Audit of Decentraland Estates Feature

v1.0.0

August 22nd, 2018

LEVELK

Introduction

The Decentraland team asked us to review their estates feature contracts. The commit used for the initial version of this audit is 5b51cb4.

We reviewed the code and laid out the results below. Overall, the code is very well-written, clean, easy to understand, and generally follows best practices regarding structure and security. There were no critical issues found in the code. We recommended a few changes all of which are outlined below.

Some particularly strong patterns in the code include: (a) usage of Solium with a well-configured `.soliumrc.json`, (b) usage of OpenZeppelin's `SafeMath`, and (c) the new `setupContracts` test helper.

Issues

1. EstateRegistry implements an outdated version of the onERC721Received function

Medium

EstateRegistry.sol#L77 - The Estate registry implements `function onERC721Received(address _from, uint256 _tokenId, bytes _data) external returns(bytes4);` which is from the outdated Decentraland erc721 standard. The current standard is `function onERC721Received(address _operator, address _from, uint256 _tokenId, bytes _data) external returns(bytes4);` which also has a different return value. Consider updating the land registry's use of `onERC721Received()` to the latest standard before deploying `EstateRegistry` and update this function in the `EstateRegistry` contract to reflect the current standard.

2. _bytesToUint function only works for bytes arrays with a length of 32

Medium

EstateRegistry.sol#L359 - In `EstateRegistry`, `_bytesToUint()` only works with bytes arrays with a length of 32. This function is currently only being used with bytes arrays of length 32 so there is no immediate security risk. Consider adding a check like `require(b.length == 32);` to ensure bytes arrays of different lengths are not accepted and this function is not misused in the future.

3. MetadataHolderBase, which EstateRegistry inherits from, implements supportsInterface() overriding the implementation in ERC721Token

Medium

MetadataHolderBase.sol#L8 - `EstateRegistry` inherits from both `MetadataHolderBase` and `ERC721Token` both of which implement `supportsInterface(bytes4)`. Because of the order of inheritance, `MetadataHolderBase`'s implementation overrides `ERC721Token`'s implementation causing the function to return `false` for the `ERC721Enumerable` (0x780e9d63) and `ERC721Metadata` (0x5b5e139f) interfaces when it should return `true`. Consider using Zeppelin's `SupportsInterfaceWithLookup` contract by having `MetadataHolderBase` inherit from `SupportsInterfaceWithLookup` like `ERC721Token` does. Then, in the `MetadataHolderBase` register the `getMetadata(uint256)` interface. There is no need to register the `supportsInterface(bytes4)` interface as it is already registered by `ERC721Token`.

4. Test coverage

Medium

a) **One test for `land.transferManyLandToEstate` is testing the wrong thing**
`'does not transfer lands if it is called by not authorized operator'` in `transferLandToEstate` `describe`. In this test, `land.transferManyLandToEstate` should be transferring to an `estateId`, not the user `creator`.

b) **The current usage of test `.sol` files presents some risks in the future**
It's worth trying to find another approach besides changing the visibility of functions for testing purposes via a special Test `.sol` file. Sometimes you'll want to write tests that guarantee the right visibility for the non-test version of those functions. We suggest only using the test contract in tests where the changed functions are needed so that visibility can still be checked with the original contract.

c) **More tests for calling functions with unexpected input data**
The current "unhappy path" tests mostly involve unauthorized users. Look at calling functions with unexpected values, like putting land instead of an estate, an estate instead of land, an estate instead of an account, etc.

d) **Assert on details of a test's outcome beyond just the revert status**
Some tests test only that there's a revert or no revert. We suggest adding an additional, more specific assertion on the result like that the recipient now has the tokens. For example, `'operator can transfer many tokens out'`. For tests that are supposed to revert, test that the original sender has them back.

e) **No integration tests for estate features**
`test/Integration.js` had no significant modifications for the estate feature PRs. If integration tests will end up being in other `*.js` test files going forward, it's worth considering removing the integration test file altogether.

