OneOf

Smart Contract Audit Report

February 05, 2021

IMMUNE BYTES

Introduction

1. About OneOf

OneOf is a new, Green NFT platform built specifically for the music, sports & LifeStyles community and designed to break down economic barriers to entry. The platform serves to connect people and collectors.

The OneOf Marketplace is where members can post their OneOf NFTs for sale to other members. It's as simple as clicking the "Post for Sale" button next to your NFT under "My Collection," setting a USD price, and it's immediately listed!

Visit https://www.oneof.com/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 125+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-ups with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The OneOf team has provided the following doc for the purpose of audit:

1. https://github.com/OneOf-Inc/oneof-poly-contract/blob/main/README.md

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: OneOf
- Token Name: Destroyable.sol, ERC1155.sol, ERC1155AllowanceWrapper.sol, NFTOneof.sol, NFTOneofExtention.sol
- GitHub Address: https://github.com/OneOf-Inc/oneof-poly-contract
- Commit Hash for Initial Audit: 3a43426ccab1c3e981422cf4ec5f546505dd7531
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

Every issue in this report was assigned a severity level from the following:

Admin/Owner Privileges can be misused either intentionally or unintentionally.

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	6
Closed	-	-	-

Contract Name: NFTOneof.sol

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

1. **External visibility should be preferred**

   Explanation:
   Functions that are never called throughout the contract should be marked as **external** visibility instead of **public** visibility.
   This will effectively result in Gas Optimization as well.

   Therefore, the following function must be marked as **external** within the contract:

   - mintBatch()
   - powerMint()
   - select()
   - transfer()
   - mintNonFungible()
   - mintFungible()

   Recommendation:
   If the **PUBLIC** visibility of the above-mentioned functions is not intended, then the **EXTERNAL** Visibility keyword should be preferred.

2. **Absence of Error messages in Require Statements**
   Line no - 149, 169

   Explanation:
   The **NFTOneOf** contract includes a few functions(at the above-mentioned lines) that don't contain any error message in the **require** statement.

   While this makes it troublesome to detect the reason behind a particular function revert, it also reduces the readability of the code.

Recommendation:
Error Messages must be included in every require statement in the contract.

Recommendations/Informational

1. **Redundant comparisons to boolean Constants**
   Line no: 120

   Explanation:
   Boolean constants can directly be used in conditional statements or require statements. Therefore, it's not considered a better practice to explicitly use **TRUE** or **FALSE** in the **require** statements.

   Recommendation:
   The equality to boolean constants could be removed from the above-mentioned line.

2. **Coding Style Issues in the Contract**

   Explanation:
   Code readability of a Smart Contract is largely influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future.

```
Parameter NFTOneof.mintBatch(address,NFTOneof.Mint[],bytes)._to (src/NFTOneof.sol#37) is not in mixedCase
Parameter NFTOneof.mintBatch(address,NFTOneof.Mint[],bytes)._data (src/NFTOneof.sol#39) is not in mixedCase
Parameter NFTOneof.getNonFungibleItems(uint256,uint256)._type (src/NFTOneof.sol#139) is not in mixedCase
Parameter NFTOneof.mintNonFungible(uint256,address[],string)._type (src/NFTOneof.sol#163) is not in mixedCase
Parameter NFTOneof.mintNonFungible(uint256,address[],string)._to (src/NFTOneof.sol#164) is not in mixedCase
Parameter NFTOneof.mintFungible(uint256,address[],uint256[])._id (src/NFTOneof.sol#189) is not in mixedCase
Parameter NFTOneof.mintFungible(uint256,address[],uint256[])._to (src/NFTOneof.sol#190) is not in mixedCase
Parameter NFTOneof.mintFungible(uint256,address[],uint256[])._quantities (src/NFTOneof.sol#191) is not in mixedCase
```

   During the automated testing, it was found that the NFTOneOf contract had quite a few code-style issues.

   Recommendation:
   Therefore, it is recommended to fix the issues like naming convention, indentation, and code layout issues in a smart contract.

3. **NatSpec Annotations should be included**

   Explanation:
   The smart contracts do not include the NatSpec annotations adequately.

   Recommendation:
   Cover by NatSpec all Contract methods.

Contract Name: ERC1155.sol

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

1. **Ambiguous and Repetitive error messages found in require statements**
   Line no: 219, 395, 396, 433, 434

   Explanation:
   During the code review, it was found the **require** statements in certain functions include very ambiguous error messages.

   Moreover, in some instances, one similar error message is repeated in multiple require statements which badly affects the code readability and also throws inadequate messages to the user in case of function revert.

   Recommendation:
   Adequate error messages should be provided in the **require** statement of the functions.

2. **Absence of Error messages in Require Statements**
   Line no - 262

   Explanation:
   The ERC1155 contract includes a few functions(at the above-mentioned lines) that don't contain any error message in the **require** statement.

   While this makes it troublesome to detect the reason behind a particular function revert, it also reduces the readability of the code.

   Recommendation:
   Error Messages must be included in every **require** statement in the contract.

Contract Name: ERC1155AllowanceWrapper.sol

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

1. **Unchecked allowance value in safeTransferFrom & safeBatchTransferFrom function**
   Line no: 95 and 121

   Explanation:
   During the code review it was found that within the safeTransferFrom and safeBatchTrasnferFrom functions, the **_value**, passed in the arguments, is deducted from the allowance without checking if the caller was approved with the sufficient value in the first place..

