OpenZeppelin | security

The Sandbox Marketplace Audit

November 22, 2023

Table of Contents

Table of Contents 2

Summary 3

Scope 4

System Overview 5

Privileged Roles and Trust Assumptions 6

Medium Severity 8

M-01 ERC-721 Tokens Can Be Disguised as ERC-20 Tokens to Circumvent Royalties and Fees 8

M-02 Non-Whitelisted NFTs Can Be Traded Even When the whitelist Is Enabled 8

Low Severity 9

L-01 Cancellation of Orders Should Be Allowed When Protocol Is Paused 9

L-02 Missing Error Message in calculateRemaining 9

L-03 Royalty and Protocol Fees Will Not Get Transferred in Some Cases 10

Notes & Additional Information 10

N-01 Ensure roles[] and permissions[] Arrays Have Matching Lengths in the __Whitelist_init Function 10

N-02 Incorrect Documentation 11

N-03 Initialize Variables In the Same Order in Which They Are Passed as Arguments 11

N-04 Missing Named Parameters in Mappings 11

N-05 Renaming Suggestions 12

N-06 Replace Zero-Address Checks With Code Existence Checks 12

N-07 Typographical Errors 13

N-08 Unnecessary override Keyword Used 13

N-09 Using int/uint Instead of int256/uint256 13

N-10 Incorrect Typecasting 14

Conclusion 15

Summary

Type	NFT
Timeline	From 2023-10-16 To 2023-11-10
Languages	Solidity

Total Issues	15 (14 resolved, 1 partially resolved)
Critical Severity Issues	0 (0 resolved)
High Severity Issues	0 (0 resolved)
Medium Severity Issues	2 (2 resolved)
Low Severity Issues	3 (3 resolved)
Notes & Additional Information	10 (9 resolved, 1 partially resolved)

Scope

We audited the `thesandboxgame/sandbox-smart-contracts` repository at commit 522131c.

In scope were the following contracts:

```
packages
└─ marketplace/
   └─ contracts/
      ├─ Exchange.sol
      ├─ ExchangeCore.sol
      ├─ OrderValidator.sol
      ├─ RoyaltiesRegistry.sol
      ├─ TransferManager.sol
      ├─ Whitelist.sol
      ├─ interfaces/
      │  ├─ IOrderValidator.sol
      │  ├─ IRoyaltiesProvider.sol
      │  ├─ ITransferManager.sol
      │  └─ IWhitelist.sol
      └─ libraries/
         ├─ LibAsset.sol
         ├─ LibMath.sol
         └─ LibOrder.sol
```

In addition, we also audited the pull request #1310 at commit `63a1b43` and pull request #1317 at commit `7e3add0` as part of the fix review process.

System Overview

The Sandbox Marketplace allows its users to exchange ERC-721, ERC-1155, and ERC-20 tokens using signed orders which are created off-chain. Once a sell order (denoted as `left` order) is ready to be fulfilled, anyone can call the `matchOrders` function in the `Exchange` contract by providing details of the sell order, seller's (or `maker`) signature, buy order (denoted as `right` order) and buyer's (or taker's) signature. An order can be filled fully or partially.

The details of the maker's and taker's orders are verified against their respective signatures. These orders are then validated in the `OrderValidator` contract to ensure that the orders have not expired, the assets are compliant with the whitelist and the signatures are authentic. The signature verification is done using OpenZeppelin's `SignatureCheckerUpgradeable` contract, thereby allowing EOAs and ERC-1271-compliant contracts to sign the transactions.

The `Whitelist` contract is part of the `OrderValidator` contract's inheritance and handles asset whitelisting verification. It specifies three roles: `TSB_ROLE` for trading Sandbox collections, `PARTNER_ROLE` for trading partner collections, and `ERC20_ROLE` for authorizing trades involving ERC20-compliant payment tokens. Once validated, the orders are analyzed to calculate the order portion available to be filled. The orders are then matched to be either partially or totally filled, depending on the parameters of the respective left and right orders.

If only one of the assets is an ERC-20 token, the exchange of assets between the maker and taker takes place after the protocol fee and royalties have been deducted. The percentage of the amount deducted as protocol fee is set by the `EXCHANGE_ADMIN_ROLE` privileged role of the `Exchange` contract. The system applies different protocol fees for primary and secondary markets.

