Security Audit
Decentraland
Collections v2

Monday, 26-Abr-2021

Agustín Aguilar

Introduction

The Decentraland team requested a security audit of their collections v2 set of contracts; the audited files can be found in the following repositories:

https://github.com/decentraland/wearables-contracts/tree/aa02e9c1f492c6d8ad8edeba9d4798750b04e927

https://github.com/decentraland/marketplace-contracts/tree/d27b695e8006e70a5b7f245a57a13a5ec97f7ccf

The contracts that constitute the project follows.

- ERC721BaseCollectionV2: Implementation contract for ERC721 based collection, includes multiple roles for creating, minting, updating, and managing the collection.
- ERC721CollectionV2: Inherits `ERC721BaseCollectionV2` without any modifications.
- ContextMixin: Provides an alternative `msg.sender` internal method to retrieve the sender of a meta-transaction.
- EIP721Base: Implements base functionality to use EIP721 domain separators and typed message hashes.
- Forwarder: Minimal 1/2 multisignature wallet.
- MetaTxForwarder: Wallet without authentication.
- MinimalProxyFactory: Bytecode level proxy contract factory.
- NativeMetaTransaction: Generic meta-transaction implementation, intended to be used alongside `ContextMixin`.
- OwnableInitializable: Generic Owner base contract, with `ContextMixin` and initializable support.
- ERC721CollectionFactoryV2: `ERC721BaseCollectionV2` creator using `MinimalProxyFactory`.
- String: Library for string manipulation.
- CollectionManager: Contract that manages and collects fees for the creation of collections.
- Committee: 1/N multisignature wallet, can only call the `manageCollection` method.
- Rarities: Item rarities repository.
- CollectionStore: ERC721 issuer that exchanges collection tokens for ERC20 tokens.
- Marketplace: Marketplace for ERC721 tokens.

Issues

Medium severity

M1 - Arbitrary external calls may conflict with Native meta transactions

The project makes widespread usage of a common pattern called `NativeMetaTransaction`; this pattern allows any method to be called by using meta-transactions without further modifications or the usage of smart contract wallets.

To achieve this functionality, all access to `msg.sender` is replaced by a call to an internal method provided by the `ContextMixin` contract; this method validates if the `msg.sender` is `address(this)`, and in that case returns the last 20 bytes of the `msg.data` as the sender instead of the provided value by the EVM.

This behavior relies on the assumption that the contract processing the meta-transaction can't call itself outside of the context of executing a meta-transaction; otherwise it could open a vector to spoofing the `msg.sender` on other parts of the contract.

Proposed solutions:

1) Forbid the contracts from calling to `address(this)` on any methods that allow external calls.
2) Toggle a flag when the contract executes a meta-transaction and only return the alternative `msg.sender` on `ContextMixin` if the caller is `address(this)` and the flag is toggled.

Update: The issue has been partially addressed using proposed solution (1) on the commit `0c20f1172dd6b29562d05fab9842bea3eadbec62`, however the ERC721 contract is still able to perform arbitrary calls to `address(this)`.

M2 - Front-running could denegate the creation of new collections

The `ERC721CollectionFactoryV2` contract makes use of the internal method `_createProxy` of the `MinimalProxyFactory` common contract, this method deploys a proxy contract using the `CREATE2` opcode, takes a salt to determine the final address of the proxy contract, and it optionally calls a method on the deployed contract (used for contract initialization).

`MinimalProxyFactory` doesn't enforce the initialization call to be part of the `CREATE2` salt. Neither does the `ERC721CollectionFactoryV2` contract which receives a salt as a parameter, and then passes it directly to the `_createProxy` method.

This behavior results in the address of the collection being fully independent of A) the creator of the collection and B) the details of the collection, leading to a possible service denegation attack vector:

- User A tries to create a collection with salt 0x02.
- Attacker sees User A transaction on the mempool.
- Attacker constructs the creation of an empty collection with the salt (0x02).
- User A transaction fails because a contract already exists on the address of 0x02.

Proposed solution:

1) Include the initialization data on the `CREATE2` salt.

Update: The issue has been fully addressed using proposed solution (1) on the commit `d819af9058bb199b6ae09c6cdb9c85053b6e6fa9`.

Low severity

L1 - Cache of chainId may break ERC712 domain separator

The `ERC712Base` contract pre-computes the `domainSeparator` of EIP712 during the initialization; the value is then cached and used every time the domainSeparator is used in scenarios like the native meta-transactions.

This `domainSeparator` depends on the value returned by the `CHAIND 0x46` opcode; this value shouldn't be treated as a constant, given that an event like a network upgrade or fork could cause it to change.

Such scenarios could lead to unexpected behavior, like meta-transactions not being replay-protected after the fork or not working at all.

Proposed solution:

1) Compute `domainSeparator` at runtime, without caching the chainId factor.

Update: The Decentraland team has acknowledged the issue; however, they consider the scenario too unlikely to justify changes on the contract.

L2 - Anyone can claim ownership of the collection template

The `ERC721CollectionV2` contract acts as the implementation of all collection proxy contracts, it contains the bytecode that will be used to execute those collections during the whole lifecycle of the proxies.