   Since the newer versions of solidity, handle the arithmetic underflows or overflows, the above-mentioned issue doesn't break the intended behavior of the contract. However, it leads to an inadequate error message.

   Recommendation:
   Adequate allowance checks could be included.

2. **Absence of Error messages in Require Statements**
   Line no - 47

   Explanation:
   The **ERC1155AllowanceWrapper** contract includes an **approve** function that doesn't contain any error message in the **require** statement.

   While this makes it troublesome to detect the reason behind a particular function revert, it also reduces the readability of the code.

   Recommendation:
   Error Messages must be included in every require statement in the contract.

Automated Audit Result

1. NFTOneofExtension.sol

```
Compiled with solc
Number of lines: 1691 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 13 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 9
Number of informational issues: 56
Number of low issues: 11
Number of medium issues: 8
Number of high issues: 0

ERCs: ERC165

+--------------------+-------------+--------+------------+--------------+--------------+
|        Name        | # functions |  ERCS  | ERC20 info | Complex code |   Features   |
+--------------------+-------------+--------+------------+--------------+--------------+
|  IERC1155Receiver  |      3      | ERC165 |            |      No      |              |
|      Address       |      11     |        |            |      No      |   Send ETH   |
|                    |             |        |            |              | Delegatecall |
|                    |             |        |            |              |   Assembly   |
|      Strings       |      4      |        |            |     Yes      |              |
| NFTOneofExtention  |      76     | ERC165 |            |     Yes      |              |
+--------------------+-------------+--------+------------+--------------+--------------+
src/NFTOneofExtention.sol analyzed (13 contracts)
```

2. NFTOneof.sol

```
Compiled with solc
Number of lines: 1892 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 14 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 13
Number of informational issues: 67
Number of low issues: 13
Number of medium issues: 9
Number of high issues: 0

ERCs: ERC165

+------------------+-------------+--------+------------+--------------+--------------+
|       Name       | # functions |  ERCS  | ERC20 info | Complex code |   Features   |
+------------------+-------------+--------+------------+--------------+--------------+
| IERC1155Receiver |      3      | ERC165 |            |      No      |              |
|     Address      |      11     |        |            |      No      |   Send ETH   |
|                  |             |        |            |              | Delegatecall |
|                  |             |        |            |              |   Assembly   |
|     Strings      |      4      |        |            |     Yes      |              |
|     NFTOneof     |      84     | ERC165 |            |     Yes      |              |
+------------------+-------------+--------+------------+--------------+--------------+
src/NFTOneof.sol analyzed (14 contracts)
```

3. ERC1155AllowanceWrapper.sol

```
Compiled with solc
Number of lines: 343 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 4 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 1
Number of informational issues: 26
Number of low issues: 0
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC165

+-------------------------+-------------+--------+------------+-------------+--------------------+
|          Name           | # functions |  ERCS  | ERC20 info | Complex code |      Features      |
+-------------------------+-------------+--------+------------+-------------+--------------------+
| ERC1155AllowanceWrapper |      18     | ERC165 |            |      No      | Tokens interaction |
+-------------------------+-------------+--------+------------+-------------+--------------------+
src/ERC1155AllowanceWrapper.sol analyzed (4 contracts)
```

Unit Testing

```
Compiling 15 files with 0.8.6
Generating typings for: 15 artifacts in dir: src/types for target: ethers-v5
Successfully generated 27 typings!
Successfully generated 3 typings for external artifacts!
Compilation finished successfully

  NFTOneof setup tests
deploying contract from: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
    deployment
      ✓ should set the owner to admin only
      ✓ only admin should be a creator
      ✓ test token type bit implementations
      ✓ nonce should be 0
      ✓ destroyed should be initalized as false
      ✓ paused should be initialized as false
    with creators
      ✓ admin should still be only owner
      ✓ admin, dakota, mark, robert, minter, and opensea should all be creators
    with NFT types
      ✓ there should be an nft type for dakota, mark, robert, minter, opensea and they should all only be minte
rs for there nft except admin
    with minted NFT
      ✓ each creator should have 15 minted nfts owned by them
    with power minted keys
      ✓ each creator should have 1000 power minted nfts created by them

  NFTOneof  Minting Tests
    ✓ Minting using steps createMixedFungibleType ->  mintNonFungible
    ✓ Minting using mintBatch
    ✓ Minting using power minting
```

```
  NFTOneof Burn Tests
    ✓ Burning all of minters nfts
    ✓ Burning dakotas first nft

  NFTOneof Transfer Tests
    ✓ Transfer minter nfts to dakota and robert
    ✓ safeBatchTransferFrom dakota  nfts  to minter
    ✓ safeTransferFrom   nft from robert  to minter
    ✓ Select powerminted nfts from minter to robert

  NFTOneof Administrative  Tests
    ✓ Pause

  NFTOneof contract
    Power Minting
      ✓ Robert should have  powermint 10 tokens and selected them for dakota
    Transfer
      ✓ Robert should have  powermint 10 tokens and selected them for dakota. Dakota then transfers them to Mark
```

Concluding Remarks

While conducting the audits of the OneOf smart contracts, it was observed that the contracts contain only Low severity issues.

Our auditors suggest that Low severity issues should be resolved by the developers. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's 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.

Our team does not endorse the OneOf platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes