Quantstamp Contract Security Certificate

June 19th 2020 — Quantstamp Verified

Decentraland

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

Executive Summary

Type	ERC721 Contract with MiniMe Integration
Auditors	Ed Zulkoski, Senior Security Engineer Leonardo Passos, Senior Research Engineer Joseph Xu, Technical R&D Advisor
Timeline	2020-06-15 through 2020-06-21
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	LAND Contracts Documentation MiniMe Contract Documentation
Documentation Quality	High
Test Quality	Undetermined

Source Code

Repository	Commit
land	d01256f6

Goals

- Do the registries correctly integrate with the `MiniMeToken` contract?
- Are the proxy changes implemented correctly?
- Are `MiniMe` balances correctly computed?

Changelog

- 2020-06-19 - Initial report

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

This report pertains to the added proxy and MiniMe functionality added in PR-150 of Decentraland. No critical issues were found, however caution should be used when deploying new versions of any upgradeable contracts. In particular, if the `MiniMeToken` contract referenced through `landBalance` is updated, care should be taken to ensure that the new `MiniMeToken` balances maintain parity with the existing registered balances, else subsequent transfers may fail. We also noted several low severity issues, and we further suggest minor changes for gas-cost improvement and adherence to best practices.
We were unable to fully evaluate the quality of the test suite due to errors running `solidity-coverage`. We strongly recommend adding instructions for producing coverage results in the `README.md` (including tool version numbers and scripts if necessary).
Note: this report pertained primarily to the changes made in PR-150. The existing code was examined for context, but not thoroughly audited.

ID	Description	Severity	Status
QSP-1	Transfers may fail if the `landBalance` token is updated	^ Medium	Unresolved
QSP-2	`getLANDsSize()` could run out of gas for large balances	^ Medium	Unresolved
QSP-3	`createCloneToken()` functionality differs from the specified logic	v Low	Unresolved
QSP-4	Missing function input validation	v Low	Unresolved
QSP-5	`registerBalance()` is a no-op if the user is already registered	o Informational	Unresolved
QSP-6	Clone-and-Own	o Informational	Unresolved
QSP-7	Unclear discrepancy in proxy logic for registry contracts	o Informational	Unresolved
QSP-8	Unlocked Pragma	o Informational	Unresolved
QSP-9	Privileged Roles and Ownership	o Informational	Unresolved
QSP-10	Unclear use of multiple `landBalance` references	? Undetermined	Unresolved

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:

- Truffle
- Ganache
- SolidityCoverage
- Mythril
- Slither

Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed Ganache: `npm install -g ganache-cli`
3. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
4. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
5. Installed the Mythril tool from Pypi: `pip3 install mythril`
6. Ran the Mythril tool on each contract: `myth -x path/to/contract`
7. Installed the Slither tool: `pip install slither-analyzer`
8. Run Slither from the project directory `slither .`

Assessment

Findings

QSP-1 Transfers may fail if the `landBalance` token is updated

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `LANDRegistry.sol`, `EstateRegistry.sol`

Description: The registry contracts maintain local `registeredBalance` mappings to determine if an address is registered for use with the `MinimeToken` contract. If the referenced `landBalance` contract is updated by the proxy owner using `setLandBalanceToken()` (e.g., to a fresh `MinimeToken` contract), any transfers from registered users may fail. This is due to the internal `updateLandBalance()` function, which would invoke `landBalance.destroyTokens(_from, 1)` (as on L568 of `LandRegistry.sol`), and would consequently revert due to L403 of `MinimeToken.sol`: `require(curTotalSupply >= _amount);`.

Recommendation: If the `landBalance` contract is updated, it should be ensured that there is parity between the token balances in the `MinimeToken` contract and the registry contracts. Upgraded `landBalance` contracts should likely utilize the `createCloneToken()` function in `MinimeToken.sol` in such a scenario.

QSP-2 `getLANDsSize()` could run out of gas for large balances

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `EstateRegistry.sol`

Description: The mapping `ownedTokensCount` returns an `uint256`; hence, in theory, one could own a lot of tokens. If invoked directly, outside a transaction, the function `getLANDsSize()` does not present any issue. However, `getLANDsSize()` is called from `registerBalance()`, which executes in the context of a transaction. If the balance is too high, the transaction could fail, as it will run out of gas. Consequently, user with large amounts of tokens may not be able to register their balance in `landBalance`.