f) **Transfering many lands and a later one reverts**
Add a test case for what happens if you're trying to transfer many lands to an estate and the 3rd one reverts. Include detailed asserts on where all the units of land ends up.

g) **Transferring land out of an estate**
Add a test to make sure the "require" of non-zero index is working. Consider calling the new test: `"You can't transfer land from an estate that the estate doesn't have."` This is important to test because the indexing starting at 1 isn't the most intuitive, and in a future commit this property could be lost accidentally.

h) **More `'random user can not transfer tokens out'` tests**
Other tests similar to this test would be useful, like sending to the `anotherUser` account rather than `hacker` but still `sentByAnotherUser`. This is not strictly required, but these are the class of tests for which it's good to be exhaustive.

5. Arithmetic operations in the _transferLand function

Low

EstateRegistry.sol#L299 - In `EstateRegistry`'s `_transferLand()` function there are multiple arithmetic operations. It is possible for some of the arithmetic operations to cause underflows and while an underflow doesn't present a vulnerability in this case, it can lead to unexpected results in the future. Consider using SafeMath instead.

6. EstateRegistry's ping() function is onlyOwner

Low

EstateRegistry.sol#L147 - EstateRegistry's ping() function is onlyOwner but doesn't need to be restricted. While reclaiming land from inactive addresses is not being enforced now, once that is implemented all land in estates could be lost if decentraland fails to ping the estate registry or loses the owner private key. Additionally, land could be left in estates by an inactive address for long periods of time or indefinitely. Consider removing onlyOwner from this function and consider tracking the latest ping for estates in a future update.

7. Pin Open Zeppelin version

Low

package.json#L23 - "OpenZeppelin does not currently follow semantic versioning. You may encounter breaking changes upon a minor version bump. We recommend pinning the version of OpenZeppelin you use, as done by the -E (--save-exact) option." Source

Notes

- `require(exists(estateId), "The estate id should exist");` is a duplicate check and can be removed.
- `function mint(address to)` is never used and can be removed.
- Style
    - **Do any "require"s at the lowest level function possible**
        - The motivation here is to stay DRY and to have a consistent place to look for them in future audits. The codebase mostly sticks to this already, but we'd propose establishing a convention of always doing any "require"s at the lowest level function possible to stay DRY. If a check is for a parameter that is not passed to the lower level function, then of course it belongs in the calling function instead. Mixing the two approaches makes it easy to forget to do it.
    - **Add missing natspec comments**
        - Using Zeppelin's Solium plugin with the `missing-natspec-comments` rule will point out any missing natspec comments.
    - **Some contracts import two sol files where one of those sol files imports the other, so it's not needed.**
        - LandProxy imports Proxy and is Proxy. Consider having LandProxy import Storage and inherit from it, because Storage already is inherited from in Proxy. Same for LandRegistry - it already is Storage via Ownable. Same for Proxy - it already is Storage via Ownable
    - **Consider using a develop branch so that merges to master correspond with deploys.**
- Typos and grammar
    - **token ID should say estate ID**
        - safeTransferManyFrom param comment should say estate ID array rather than token ID
    - **Use "an" rather than "a" where appropriate**
        - In this file it comes up a lot, any time the next word starts with a vowel, like "estate".
    - **Change "destinatary" to a more common word like "recipient"**
        - Destinatary isn't commonly used in English.
    - **Typo: Interal -> Internal**
        - @dev Interal function to mint a new Estate with some metadata
    - **Typo: cant -> can't**
        - cant is used here
    - **transferManyLandToEstate -> transferManyLandsToEstate OR vice versa**
        - In EstateRegistry.sol, and LANDRegistry.sol there is inconsistent pluralization of lands when many are being transferred
    - **One "require" has the opposite message of what it should**
        - require(landIndex[landId] != 0, "The land is already owned by the estate");

- Error message should say what the situation is if the require fails, so “The land is not already owned by the estate”

Conclusion

No critical issues were found. The code is very well written and easy to follow. We made a few suggestions to fix some minor issues and avoid potential issues in the future.