Security Audit
Decentraland
ERC721Wearables contracts

Monday, 04-May-2020

Agustín Aguilar

Introduction

The Decentraland team requested a security audit of their ERC721 wearables project; the audited repositories follows:

https://github.com/decentraland/wearables-contracts/tree/3b9d7deae06039806b297bc63b27f6ad20797654

The contracts that constitute the project follow.

- ERC721BaseCollection.sol: Implements the base wearables ERC721, each wearable has an unique id and a maximum supply, an owner can add new wearable types but can't modify the total supply.

- ERC721Collection.sol: Implements a wearable collection using ERC721BaseCollection, issuance ids auto-increment until the maximum supply has been reached. Allowed addresses can mint the tokens, the owner determines those allowed addresses.

- ERC721DeterministicCollection.sol: Implements a wearable collection using ERC721BaseCollection, tokens can be minted out-of-order. Allowed addresses can mint the tokens, the owner determines those allowed addresses.

- libs/String.sol: Helper library used to generate and manipulate strings.

Issues

Low severity

L1 - Improper check of issued and max issuance of tokens

The ERC721BaseCollection contract implements a fixed upper bound for each minted wearable. Nevertheless, the enforcement of these limits relies on the contract implementing the base collection.

The mint method requires `_issuedId` to be under the defined maximum issuance (line 208). However, this parameter is controlled by the contract implementing the collection, and it could lead to a total issued tokens above the maximum defined if misused.

Proposed solution:

- Consider explicitly validating that `issued[_wearableId]` is below of `maxIssuance[_wearableIdKey]`.

Notes

N1 - Unused roles on base collection

The ERC721BaseCollection contract defines the `allowed` role and according to his documentation this role allows an address to issue tokens.

This feature is not implemented on the base collection contract and relies on the contract implementing the collection to properly implement the functionality.

Proposed solution:

- Consider moving the `allowed` logic to a standalone contract.
- Add more documentation on the ERC721BaseCollection contract, specify that `onlyAllowed` is not implemented and how it should be implemented.

N2 - Redundant validation of asset existence

The ERC721Collection contract implements the internal method `_setTokenURI` as a mechanism to assign a URI suffix to each minted token.

This method validates that each token exists before storing the suffix. However, `_setTokenURI` is only called after calling `_mint`, making the required condition always `true`.

Additionally, the `tokenURI` getter also validates that the given `_tokenId` exists.

Proposed solution:

- Remove `require(_exists(_tokenId))` from `_setTokenURI`.

N3 - Inefficient usage of completed and only owner

The ERC721BaseCollection contract implements the `onlyOwner` modifier and `!isComplete` on the `addWearable` method; this method is used internally by the `addWearables` function.

This pattern leads to `onlyOwner` and `!isCompleted` being applied N + 1 times for all `addWearables` batch operations.

Proposed solution:

- Consider implementing the `addWearable` logic on a private method, and using `onlyOwner` and `!isComplete` only on external functions.

N4 - Inefficient ID key pre-image

The ERC721BaseCollection contract computes the key of a wearable as `keccak256(abi.encodePacked(_wearableId))`, the pre-image requires the `_wearableId` to be packed before hashing.

It has to be noted that `abi.encodePacked` adds an additional cost, consider replacing it with `bytes(_wearableId)` or `abi.encode(_wearableId)`.

N5 - Redundant condition on issue deterministic token

The `_issueToken` method on the ERC721DeterministicCollection contract enforces that `_optionId >= 0`, this condition is always true given that `_optionId` is an unsigned integer and can't be a value below zero.

Final thoughts

The wearable contracts are well written and have substantial test coverage; no issues have been found on the project. Additional documentation on how to use ERC721BaseCollection could improve future maintainability.

- May 2020 - Agustín Aguilar