Recommendation: Perform gas analysis to determine if this will be an issue or not in practice. If necessary, change the logic to allow registering portions of a balance; then, if users want to register all of their balance, they could make multiple transactions.

QSP-3 `createCloneToken()` functionality differs from the specified logic

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `MinimeToken.sol`

Description: The `createCloneToken()` functionality differs from the specified logic in the comment above when the `_snapshotBlock` is 0. In this case, `snapshot` is set to `block.number - 1`, whereas the comment suggests that it should be set to `block.number` (as is the case in https://github.com/Giveth/minime/blob/master/contracts/MiniMeToken.sol).

Recommendation: Remove the `- 1`, or provide clarification as to why this change was implemented.

QSP-4 Missing function input validation

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `LANDRegistry.sol`

Description: There are two functions that could use additional input validation:

1. In the function `assignMultipleParcels()`, it is not verified that the input arrays `x` and `y` are of the same length. If the lengths differ, the function will revert and the `deployer` will lose gas as a consequence.
2. In `setLandBalanceToken()`, it should be ensured that `_isContract(_newLandBalance)` holds.

Recommendation: Add a require-statement to ensure that `x.length == y.length` in the `assignMultipleParcels()` function. Add a require-statement that `_isContract(_newLandBalance)` holds in `setLandBalanceToken()`.

QSP-5 `registerBalance()` is a no-op if the user is already registered

Severity: *Informational*

Status: Unresolved

File(s) affected: `LANDRegistry.sol`, `EstateRegistry.sol`

Description: In both registry contracts, if the user tries to register an already registered account with `registerBalance()`, this will first destroy the allocated token balance and then re-generate the same amount. This is gas-inefficient, and also creates two unnecessary `Transfer` events.

Recommendation: If the user is already registered, simply return from the function immediately.

QSP-6 Clone-and-Own

Severity: *Informational*

Status: Unresolved

File(s) affected: `MinimeToken.sol`

Description: The clone-and-own approach involves copying and adjusting open source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability, or may include intentionally or unintentionally modified upstream libraries.

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries.
This particular contract is available using `npm i minimetoken`.

QSP-7 Unclear discrepancy in proxy logic for registry contracts

Severity: *Informational*

Status: Unresolved

File(s) affected: `LANDRegistry.sol`, `EstateRegistry.sol`

Description: The state variable `proxyOwner` is inherited from two separate dependencies for the registry contracts:

1. In `LANDRegistry.sol` inherits `proxyOwner` through `contracts/upgradable/Ownable.sol`, which inherits `contracts/Storage.sol` that further inherits `contracts/upgradable/ProxyStorage`.
2. `EstateRegistry.sol` inherits `openzeppelin-zos/contracts/ownership/Ownable.sol` and specifies `proxyOwner` directly in `EstateStorage`.

It is not clear why the same dependencies are not used in both registries.

Recommendation: Clarify whether this inheritance hierarchy is implemented as intended.

QSP-8 Unlocked Pragma

Severity: *Informational*

Status: Unresolved

File(s) affected: `All Contracts`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. 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-9 Privileged Roles and Ownership

Severity: *Informational*

Status: Unresolved

File(s) affected: `LANDRegistry.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.
In `LANDRegistry.sol`, by means of `assignNewParcel()`, the deployer could give themself as much land as wanted.

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

QSP-10 Unclear use of multiple `landBalance` references

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `LANDRegistry.sol`, `EstateRegistry.sol`

Description: Both registry contracts have a reference to a `landBalance` contract. It is not clear if these are intended to refer to the same `MinimeToken` contract or not.

Recommendation: If it is expected that both registry contracts refer to the same `MinimeToken`, require-statements should be added to `EstateRegistry.setLandBalanceToken()` to ensure that it is the same as in `LandRegistry()`. Documentation should be added in either case clarifying these references.

Automated Analyses

Mythril

Mythril did not report any issues.

Slither

Slither detected that the return-values of `generateTokens()` and `destroyTokens()` are never checked, however since these functions always return true, this is inconsequential.

Adherence to Specification

The code adheres to the provided specifications, aside from minor deviations already discussed above.

