SOLIDIFIED

Audit Report for OpenSea Contract Creator - August 8, 2021

Summary

Audit Report prepared by Solidified covering the OpenSea Contract Creator smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on August 02, 2021, and the results are presented here.

Audited Files

The source code has been supplied in a private source code repository:

https://github.com/ProjectOpenSea/opensea-contract-creator

Commit number: `94599d5eea1d42e3c9f88530ff1916428731a06f`

UPDATE: Fixes received on August 4th in PR:
https://github.com/ProjectOpenSea/opensea-contract-creator/pull/17

Final Commit number: `23c44328b4e9a509541e666af909e422c19fd942`

Intended Behavior

OpenSea Contract Creator is a contract for easily creating custom collections for OpenSea, sharing one asset contract.

Findings

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for OpenSea Contract Creator - August 8, 2021

Issues Found

Solidified found that the OpenSea Contract Creator contracts contain 1 critical issue, 1 major issue, 1 minor issue, and 5 informational notes.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	ERC1155Tradable.sol: Anyone can burn() or batchBurn() tokens belonging to other accounts	Critical	Pending
2	AssetContract.sol: Function batchMint() allows minting more tokens than TOKEN_SUPPLY_CAP	Major	Resolved
3	AssetContract.sol: Function setPermanentURI() does not check that token _id exists and _uri is valid	Minor	Resolved
4	NativeMetaTransaction.sol: Consider declaring function executeMetaTransaction() as external	Note	Resolved
5	ERC1155Tradable.sol: Contract ERC1155Tradable redundantly re-implements functionality already present in its parent ERC1155 contract	Note	Acknowledged
6	EIP712Base.sol: In case a hard-fork/chain-split happens after the deployment of the contract, the signed messages will be valid on both chains, since chainid() is considered only during the initialization	Note	-
7	NativeMetaTransaction.sol: Last failed transaction can be replayed any time until the user issues a new successful transaction	Note	-
8	Misc Notes	Note	-

Critical Issues

1. ERC1155Tradable.sol: Anyone can burn() or batchBurn() tokens belonging to other accounts

The functions `burn()` and `batchBurn()` conduct no checks to determine if `_msgSender()` owns or has permission to burn `_from` tokens.

Contracts `AssetContract.sol` and `AssetContractShared.sol` are also vulnerable because they inherit the functionality from `ERC1155Tradable.sol`.

Recommendation

Consider checking that `_msgSender()` has ownership or approval to burn the tokens.

Major Issues

2. AssetContract.sol: Function batchMint() allows minting more tokens than TOKEN_SUPPLY_CAP

The `TOKEN_SUPPLY_CAP` check in function `batchMint()` can be overridden in case the same `id` is passed multiple times inside the `_ids` array.

Recommendation
Consider adding a new `beforeMint()` function that gets called before each new asset is minted in `ERC1155Tradable._batchMint()`, then override this function in `AssetContract` and perform the check there instead.

Status
Resolved

Minor Issues

3. AssetContract.sol: Function setPermanentURI() does not check that token _id exists and _uri is valid

The `setPermanentURI()` function can be used to permanently set a `uri` for a non-existent token `id`. Also, if the passed `_uri` is invalid (e.g. null), the contract will be left stuck with a permanent invalid token uri that cannot be fixed.

Recommendation
Check that the passed token `_id` exists and that the passed `_uri` has a valid value.

Status
Resolved

Informational Notes

4. NativeMetaTransaction.sol: Consider declaring function executeMetaTransaction() as external

Declaring function `executeMetaTransaction()` as `external` instead of `public` can potentially save a non trivial amount of gas, as it will prevent all the passed parameter arrays from being copied to memory, and instead read them directly from calldata.

Note
The same issue applies to all batch functions in `ERC1155Tradable`.

Status
Resolved

5. ERC1155Tradable.sol: Contract ERC1155Tradable redundantly re-implements functionality already present in its parent ERC1155 contract

Contract `ERC1155Tradable` descends from `ERC1155`, which already provides implementations of functions such as `balanceOf()`, `balanceOfBatch()`, `isApprovedForAll()`, `safeTransferFrom()`, amongst several others. The contract also redundantly declares the `balances` mapping, which has already been declared in its parent `ERC1155` contract.

Recommendation
Eliminate all redundant implementations.

Status

Acknowledged. Team's response: "*we reimplemented a lot of the ERC1155 base contract functionality because we needed direct access to the `balances` mapping, in order to properly emit events when lazy-minted NFTs are created*".

6. EIP712Base.sol: In case a hard-fork/chain-split happens after the deployment of the contract, the signed messages will be valid on both chains, since chainid() is considered only during the initialization

---

Recommendation

Consider storing the `chainid` set during the initialization and check if it is still the same in function `getDomainSeperator()`. If a difference is detected - recalculate `domainSeperator`.

7. NativeMetaTransaction.sol: Last failed transaction can be replayed any time until the user issues a new successful transaction

---

Recommendation

Consider adding an execution deadline to a signed `MetaTransaction`.

8. Misc Notes

- TokenIdentifiers.sol: Constant `ADDRESS_BITS` is declared but never used.
- TokenIdentifiers.sol: Function `tokenIndex()` is declared but never used.

Audit Report for OpenSea Contract Creator - August 8, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of OpenSea or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.