Proxies can't use constructors; for this reason, the collection contract implements the initialize method; this method allows anyone to set up the collection with any parameters and can only be called once.

Because the reference `ERC721CollectionV2` also implements this method, it can also be initialized by anyone unless it's initialized before reaching a production state.

Proposed solutions:

1) Define `isInitialized` as true on the `ERC721CollectionV2` constructor; this will only be executed at the creation of the base contract but not on the proxies.
2) Make sure the template is properly initialized before starting using the contracts in a production environment.

Update: The Decentraland team addressed the issue by committing to solution (2), and adding a helper method that initializes the collection without having to provide all extra parameters, in commit `215d1885d67639cb6b50d5d5d58268d808caa4d2`.

Notes

N1 - Committee can't recover strained funds

The Committee contract serves as a feature-limited multi-signature wallet with `1/N` threshold; this wallet is used internally to manage the ownership of some contracts.

Consider replacing this wallet with a proper multi-signature wallet. This change would allow for greater flexibility. It would also enable the wallet to fulfill other roles in case it would become necessary, for example in the event of having to retrieve lost tokens sent to the committee contract.

N2 - Native meta transactions don't support smart contract wallets

The `NativeMetaTransaction` common contract provides a mechanism for EOAs to sign a typed message to be later interpreted on the contract as a meta-transaction.

This functionality is only available for EOAs, consider the possibility of some users interacting with the system using smart contract wallets, and implement signature validation using EIP1271 to add support for such wallets.

N3 - Collection store may disrupt ongoing transactions during price changes

The buy method of the `CollectionStore` contract takes a price parameter per item to buy; the contract validates if the price matches the one defined by the collection, and if it does, it allows the trade to happen.

If the collection price is reduced while buyers are sending buy transactions, a certain amount of buy transactions may fail due to price-mismatch, even when the new price is lower, and thus all users would accept the new price.

Consider replacing the price parameter with a `maxPrice` one. If the price changes but below the maximum price that the buyer is willing to pay, the transaction still confirms correctly.

N4 - Strict redundant requirements may difficult management with time-delay

Similar to note N3, multiple methods on the Collection contract revert the whole operation if one of the values on a batch of changes is a no-op, meaning it changes the value to the same value the property already has.

This seemingly innocuous property may cause a problem when multi-signature wallets and time-lock wallets try to interact with the collections; a valid transaction at the time of its construction may become invalid at execution time, causing delays in operational overhead.

Consider not reverting operations unless it's strictly required; if a batch of operations contains a no-op, it should be considered safe to ignore such operations.

N5 - Issuing depletes allowance even when the minter is global

The `ERC721Collection` instance contracts allow for issuing tokens if any of the following conditions is true:

1. The sender is the creator of the collection
2. The sender is currently defined as a globalMinter of the collection
3. The sender has enough allowance to mint the itemId on that collection

When the sender uses the mechanism (3) it depletes the allowance for mint that given token; however, if the sender is also a `globalMinter` or the creator of the collection, it also depletes the allowance until it reaches zero.

Update: The Decentraland team addressed the issue in commit `53094ae995341f4c994862a34abc9b9130bd63d2`.

N6 - Native meta transactions use incorrect naming for data

The `NativeMetaTransaction` contract defines an `executeMetaTransaction` method that takes a parameter named `functionSignature`; this parameter takes the form of the transaction data of the meta-transaction.

Consider renaming the parameter to `data`, `metaTransactionData` or `transactionData`; given that `functionSignature` is usually reserved for the first 4 bytes of the data that corresponds to the selector of solidity functions.

N7 - Fee collector can be bypassed by creating empty collections

The `CollectionManager` contract requires a fee that has to be paid for each created collection, must be paid by the creator of the collection, and it's determined by the cost of the rarities of each item on the collection.

Created collections also have the means to add items after creation using the `addItems` method; this method is callable by the creator of the collection. Still, it doesn't require any fee to be paid to the `feeCollector`.

At present form of the contracts, this is not an issue given that `CollectionManager` only allows for the creation of completed collections. Such collections cannot include additional items after creation; however, this code path can easily be missed in a future change of the contract set.

Update: The Decentraland team addressed the issue by only allowing the owner to add new items to the collection, in commit `0583623692834d6ab534a97b4cc94c1b7c4622cf`

N8 - Minimal proxy factory makes inefficient usage of storage

The `MinimalProxyFactory` common contract implements a variation of the clone proxy pattern; this pattern requires sandwiching the implementation contract between two pieces of bytecode, resulting in a deploy code that results in a simple proxy contract, pointing to the given implementation.

The factory implements this behavior by computing the full code and its codeHash during initialization and storing those values on contract storage. Subsequent proxy creation or getAddress calls load the pre-computed values from storage.

This pattern is inefficient, given that computing the bytecode of the proxy contract is orders of magnitude cheaper than accessing multiple values from contract storage.

Consider only storing implementation on contract storage and instead of having an internal method that constructs the bytecode from a given implementation.

Final thoughts

The contracts composing the audited projects are well written, no critical or high vulnerabilities have been found.

The M2 vulnerability must be addressed before deploying the project in a production environment, otherwise the project faces severe denegation of service risk, and being forced to deploy a new set of contracts.

- April 2021 - Agustín Aguilar