Code Documentation

The code is generally very well documented. We have only a few minor suggestions:

1. A comment could be added to L602 of `EstateRegistry.sol` indicating the significance of address `0x9a6ebe7e2a7722f8200d0ffb63a1f6406a0d7dce` (as it relates to the proxy logic).
2. In `MinimeToken.sol`, the version 'MMT_0.1' does not align the latest version 'MMT_0.2' in the Giveth repository https://github.com/Giveth/minime/blob/master/contracts/MiniMeToken.sol.
3. Documentation should be added alerting users that they will have a zero `landBalance` until they register for the feature.

Adherence to Best Practices

The code generally follows best practices, with a few minor exceptions:

1. There are a few locations where `uint` is used instead of `uint256`. These should be standardized to `uint256` throughout.
2. On L116 in `EstateRegistry.sol`, the constant 0 should be explicitly casted to `address(0)`.
3. In `LANDRegistry.sol`, `assignNewParcel()` does not check if `beneficiary` is different from 0x0. Add such a check; if equal to 0x0, revert.
4. On L143 of `EstateRegistry.sol`, explicitly initialize `i`, instead of relying on its default implicit value (zero).

Test Results

Test Suite Results

```
  Contract: EstateRegistry
    name
```

```
    ✓ has a name
  symbol
    ✓ has a symbol
  set LAND Registry
    ✓ set works correctly (103ms)
    ✓ should throw if setting a non-contract
    ✓ unauthorized user can not set registry (80ms)
  create Estate
    ✓ the registry can create estates (457ms)
    ✓ only the registry can create estates
    ✓ supports setting the metadata on create (204ms)
    ✓ should emit the CreateEstate event on mint (103ms)
    ✓ should allow operator to create an estate (232ms)
    ✓ fails if sender is not owner or operator of all LANDs (200ms)
    ✓ fails if somebody else tries to steal LAND (82ms)
  transfer many Estates
    ✓ the owner can transfer many estates (461ms)
    ✓ only the owner can transfer many estates (373ms)
  update metadata and update operator
    ✓ update works correctly :: holder (217ms)
    ✓ update works correctly :: updateOperator (259ms)
    ✓ update works correctly :: operator (266ms)
    ✓ update works correctly :: approved for all (274ms)
    ✓ unauthorized user can not update (200ms)
    ✓ unauthorized user can not set update operator (195ms)
    ✓ update operator can not transfer tokens out (225ms)
    ✓ should not allow an owner of an Estate transfer tokens out from an Estate which is not the owner
(366ms)
  transfer tokens
    ✓ owner can transfer tokens in (311ms)
    ✓ transfering tokens in fires the AddLand event (381ms)
    ✓ user cannot transfer tokens to an undefined estate (91ms)
    ✓ random user can transfer tokens in (319ms)
    ✓ owner can transfer tokens out (273ms)
    ✓ random user can not transfer tokens out (191ms)
    ✓ random user can not transfer many tokens out (217ms)
    ✓ owner can not transfer tokens out to the empty address (191ms)
    ✓ transfering tokens out should emit the RemoveLand event (253ms)
    ✓ owner can transfer many tokens out (2837ms)
  operator transfering tokens
    ✓ operator can transfer tokens out (388ms)
    ✓ operator can transfer many tokens out (428ms)
    ✓ operator can not transfer tokens out after deauth (414ms)
    ✓ operator can not transfer many tokens out after deauth (430ms)
  order of tokens is correctly accounted
    ✓ five in, middle out, one in, middle out (786ms)
    ✓ five in, empty, refill (1335ms)
  fingerprint management
    ✓ supports verifyFingerprint interface
    ✓ creates the fingerprint correctly (621ms)
    ✓ should change the fingerprint as the composable children change (1151ms)
    ✓ should encode only the id on empty estates (229ms)
    ✓ should generate the same hash even if the parcel order changes (1639ms)
    ✓ verifies the fingerprint correctly (862ms)
    ✓ should not have checksum collision with one LAND (399ms)