A primary market is one where the creator of the asset is the seller. In this scenario, no royalty is taken and the entire value of the transaction after the protocol fee deduction is paid out to the creator. For the secondary market, where the assets are sold by accounts other than the creator, a portion of transaction amount is deducted to pay a royalty to the creator and to Sandbox. The address and the split for each recipient are obtained from the `RoyaltiesRegistry` contract. By design, a royalty can never be more than or equal to half of the transaction value and the protocol fee is strictly less than 50% of the transaction value. After the deductions, the assets are transferred to the parties involved by the `TransferManager` contract and the transaction is completed.

Multiple orders sharing the same details can be distinguished by a salt. When the salt is equal to zero, only the order maker is allowed to match the orders. As such, this process skips the maker's signature verification. An order with zero salt cannot be partially filled or cancelled.

The Marketplace supports ERC-1776 meta-transactions that allow users to exchange assets using `SAND` tokens through the `matchOrdersFrom` function in the `Exchange` contract. In addition to matching assets, the `Exchange` contract allows the maker to cancel an order. Orders are cancelled using their `hashKey`, which are calculated using some fields of an order. Different orders can have the same `hashKey`.

The `Exchange` contract inherits the `PausableUpgradeable` contract from OpenZeppelin's upgradeable contracts library, which allows the system to be paused. Any account that possesses the `PAUSER_ROLE` privileged role can pause the system but it can only be unpaused by the `EXCHANGE_ADMIN_ROLE` of the `Exchange` contract. Orders cannot be matched or cancelled when the system is paused.

Privileged Roles and Trust Assumptions

The `Exchange` contract contains the following privileged roles:

- The `ERC1776_OPERATOR_ROLE` can send a meta-transaction for matching orders on behalf of the users who own `SAND` tokens but no native currency.
- The `EXCHANGE_ADMIN_ROLE` has the following capabilities:
  - Set a limit to the number of orders that can be matched in a transaction
  - Set protocol fee
  - Set the address that receives the protocol fee
  - San unpause the contract
  - Accounts with this role can skip paying fees and royalties
- The `PAUSER_ROLE` can pause the contract.
- The `DEFAULT_ADMIN_ROLE` has the following capabilities:
  - add a royalty address to the royalty registry
  - set the `OrderValidator` contract address

    - set the address of the trusted forwarder

The `Whitelist` contract contains the following privileged roles:

- The `DEFAULT_ADMIN_ROLE` can:
    - enable or disable a role
    - enable or disable a whitelist

Access to the back-end database allows for deletion of signed orders waiting to be fulfilled, posing a minor DOS risk.

Medium Severity

M-01 ERC-721 Tokens Can Be Disguised as ERC-20 Tokens to Circumvent Royalties and Fees

When trading an ERC-721 token for an ERC-20 token, fees and royalties are typically charged as part of the transaction. However, there is a vulnerability whereby if the `ERC20_ROLE` is disabled, malicious actors could exploit this by misrepresenting the `AssetClass` in both the buy and sell orders and by replacing the `Asset.value` with the `tokenId`.

The exploit is feasible because the `transferFrom` function signature is identical for both ERC-721 and ERC-20 tokens. As a result, the smart contract could be tricked into treating an ERC-721 token as an ERC-20 token. Since trades involving only ERC-20 tokens do not incur fees or royalties in this system, the transaction would succeed without charging the appropriate fees and royalties, effectively bypassing the intended trade rules.

To prevent this type of circumvention, ensure that the `ERC20_ROLE` remains enabled.

Update: *Resolved in pull request #1305 at commit 2f6207c. The Sandbox team stated:*

> *Following your suggestion, we enabled the ERC20 whitelist all times*

M-02 Non-Whitelisted NFTs Can Be Traded Even When the `whitelist` Is Enabled

The codebase has a whitelist for non-fungible tokens (NFTs) to ensure that an NFT has the `PARTNER_ROLE` or `TSB_ROLE`, and the role is enabled. However, the code incorrectly checks for this condition, thereby allowing non-whitelisted NFTs to be traded on the exchange when `whitelist` is enabled.

Consider implementing the check to first ensure that `whitelist` is enabled, and if it is, performing subsequent role-based checks.

