4a3b4c1fe65cf631cf3534957fda992ca7c1c1bc4fb56c4a2cd528a061 ./adapter.test.ts
9877e04a2835918974a5b0288ade5f8dad38cbd9490137a4d76e247c198bc16e ./erc721facet.test.ts
7023132907eecc4dcdbc441e50f4b4c04a934c4739cc4177227f0c028e07ea75 ./test/decentraland-staking.ts
```

Changelog

- 2022-03-16 - Initial report (11fbdb8)
- 2022-03-28 - Re-audit report update

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.