    ✓ should not have checksum collision with multiple LANDs (914ms)
  LAND update
    ✓ should allow owner of an Estate to update LAND data (288ms)
    ✓ should allow operator of an Estate to update LAND data (273ms)
    ✓ should allow update operator of an Estate to update LAND data (276ms)
    ✓ should allow a LAND updateOperator to update LAND data (277ms)
    ✓ should not allow owner an Estate to update LAND data of an Estate which is not the owner (388ms)
    ✓ should not allow neither operator, nor owner nor updateOperator nor LAND updateOperator of an Estate
to update LAND data (226ms)
    ✓ should not allow old owner to update LAND data after creating an Estate (209ms)
    ✓ should not allow old operator to update LAND data after creating an Estate (249ms)
  LANDs update
    ✓ should allow owner of an Estate to update LANDs data (349ms)
    ✓ should allow operator of an Estate to update LANDs data (403ms)
    ✓ should allow update operator of an Estate to update LANDs data (464ms)
    ✓ should not allow owner an Estate to update LANDs data of an Estate which is not the owner (371ms)
    ✓ should not allow neither operator nor owner nor updateOperator of an Estate to update LANDs data
(556ms)
    ✓ should not allow old owner to update LANDs data after creating an Estate (362ms)
    ✓ should not allow old operator to update LANDs data after creating an Estate (414ms)
  support interfaces
    ✓ should support InterfaceId_GetMetadata interface
    ✓ should support inherited InterfaceId_ERC721 and InterfaceId_ERC721Exists interfaces
    ✓ should not support not defined interface
  Update Operator
    ✓ should clean update operator after transfer the Estate :: safeTransferFrom (547ms)
    ✓ should clean update operator after transfer the Estate :: safeTransferFrom with bytes (402ms)
    ✓ should clean update operator after transfer the Estate :: transferFrom (671ms)
    ✓ should clean update operator after transfer the Estate :: safeTransferManyFrom (933ms)
    ✓ should set an update operator by updateManager (393ms)
  Update LAND Update Operator
    ✓ should update LAND updateOperator by estate owner (308ms)
    ✓ should update LAND updateOperator by estate operator (323ms)
    ✓ should update LAND updateOperator by updateManager (325ms)
    ✓ should clean LAND updateOperator (453ms)
    ✓ reverts when updating LAND updateOperator by estate updateOperator (262ms)
    ✓ reverts when updating LAND updateOperator for a LAND outside the estate (224ms)
    ✓ reverts when updating LAND updateOperator for a LAND from another estate (433ms)
    ✓ reverts when updating LAND updateOperator by hacker (212ms)
  UpdateManager
    ✓ should set updateManager by owner (166ms)
    ✓ should set updateManager by approvedForAll (132ms)
    ✓ should allow updateManager to update content (246ms)
    ✓ should allow updateManager to update content on new Estate (375ms)
    ✓ should allow updateManager to update content on LANDs as part of the Estate (127ms)
    ✓ should has false as default value for updateManager
    ✓ should set multiple updateManager (108ms)
```

```
      ✓ clears updateManager correctly  (183ms)
      ✓ reverts when updateManager trying to change content of no owned by the owner Estate (193ms)
      ✓ reverts if owner set himself as updateManager
      ✓ reverts if not owner or approvedForAll set updateManager (191ms)
      ✓ reverts when updateManager trying to transfer (58ms)
      ✓ reverts when updateManager trying to set updateManager (55ms)
      ✓ reverts when updateManager trying to set operator (69ms)
      ✓ reverts when updateManager trying move LANDs from Estate (60ms)
    setManyUpdateOperator
      ✓ should set update operator (65ms)
      ✓ should set many update operator :: owner (119ms)
      ✓ should set many update operator :: approvedForAll (144ms)
      ✓ should set many update operator :: operator (185ms)