Update: *Resolved in pull request #1280 at commit 5375082. The Sandbox team stated:*

> *Indeed, we renamed the variable for enabling the whitelist from 'openMarket' to 'isWhitelistsEnabled,' which have opposite meanings, resulting in the verification for the*

> *enablement of the whitelist to be inverted. This inversion led to a reversal in the verification process for the whitelist.*

Low Severity

L-01 Cancellation of Orders Should Be Allowed When Protocol Is Paused

The `cancel` function in the `Exchange` contract has the `whenNotPaused` modifier which prevents users from calling the function when the system is paused.

To give full control to the users over their assets, consider removing the `whenNotPaused` modifier from the `cancel` function to allow `makers` to cancel their orders whenever required.

Update: *Resolved in pull request #1299 at commit 5df1bc0.*

L-02 Missing Error Message in `calculateRemaining`

The protocol allows for partial fillings of orders. Each order is associated with a `hashKey` and multiple orders can share the same `hashKey` since it is derived from only certain components of the order. In the `calculateRemaining` function, which is used to calculate the remaining fillable amount of an order, a subtraction operation is performed.

This operation can result in an underflow and cause a revert without an error message. The underflow can occur because the fill amount of an order, which is shared among multiple orders with the same `hashKey`, can exceed the `takeAsset.value` of a particular order.

Consider adding an error message in the `calculateRemaining` function.

Update: *Resolved in pull request #1299 at commit cba4d23. The Sandbox team stated:*

> *We accepted the recommendation and added a require with an error message in the calculateRemaining function.*

L-03 Royalty and Protocol Fees Will Not Get Transferred in Some Cases

The `_transferPercentage` function of the `TransferManager` contract is used to calculate and transfer the royalty or the protocol fee when given the remaining amount of a sale at the time and a percentage.

If the amount calculated to be the royalty or the protocol fee turns out to be greater than the remaining amount, it is not transferred. This happens because the remaining amount is first set to zero and the royalty or protocol fee is then set to the remaining amount (which is zero). This change should ideally happen in the reverse order so that the royalty or protocol fee becomes equal to the remaining amount and the remaining amount becomes zero.

Consider changing the order of these two expressions.

Update: *Resolved in pull request #1295.*

Notes & Additional Information

N-01 Ensure `roles[]` and `permissions[]` Arrays Have Matching Lengths in the `__Whitelist_init` Function

In the `__Whitelist_init` function of the `Whitelist` contract, the check that the lengths of the `roles[]` and `permissions[]` are equal is missing.

Make sure that the lengths of `roles[]` and `permissions[]` arrays are equal.

Update: *Resolved, not an issue. The `__Whitelist_init` function in the `Whitelist` contract internally calls the `_setRolesEnabled` function which checks the length of the arrays.*

N-02 Incorrect Documentation

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

- Lines 86 and 87 of the `TransferManager` contract refer to a non-existent file.
- Lines 14 and 15 of the `ITransferManager` abstract contract refer to a non-existent file.
- Line 20 of the `IRoyaltiesProvider` interface refers to a non-existent file.
- Line 66 of the `RoyaltyRegistry` contract incorrectly mentions `royaltyType` to be `4` instead of `2`.
- Line 177 of the `ExchangeCore` contract refers to a non-existent data structure.

Consider fixing any instances of incorrect documentation.

Update: *Resolved at commit 9b66cb8 and at commit 046e942.*

N-03 Initialize Variables In the Same Order in Which They Are Passed as Arguments

In the `__TransferManager_init_unchained` function, `newRoyaltiesProvider` is initialized before `newDefaultFeeReceiver`.

To improve readability, consider initializing the variables in the same order in which they are passed as arguments.

Update: *Resolved in pull request #1302 at commit ae27b03.*

N-04 Missing Named Parameters in Mappings

Since Solidity 0.8.18, developers can utilize named parameters in mappings. This means mappings can take the form of `mapping(KeyType KeyName? => ValueType ValueName?)`. This updated syntax provides a more transparent representation of the mapping's purpose.

Consider adding named parameters to the `_rolesEnabled` mapping in the `Whitelist` contract to improve the readability, consistency and maintainability of the codebase.

