

# **Rentals Contract Audit Report**

Date: 10/25/2022

Auditor: Xinghui Chen

Email: [ChenXinghui@protonmail.com](mailto:ChenXinghui@protonmail.com)

# Summary

---

## Scope

---

Common Contracts

<https://github.com/decentraland/common-contracts/tree/93a3471df9e53b11bf2f36fc8139f8ae8f99b34e>

Rentals

<https://github.com/decentraland/rentals-contract/tree/bfe7522314ae24d27aa927d74dee4718b90e55a7>

Every contract inside /contracts except for the ones in the /mocks folder.

## Findings

---

	Critical	High	Middle	Low	Gas	Recommendation	Misc
NativeMetaTransaction.sol	-	-	-	-	1	-	-
AssetNonceVerifiable.sol	-	-	-	-	-	-	-
ContractNonceVerifiable.sol	-	-	-	-	-	-	-
SignerNonceVerifiable.sol	-	-	-	-	1	-	-
IERC721Rentable.sol	-	-	-	-	-	-	1
Rentals.sol	-	-	-	-	5	5	1

## Audit Revisions

---

Common Contracts

<https://github.com/decentraland/common-contracts/tree/1c85438e913fe5affbef8e480c467585738e694a>

Rentals

<https://github.com/decentraland/rentals-contract/tree/51a2cd2951920c4573dc0368844a19279f00eba1>

# Details

---

## NativeMetaTransaction.sol

---

1. In function **executeMetaTransaction**, param `_functionData` and `_signature` are declared as memory, it's better using calldata as it's more gas friendly( `_signature` param in `_verify` function can be declared as calldata too).

**Result:** Changed.

## SignerNonceVerifiable.sol

---

1. As `_bumpSignerNonce` is only called by **bumpSignerNonce**, it's better removing `_bumpSignerNonce` and move its logic into `bumpSignerNonce`.

**Result:** Not change. Reason:

Contracts from Common Contracts are intended to provide utilities to many different contracts. “\_bumpSignerNonce” might not be used by Rentals, but it might be used by other new contracts internally. So it should remain separated from the external call.

## IERC721Rentable.sol

---

1. Line 17 and 22, `memory` can be changed to `calldata` as these params are actually `calldata` in `EstateRegistry.sol`.

**Result:** Not change. Reason:

These functions are declared as public in `EstateRegistry.sol`, so `memory` is right. I mixed up as they are declared as external here, sorry.

## Rentals.sol

---

### Comment mistake

1. Line 202, **Get if and asset..** should be **Get if an asset...**

**Result:** Changed.

## Gas optimization

1. In function **acceptListing**, param `_listing` is declared as memory, it's better using calldata as it's more gas friendly(`_listing` param in `_verifyListingsSigner` function can be declared as calldata too). Same for other external functions:

- function **acceptOffer** , param `_offer` .
- function **claim**, all params.
- function **setUpdateOperator**, all params.
- function **setManyLandUpdateOperator**, param `_landTokenIds` and `_operators` .
- function **onERC721Received**, param `_data` .

**Result:** Changed.

2. As the existence of line 275, line 270 and 271 can be removed.

**Result:** Not change. Reason:

The require can be removed, but will leave it there as it expresses intention better.

3. line 402 use `.length` in the loop condition, it's better saving it first. e.g.,

```
uint256 tokenIdLength = _landTokenIds.length;
for (uint256 i; i < tokenIdLength; ++i) {...}
or
for (uint256 i; i < tokenIdLength; ) {
    ..
    unchecked {
        ++i;
    }
}
```

Same for line 314, 359.

**Result:** Changed.

4. line 321 `Rental memory rental =` can be changed to `Rental storage rental =` as the logic below only access its one field one times. As Rental struct has 3 members, if use memory here, will cost more than 3 SLOAD(cold), if use storage, will only cost 1 SLOAD(cold). Same for line 362, 393.

**Result:** Changed.

## Out of block gas limit

1. Line 553 calls `verifyFingerprint` in `EstateRegistry.sol`, which may cause `out of gas` if the estate contains too many lands. In my test(use line 1302 in `Rentals.spec.ts`), 3000 lands cost around 9\_000\_000 gas( $x \in [-8,7]$ ,  $y \in [-100,100]$ ), 10000 lands will fail( $x \in [-25,25]$ ,  $y \in [-100,100]$ ). I think it's ok as few people will trigger it.

**Result:** Known issue.

## Recommendation

1. Line 428 use `msg.sender` directly. As the contract supports meta-transactions, it's better using `_msgSender()` everywhere in case of unintended behavior.

**Result:** Not change. Reason:

The "onERC721Received" method has not use through meta transactions as it should be called through ERC721 safeTransfers. Using `msg.sender` instead consumes less gas as we avoiding the "if (msg.sender == address(this)) {" of the `_msgSender`.

I still recommend use `_msgSender()` everywhere if a contract supports meta-transactions as `msg.sender` could be the contract itself. Using `msg.sender` will introduce an attack vector, and need pay attention if the contract upgrades in the future.

2. Line 7 can be removed as `Rentals` doesn't inherit `OwnableUpgradeable` directly.

**Result:** Changed.

3. Term `asset` is arbitrary: It means a nft in the comment, means nft contract in the code(line 330, 549).

**Result:** Not change.

4. `pragma solidity ^0.8.7` could be changed to `pragma solidity 0.8.7` (all files), for more detail please visit [here](#).

**Result:** Changed.

5. In function `_rent`, when `isExtend` is true or `isReRent` is true, should not try to transfer asset any more(line 600-602). Instead, shall assert that current contract must already own the asset. So maybe can change as below:

```
if (isExtend || isReRent) {
    require(_ownerOf(_rentParams.contractAddress, _rentParams.tokenId) ==
address(this), "Not own(accept[Listing|Offer])");
} else {
    // verify that the asset is not already rented. (line 566)
    require(!isRented, "Rentals#_rent: CURRENTLY_RENTED");
    if (_msgSender() == _rentParams.contractAddress) {
        require(_ownerOf(_rentParams.contractAddress, _rentParams.tokenId)
== address(this), "Not own(onERC721Received)");
    } else {
        asset.safeTransferFrom(_rentParams.lessor, address(this),
_rentParams.tokenId);
    }
}
```

Then remove line 600-602 and function `_verifyUnsafeTransfer` (not need deal unsafe transfer situation specially any more).

**Result:** Not change. Reason:

Making the change provided decreases the gas consumption of `acceptListing` and `acceptOffer` by about ~5k gas (however, it increases the `onERC721Received` consumption by ~4k). Saving that gas would be nice, but the change involves the core part of the contract and we are not willing to change it.

## PS

---

1. The contract depends on `openZeppelin 4.5.0`, please be aware that it has some [issues](#), all these won't affect current contract currently.

**Result:** No impact.