      ✓ should set many update operator :: updateManager (145ms)
      ✓ should clean many update operator (145ms)
      ✓ reverts when updateOperator try to set many update operator (63ms)
      ✓ reverts when unauthorized user try to set many update operator
    setManyLandUpdateOperator
      ✓ should set LAND update operator (75ms)
      ✓ should set many LAND update operator :: owner (197ms)
      ✓ should set many LAND update operator :: approvedForAll (293ms)
      ✓ should set many LAND update operator :: operator (435ms)
      ✓ should set many LAND update operator :: updateManager (427ms)
      ✓ should clean many LAND update operator (336ms)
      ✓ reverts when updateOperator try to set many LAND update operator (61ms)
      ✓ reverts when setting LAND updateOperator for a LAND outside the estate (248ms)
      ✓ reverts when unauthorized user try to set many update operator
    LANDs size
      ✓ should return the amount of LANDs (996ms)
      ✓ should update the amount of LANDs (2239ms)
      ✓ should returns 0 for an address with 0 Estates
    ProxyOwner
      ✓ should init proxyOwner to dcl Multisig (64ms)
      ✓ should transfer proxyOwner (115ms)
      ✓ reverts when transferring proxyOwner by unauthorized user (78ms)
      ✓ reverts if trying to init proxyOwner after transferred (139ms)
      ✓ reverts if trying to init proxyOwner more than once (85ms)
    LandBalance
      setBalanceToken
        ✓ should set balance token
        ✓ reverts if a hacker try to set balance token
      Register balance
        ✓ should register balance (462ms)
        ✓ should re-register balance (753ms)
        ✓ should unregister balance (170ms)
      Update balance
        ✓ should register balance only one balance (138ms)
        ✓ should register owner balance if it was transferred by operator :: approvalForAll (174ms)
        ✓ should register owner balance if it was transferred by operator :: Operator (204ms)
        ✓ should register balance both (215ms)
        ✓ should not register transfer to the land registry (614ms)
        ✓ should register land balance and estate balance (1139ms)
        ✓ should update on transfer :: transferFrom (217ms)
        ✓ should update on transfer :: safeTransferFrom (234ms)
        ✓ should update on transfer :: safeTransferFrom with bytes (214ms)
        ✓ should update on transfer :: safeTransferManyFrom (514ms)

  Contract: LANDRegistry
    Combinations of calls
      ✓ before transfer, update is possible, after, it is not (193ms)
      ✓ before owning, update is impossible, after, it is not (294ms)

  Contract: LANDProxy
    upgrade
      ✓ should upgrade proxy by owner (104ms)
      ✓ should throw if not owner upgrade proxy
      ✓ should transfer ownership
      ✓ should throw if transfering to address 0x0
      ✓ should throw if trying to transfer and not owner

  Contract: LANDRegistry
    name
      ✓ has a name
    symbol
      ✓ has a symbol
    totalSupply
      ✓ has a total supply equivalent to the inital supply
      ✓ has a total supply that increases after creating a new LAND (118ms)
    new parcel assignment,
    one at a time:
      ✓ only allows the creator to assign parcels
      ✓ allows the creator to assign parcels (65ms)
    multiple
      successfully registers 10 parcels
        ✓ works for 4,-3
        ✓ works for -8,9 (60ms)
        ✓ works for 16,-27
        ✓ works for -32,81
        ✓ works for 64,-243
        ✓ works for -128,729
        ✓ works for 256,-2187
        ✓ works for -512,6561
        ✓ works for 1024,-19683
        ✓ works for -2048,59049
    tokenId
      encodeTokenId
        ✓ correctly encodes 0,0 (38ms)
        ✓ correctly encodes 0,1
        ✓ correctly encodes 1,0
        ✓ correctly encodes 0,-1
        ✓ correctly encodes -1,-1
        ✓ correctly encodes 0,256
```

```
      ✓ correctly encodes -256,0
      ✓ correctly encodes -23,-23
    decodeTokenId
