OpenZeppelin | security

Catalyst Audit

September 13, 2023

Table of Contents

Summary

Type	NFT
Timeline	From 2023-08-04 To 2023-08-18
Languages	Solidity

Total Issues	8 (8 resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	4 (4 resolved)
Notes & Additional Information	4 (4 resolved)
Client Reported Issues	0 (0 resolved)

Scope

We audited the thesandboxgame/sandbox-smart-contracts repository at the 5789d5a3f7dc4dc5c32293cd7615f97d395c93ae commit.

In scope were the following contracts:

```
packages/
└─ asset/
   └─ contracts/
       └─ Catalyst.sol
       └─ interfaces/
         └─ ICatalyst.sol
```

System Overview

This system introduces Catalyst which is an ERC-1155 token. It adopts royalty distribution to users based on the ERC-2981 standard and complies with OpenSea's operator filter registry across all chains to stay updated with prevailing standards. Catalysts have specific tiers that allow the minter role to distribute diverse tokens to users. Administrators have the flexibility to augment these tiers, increasing the range of token-minting possibilities in the future.

Catalyst

The Catalyst contract is used to manage distinctive ERC-1155 tokens, also called catalysts, and pay the royalty through the `RoyaltyDistributor` contract. These catalyst tokens can be minted individually or in a batch by the `MINTER_ROLE` privileged role. Similarly, they can be burnt, individually or in batches by the users or by the `BURNER_ROLE` privileged role.

The audited codebase defines 7 types of catalysts: TSB_EXCLUSIVE, COMMON, UNCOMMON, RARE, EPIC, LEGENDARY, and MYTHIC. Administrators have the flexibility to add new catalyst types, thereby increasing the range of token-minting possibilities in the future.

Privileged Roles and Trust Assumptions

The audited contracts contain the following privileged roles:

- The `admin` of the `Catalyst` contract can perform the following actions:
  - Add new catalyst types
  - Change the address of the `trustedForwarder`
  - Assign a new URI for a valid `tokenId`
  - Alter the base URI of the contract

- The `MINTER_ROLE` defined in the `Catalyst` contract can mint individual or batches of ERC-1155 catalyst tokens.
- The `BURNER_ROLE` defined in the `Catalyst` contract can burn individual or batches of ERC-1155 catalyst tokens from a specific address.

During the audit, it is assumed that the privileged addresses are trusted entities and would work in the best interest of the platform and its users.

Low Severity

L-01 Incorrect Documentation

Several docstrings and inline comments throughout the codebase were found to be erroneous and should be fixed. In particular:

- The docstring above the `addNewCatalystType` function states an incorrect definition of the `ipfsCID` variable.
- The docstring above the `setOperatorRegistry` function incorrectly states that the function "sets filter registry address deployed in test". The function will also be used on the mainnets.

Update: *Resolved in pull request #1111. The Sandbox team stated:*

> *Documentation has been updated*

L-02 Lack of Event Emission After Sensitive Action

The following functions do not emit relevant events after executing sensitive actions:

- When the `_baseUri` is set in the `initialize` function in the `Catalyst` contract.
- When the `_baseUri` is set in the `setBaseURI` function in the `Catalyst` contract.
- When the `operatorFilterRegistry` is set in the `setOperatorRegistry` function in the `Catalyst` contract.

Consider emitting events after sensitive changes occur to facilitate tracking and notify off-chain clients following the contracts' activity.

Update: *Resolved in pull request #1113. The Sandbox team stated:*

> *Added all suggested events*

L-03 Lack of `gap` Variables

Throughout the codebase, there are multiple upgradeable contracts that do not have a gap variable. For instance:

- The `Catalyst` contract

Consider adding a gap variable to avoid future storage clashes in upgradeable contracts.

Update: *Resolved in pull request #1114. The Sandbox team stated:*

> *Added gap variables.*

L-04 The `Catalyst` Contract Allows the Burning and Transfer of Non-Existent Tokens

A user can burn their own tokens or tokens that they are approved to use by calling the `burn` or `burnBatch` functions respectively. Moreover, the `burnFrom` and `burnBatchFrom` functions in the `Catalyst` contract allow the privileged addresses with the `BURNER_ROLE` to burn tokens from any account.

The `safeTransferFrom` and `safeBatchTransferFrom` functions can be called by whitelisted operators for transferring tokens to the users.

The `Catalyst` contract allows the minting or transferring of tokens that have a `tokenId` between 1 and `highestTierIndex`. However, the valid `tokenId`s are not checked during the burning or transfer of tokens, thereby allowing users to transfer or burn arbitrary ERC-1155 tokens.

Consider checking for valid `tokenID`s before burning tokens.

Update: *Resolved in pull request #1117. This is not an issue. The Sandbox team stated:*

> *Its not possible to burn or transfer non-existing tokens due to checks implemented in ERC1155. Added test cases.*

Notes & Additional Information

N-01 `BURNER_ROLE` Role Not Initialized During `Catalyst` Contract Initialization

The `Catalyst` contract defines a `BURNER_ROLE` role that can burn single or batch of tokens from a specified address by calling the `burnFrom` or `burnBatchFrom` functions respectively. However, no address is assigned to this role.

While the `admin` of the contract can call the `grantRole` function inherited from the `AccessControlUpgradeable` contract, the `burnFrom` and `burnBatchFrom` functions cannot be called until the `BURNER_ROLE` role is assigned.

Consider assigning the `BURNER_ROLE` role from within the `initialize` function.

Update: *Resolved. This is not an issue. The Sandbox team stated:*

> *Only AssetCreate will be given a burner role. The role will be assigned only after AssetCreate is deployed through deployment scripts. No action taken here.*

N-02 `public` Functions That Should Have `external` Visibility

The following `public` functions should be `external`:

- The `initialize` function in the `Catalyst` contract.

Consider changing the visibility of these functions to `external` in order to clarify that these functions will only be called by external contracts.

Update: *Resolved in pull request #1126.*

N-03 Typographical Errors

To improve code readability, consider removing the following typographical errors from the codebase:

- On line 41 of `Catalyst.sol`, "*THis*" should be "*This*"
- On line 239 and line 256 of `Catalyst.sol`, "*aditional*" should be "*additional*"
- On line 297 of `Catalyst.sol`, "*Opensea.can*" should be "*Opensea. Can*"

Update: *Resolved in pull request #1128. The Sandbox team stated:*

> *Fixed suggested and other found typographical errors.*

N-04 Unused Imports

Throughout the codebase, there are imports that are unused and could be removed. For instance:

- Import `ERC2981Upgradeable` of `Catalyst.sol`
- Import `IRoyaltyManager` of `Catalyst.sol`
- Import `IERC2981Upgradeable` of `Catalyst.sol`

Consider removing unused imports to improve the overall clarity and readability of the codebase.

Update: *Resolved. The issue was resolved here.The Sandbox team stated:*

> *Removed as part of the code delivered shortly after audit start.*

Conclusion

Several low-severity issues and notes were reported, mainly addressing improvement opportunities to the overall quality of the codebase.