Update: *Resolved in pull request #1302 at commit f49b5e8.*

N-05 Renaming Suggestions

Some of the variable names do not accurately reflect their purpose. As such, below is a non-comprehensive list of variable renaming suggestions, which would better convey the intention behind them:

- `value` should be renamed to `basisPoints`.
- `BASIS_POINTS` should be renamed to `TOTAL_BASIS_POINTS`.
- `royaltyProviders` should be renamed to `externalRoyaltyProviders`.
- `BY_TOKEN` should be renamed to `SET_INTERNALLY`.
- `setRoyaltiesByToken` should be renamed to `setRoyaltiesInternally`.
- `setProviderByToken` should be renamed to `setExternalProvider`.
- `getProvider` should be renamed to `getExternalProvider`.
- `_whitelistEnabled` should be renamed to `_NFTwhitelistEnabled`.
- `WhitelistsEnabled` should be renamed to `NFTWhitelistsEnabled`.
- `WhitelistsDisabled` should be renamed to `NFTWhitelistDisabled`.
- `enableWhitelists` should be renamed to `enableNFTWhitelists`.
- `disableWhitelists` should be renamed to `disableNFTWhitelists`.
- `isWhitelistsEnabled` should be renamed to `isNFTWhitelistsEnabled`.
- `"Exchange", "1"` should be renamed to `"The Sandbox Marketplace", "1.0.0"`.

Consider implementing these renaming suggestions to improve the readability and clarity of the codebase.

Update: *Partially resolved in pull request #1307 at commit aff4999. The Sandbox team stated:*

> *We discussed it internally, and decided to only apply the following recommendations:*
> *- value should be renamed to basisPoints.*
> *- BASIS_POINTS should be renamed to TOTAL_BASIS_POINTS.*
> *- "Exchange", "1" should be renamed to "The Sandbox Marketplace", "1.0.0".*

N-06 Replace Zero-Address Checks With Code Existence Checks

In the instances listed below, to verify if a smart contract exists at a given address, consider using the `isContract` function from the OpenZeppelin contracts library's `Address` contract.

- At Line 112 of `TransferManager.sol`

- At Line 86 of `ExchangeCore.sol`

***Update:** Resolved in pull request #1298.*

N-07 Typographical Errors

The following typographical errors were identified in the codebase:

- At line 20 of `IRoyaltiesProvider.sol`, "allroyalties" should be "all royalties".
- At line 83 of `RoyaltiesRegistry.sol`, "roayalty" should be "royalty".
- At line 115 of `RoyaltiesRegistry.sol`, "royalties sum more, than 100%" should be "royalties sum is more than 100%".
- At line 195 of `TransferManager.sol`, "in base points" should be "in basis points".
- Line 21 of `Whitelist.sol` mentions an incorrect word "enableability".

Consider fixing any instances of typographical errors to improve the readability of the codebase.

***Update:** Resolved in multiple pull requests, such as pull request #1302 and pull request #1316. All fixes are present in the codebase at the commit 63d5c9f.*

N-08 Unnecessary `override` Keyword Used

The `getRoyalties` function of the `RoyaltiesRegistry` contract has an unnecessary `override` keyword.

Consider removing the unnecessary `override` keyword to improve the clarity of the codebase.

***Update:** Resolved in pull request #1300.*

N-09 Using `int/uint` Instead of `int256/uint256`

In `LibAsset.sol`, `int/uint` is used instead of `int256/uint256`.

In favor of explicitness, consider replacing all instances of `int/uint` with `int256/uint256`.

***Update:** Resolved in pull request #1297.*

N-10 Incorrect Typecasting

There are some instances in the codebase where incorrect typecasting is being performed:

- In the `_getRecipients` function of the `RoyaltiesRegistry` contract, the `value` field of a `Part` array is typecasted from `uint256` to `uint96`. There is no need to do so as the `value` field is declared as `uint256`.
- In the `_calculateRoyalties` function of the `RoyaltiesRegistry` contract, `address` is unnecessarily typecasted to `address payable` even though `Part.account` is of type `address`.

Consider removing these unnecessary instances of typecasting.

Update: *Resolved in pull request #1296.*

Conclusion

The audited codebase is a new implementation of The Sandbox Marketplace. It allows users to exchange ERC-721, ERC-1155, and ERC-20 tokens using signed orders which are created off-chain.

The Sandbox team has been very receptive to the suggestions made after an early assessment of the project and has introduced substantial changes to the codebase. There has been a significant improvement in the quality of the codebase thereby proving that they value the security of their protocol and its users.

The Sandbox team was also very responsive to the queries asked by the auditors during the course of the current engagement and provided additional documentation where needed.