      ✓ correctly decodes 0x0000000000000000000000000000000000000000000000000000000000000000
      ✓ correctly decodes 0x0000000000000000000000000000000000000000000000000000000000000001
      ✓ correctly decodes 0x0000000000000000000000000000000100000000000000000000000000000000
      ✓ correctly decodes 0x00000000000000000000000000000000ffffffffffffffffffffffffffffffff
      ✓ correctly decodes 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
      ✓ correctly decodes 0x0000000000000000000000000000000000000000000000000000000000000100
      ✓ correctly decodes 0xffffffffffffffffffffffffffffffff0000000000000000000000000000000
      ✓ correctly decodes 0xfffffffffffffffffffffffffffffffe9fffffffffffffffffffffffffffffe9
  getters
    ownerOfLand
      ✓ gets the owner of a parcel of LAND
    ownerOfLandMany
      ✓ gets the address of owners for a list of parcels (64ms)
    landOf
      ✓ gets the parcel coordinates for a certain owner
    exists
      ✓ returns true if the parcel has been assigned
      ✓ returns false if the has not been assigned
      ✓ throws if invalid coordinates are provided
      ✓ throws if no coordinates are provided
    landData
      ✓ returns an empty string for a freshly-assigned parcel
      ✓ allows updating your own LAND data (59ms)
      ✓ throws if updating another user LAND data
      ✓ allow updating LAND data if given authorization (273ms)
      ✓ returns LAND data for a parcel that belongs to another holder (530ms)
      ✓ returns an empty string for a set of coordidnates with no associated parcel (66ms)
      ✓ throws if invalid coordinates are provided
      ✓ throws if no coordinates are provided
    updateLandData
      ✓ updates the parcel data if authorized :: operator (121ms)
      ✓ updates the parcel data if authorized :: approve for all (118ms)
      ✓ updates the parcel data if authorized :: update operator (113ms)
      ✓ sets an empty string if invalid data is provided (167ms)
      ✓ reverts if the sender is not an authorized operator
      ✓ emits Update event on LAND update (57ms)
    authorizeDeploy
      ✓ authorizes an address (56ms)
      ✓ verifies that deployments are not authorized by default
      ✓ reverts if address is already authorized  (57ms)
      ✓ reverts if authorizing invalid address
      ✓ reverts if the sender is not the owner
      ✓ should use proxy owner to validate deploy call (67ms)
      ✓ should use proxy owner to validate forbid call (65ms)
      ✓ reverts if user tries to assign LAND and it not deployer
      ✓ deployer must be able to assign new LAND (89ms)
      ✓ emits DeployAuthorized event (44ms)
    forbidDeploy
      ✓ reverts if address is already forbidden
      ✓ forbids the deployment for an specific address after authorization (130ms)
      ✓ reverts if the sender is not the owner
      ✓ reverts if deauthorize invalid address
      ✓ emits DeployForbidden event (76ms)
  Transfers
    transfer from
      ✓ does not transfer LAND if the destinatary is the EstateRegistry (40ms)
    transferLand
      ✓ transfers LAND if it is called by owner (94ms)
      ✓ transfers LAND if it is called by operator (130ms)
      ✓ does not transfer LAND if it is called by not authorized operator
      ✓ does not transfer LAND if land does not exist
    transferManyLand
      ✓ transfers LANDs if it is called by owner (144ms)
      ✓ transfers LANDs if it is called by operator (176ms)
      ✓ does not transfer LANDs if it is called by not authorized operator (44ms)
      ✓ does not transfer LANDs if land does not exist
      ✓ does not transfer LANDs if x length is not equal to y length
  transfer LAND to estate
    transferLandToEstate
      ✓ should not transfer the LAND to an Estate if it not is owned by the sender (69ms)
      ✓ transfers LAND to an Estate if it is called by owner (133ms)
      ✓ does not transfer LAND if it is called by not authorized operator
      ✓ does not transfer LAND if LAND does not exist
    transferManyLandToEstate
      ✓ should not transfer the LANDs to an Estate if it is not owned by the sender (68ms)
      ✓ transfers LANDs if it is called by owner (245ms)
      ✓ does not transfer LANDs if it is called by not authorized operator (52ms)
      ✓ does not transfer LANDs if land does not exist
      ✓ does not transfer LANDs if x length is not equal to y length
  update authorized
    ✓ update not allowed before setUpdateOperator
    ✓ update allowed after setUpdateOperator (104ms)
    ✓ update disallowed after setUpdateOperator to different address (146ms)
    ✓ update disallowed after transfer (133ms)
    ✓ update operator emits UpdateOperator event (57ms)
    ✓ should set an update operator by an operator (103ms)
    ✓ should set an update operator by an operator approved for all (101ms)
    ✓ should set an update operator by updateManager (101ms)
    ✓ reverts if not owner want to update the update operator (49ms)
    ✓ should be clear on transfer :: transferFrom (162ms)
    ✓ should be clear on transfer :: safeTransferFrom (230ms)
    ✓ should be clear on transfer :: transferLand (158ms)
    ✓ should be clear on transfer :: transferManyLand (293ms)
    ✓ should be clear on transfer :: transferLandToEstate (388ms)
    ✓ should be clear on transfer :: transferManyLandToEstate (549ms)
  UpdateManager
    ✓ should set updateManager by owner (104ms)
    ✓ should set updateManager by approvedForAll (138ms)
    ✓ should allow updateManager to update content (186ms)
```

```
    ✓ should allow updateManager to update content on new LANDs (158ms)
    ✓ should has false as default value for updateManager
    ✓ should set multiple updateManager (100ms)
    ✓ clears updateManager correctly  (225ms)
    ✓ reverts when updateManager trying to change content of no owned by the owner LAND (369ms)
    ✓ reverts if owner set himself as updateManager
    ✓ reverts if not owner or approvedForAll set updateManager (185ms)
    ✓ reverts when updateManager trying to transfer (64ms)
    ✓ reverts when updateManager trying to set updateManager (65ms)
    ✓ reverts when updateManager trying to set operator (68ms)
    ✓ reverts when updateManager trying to set create an Estate (77ms)
    ✓ reverts when updateManager trying to assign LANDs (58ms)
  setManyUpdateOperator
    ✓ should set update operator (95ms)
    ✓ should set many update operator :: owner (119ms)
    ✓ should set many update operator :: approvedForAll (147ms)
    ✓ should set many update operator :: operator (199ms)
    ✓ should set many update operator :: updateManager (159ms)
    ✓ should clean many update operator (189ms)
    ✓ reverts when updateOperator try to set many update operator (59ms)
    ✓ reverts if not owner want to update the update operator (55ms)
  LandBalance
    setBalanceToken
      ✓ should set balance token (40ms)
      ✓ reverts if a hacker try to set balance token
    Register balance
      ✓ should register balance (101ms)
      ✓ should re-register balance (425ms)
      ✓ should unregister balance (343ms)
    Update balance
      ✓ should register balance only one balance (574ms)
      ✓ should register owner balance if it was transferred by approval for all or operator (405ms)
      ✓ should register balance both (344ms)
      ✓ should not register transfer to the estate registry (738ms)
      ✓ should register land balance and estate balance (885ms)
      ✓ should update on transfer :: transferFrom (356ms)
      ✓ should update on transfer :: safeTransferFrom (236ms)
      ✓ should update on transfer :: transferLand (225ms)
      ✓ should update on transfer :: transferManyLand (312ms)
      ✓ should update on transfer :: transferLandToEstate (867ms)
      ✓ should update on mint :: assignNewParcel (103ms)
      ✓ should update on mint :: assignMultipleParcels (122ms)


  281 passing (8m)
```

Code Coverage

We were unable to run standard coverage tools such as `solidity-coverage` on the repository. This appears partially related to solidity files existing in the `test/` directory, rather than nested under `contracts/` such as in `contracts/test`. We recommend adding necessary dependencies to the `package.json`, and documenting steps to produce coverage results in the `README.md`.

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

```
d362701bf0a5a3233ef4b2030fc95ba4df1d6e7baae7eba3736d6dd827b58434 ./contracts/Storage.sol
749b8d6b4c2ccf8140b4e56c9d5e998091cd92747ccb0ef3052b69b43dd8558f ./contracts/metadata/IMetadataHolder.sol
431540245834d0a66287e9faa3aa27984bc0470bbfedde49dd6b3089ecb15b3f ./contracts/utils/Migrations.sol
459f2a66a44dbb176fbdd9f09e1f06e72681dcdd87d43119dda52d3917175853 ./contracts/minimeToken/MinimeToken.sol
c6d434b16d0dd0fbd3abc0e5b2710040adb79a0b6cad2d54ded60285c5de4249 ./contracts/minimeToken/IMinimeToken.sol
c8f9d1d47c84e9a10c04ea60aff4aa9f010a4d00083f859ff07decc81bdf1fa4 ./contracts/minimeToken/ITokenController.sol
d1b36a163fa249a7bb2be6f79e945cc595471f695b8e2b31c0c2fa1fd7192356 ./contracts/upgradable/LANDProxy.sol
3651edb09b164f4c34ca50151d2cad0e4872d209104e6ae6f9c35e165d4b7eeb ./contracts/upgradable/ProxyStorage.sol
444802b84a500db3c4f59ae236b664c41cedef89d9bd4b6ffbbafd522658c725 ./contracts/upgradable/IApplication.sol
63578420e0403b518ae2774270742f3ccd11d38b65436afab45c2a1eba0ba545 ./contracts/upgradable/Ownable.sol
1ebf067c245909cea4f63fa7cc36fe7f31d5242dfa32f81398b41beee3659688 ./contracts/upgradable/Proxy.sol
8bdfcdb86b3ac0f1ec18cd4d9b6a66b317d8442400fe30fd7b6cd8dd006dc077 ./contracts/upgradable/DelegateProxy.sol
5188b9d857dc24e4789b90481aa2715b7ce6da9707ad9da68fb4cda99854841f ./contracts/upgradable/OwnableStorage.sol
e50b5d3f2d538cb9b2924dd8253d3325922e337f77de273daab76a338581aca2 ./contracts/estate/EstateRegistryTest.sol
5fcae9014b067b13a938032336d50df6fde62a4815ebc22d5eb5481014c8504c ./contracts/estate/EstateRegistry.sol
6ef45c483accc350f8066d989f2b14ac1a4f1b54085de963ef61c983a42a21cf ./contracts/estate/IEstateRegistry.sol
a3703d84ad2cb109547d47c219cebe9c7b58e8ed1ccb352e25b22ccc2d4d76d2 ./contracts/estate/EstateStorage.sol
278edaa619326a6c92c61946210f1c3fc41474e9c25f81d4700be2dae7f01591 ./contracts/land/LANDRegistry.sol
b186102f31214ffe0380640678a3a2663cf7d63bf63b84e1569e708f7fa65219 ./contracts/land/LANDStorage.sol
13bdeb1af8605f6cc23456486aa8a366cbe1248e815c98865bc9768de96562cb ./contracts/land/ILANDRegistry.sol
1df74b14b6fe873ae570d8fdd53583b99f906536ef103d1bfacf92d7a0eb3cd5 ./contracts/land/LANDRegistryTest.sol
```

Tests

```
0b2256d137d4f3c5ab16291e32750523c2ece2ce1601a0a881e114cd72c04c10 ./test/EstateRegistryTest.sol
fae0987ad81b8f03c9a17073a12f2b6fb3a3b39df08da85863ef973bc2661dae ./test/LANDRegistryTest.sol
9390bc14e83c672dc51dbd3f1fe16a251ba5debf7140c04868568b7020be2a6b ./test/EstateRegistry.js
26cf7a06695fe399ba02542d555fd90d830d4af02ef7a0c0ef6a464f39d2c17e ./test/LANDRegistry.js
336b51ba68e66e576404c8bbb4c4543cad59cd6c4520ead6b669b278a7c8893e ./test/LANDProxy.js
cbc143651f68b4f8293363c019b5b83d95d4981b26c3619f39e0b70a44e36513 ./test/utils.js
2ee8e3690f17998a0fe5ede97d6b13f7e7a845755dec14a9f413fd2777e9d181 ./test/Integration.js
03170357d5fcd02e544851c3639f278237fd7b2e877f686c60b39967b5ea6cd0 ./test/helpers/increaseTime.js
bcb031eb21953320b22c38a9415837fdd2d7ede16893dfee373eba5fd1782b87 ./test/helpers/getSoliditySha3.js
6dcdda6df60f965bb7c56d4b73854ebd3e7f24c586522810cee1db6dc53dfa78 ./test/helpers/assertRevert.js
d349214203a7828e099e178b0243add881788fc523bc03f3dbda246589d4c38b ./test/helpers/setupContracts.js
bececbdafc2b7139f9b98f0a0d485c71e37afec469cb02e0b12667857ddd14cd ./test/helpers/createEstateFull.js
```

About Quantstamp

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

Quantstamp's team boasts decades of combined experience in formal verification, static analysis, and software verification. Collectively, our individuals have over 500 Google scholar citations and numerous published papers. In its mission to proliferate development and adoption of blockchain applications, Quantstamp is also developing a new protocol for smart contract verification to help smart contract developers and projects worldwide to perform cost-effective smart contract security audits.

To date, Quantstamp has helped to secure hundreds of millions of dollars of transaction value in smart contracts and has assisted dozens of blockchain projects globally with its white glove security auditing 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.

Finally, Quantstamp's dedication to research and development in the form of collaborations with leading academic institutions such as National University of Singapore and MIT (Massachusetts Institute of Technology) reflects Quantstamp's commitment to enable world-class smart contract innovation.