Niftify Smart Contracts Preliminary Audit Report

Project Synopsis

Project Name	**Niftify**
Platform	Ethereum, Solidity
Github Repo	https://github.com/Niftify-io/token-smart-contracts
Deployed Contract	Not Deployed
Total Duration	6 Days
Timeline of Audit	**25th August 2021 to 2nd September 2021**

Contract Details

Total Contract(s)	4
Name of Contract(s)	*NiftifyERC1155.sol, NiftifyERC721.sol, TokenStorage.sol, NiftifyNFTVoucher.sol*
Language	Solidity
Commit Hash	`19d63f7369e3b3c61a860ea147ac7bfb8c689ad7`

Contract Vulnerabilities Synopsis

Issues	Open Issues	Closed Issues
Critical Severity	0	0
Medium Severity	2	0
Low Severity	2	0
Information	6	0
Total Found	0	0

Detailed Results

The contract has gone through several stages of the audit procedure that includes structural analysis, automated testing, manual code review, etc.

All the issues have been explained and discussed in detail below. Along with the explanation of the issue found during the audit, the recommended way to overcome the issue or improve the code quality has also been mentioned.

A. Contract Name: TokenStorage

High Severity Issues

None Found

Medium Severity Issues

A.1 Multiplication is being performed on the result of Division

Line no - *83-94*

***Explanation*:**

During the automated testing of the **TokenStorage** contract, it was found that the contract includes a function that performs multiplication on the result of a Division.

Integer Divisions in Solidity might truncate. Moreover, this performing division before multiplication might lead to a loss of precision.

The following function involves division before multiplication in the mentioned lines:

- ***mulScale*** *at 83-94*

Automated Tets result has been attached below:

```
TokenStorage.mulScale(uint256,uint256,uint32) (myFlats/flatTokenStorage.sol#532-543) performs a multiplication on the result of a division:
        -c = y / scale (myFlats/flatTokenStorage.sol#539)
        -a * c * scale + a * d + b * c + (b * d) / scale (myFlats/flatTokenStorage.sol#542)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

Recommendation:

Solidity doesn't encourage arithmetic operations that involve division before multiplication.

Therefore it is highly recommended to write extensive test cases to ensure the function behavior is as per the expectation. The above-mentioned function should be redesigned only if they do not lead to expected results.

A.2 Operator Role has not been set up in the contract

Line no - 8

***Explanation*:**

The **OPERATOR_ROLE** in the TokenStorage contract has not been assigned to any particular address within the contract.

Is this Intended?

While the ***Minter*** and ***Redeemer*** role can be handled later via admin, the operator role might need an initial setup as functions like ***setBlocked*** uses Operator_Roles as a requirement, in the contract.

```
66
67    function setBlocked(uint256 _tokenId, bool _value) external onlyOperator {
68      _blocked[_tokenId] = _value;
69    }
70
```

Recommendation:
If the above-mentioned scenario is not intended, then the Operator Role should be assigned to a particular address while deployment.

Low Severity Issues

A.3 Absence of Error messages in Require Statements

Line no - 47
Description:
The **TokenStorage** contract includes a **require statement** in the ***_setRoyalty*** function(*at the above-mentioned line*) that doesn't contain any error message within itself.
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.

A.4 External Visibility should be preferred

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

The following function in the TokenStorage contract has been assigned **public visibility** but never called form within the contract:

- ***metadataHash() #Line 22***

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

Informational

A.5 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.

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

```
INFO:Detectors:
Parameter TokenStorage.metadataHash(uint256)._tokenId (myFlats/flatTokenStorage.sol#471) is not in mixedCase
Parameter TokenStorage.creator(uint256)._tokenId (myFlats/flatTokenStorage.sol#483) is not in mixedCase
Parameter TokenStorage.royalty(uint256)._tokenId (myFlats/flatTokenStorage.sol#491) is not in mixedCase
Parameter TokenStorage.royaltyInfo(uint256,uint256)._tokenId (myFlats/flatTokenStorage.sol#500) is not in mixedCase
Parameter TokenStorage.royaltyInfo(uint256,uint256)._salePrice (myFlats/flatTokenStorage.sol#500) is not in mixedCase
Parameter TokenStorage.blocked(uint256)._tokenId (myFlats/flatTokenStorage.sol#512) is not in mixedCase
Parameter TokenStorage.setBlocked(uint256,bool)._tokenId (myFlats/flatTokenStorage.sol#516) is not in mixedCase
Parameter TokenStorage.setBlocked(uint256,bool)._value (myFlats/flatTokenStorage.sol#516) is not in mixedCase
Parameter TokenStorage.supportsInterface(bytes4)._interfaceId (myFlats/flatTokenStorage.sol#520) is not in mixedCase
```

Recommendation:

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

A.6 NatSpec Annotations must be included

Description:

The smart contracts do not include the NatSpec annotations adequately.

Recommendation:

Cover by NatSpec all Contract methods.

B. Contract Name: NiftifyERC721

Informational

B.1 Coding Style Issues in the Contract

Explanation:

Code readability of a Smart Contract is largely influenced by the Coding Style.

During the automated testing, it was found that the **NiftifyERC721** contract had quite a few code-style issues.

```
Parameter NiftifyERC721.redeem(NiftifyNFTVouchers.ERC721Voucher)._voucher (myFlats/flaterc721.sol#2027) is not in mixedCase
Parameter NiftifyERC721.mint(NiftifyNFTVouchers.ERC721Voucher)._voucher (myFlats/flaterc721.sol#2044) is not in mixedCase
Parameter NiftifyERC721.mintAndTransfer(NiftifyNFTVouchers.ERC721Voucher,address)._voucher (myFlats/flaterc721.sol#2058) is not in mixedCase
Parameter NiftifyERC721.mintAndTransfer(NiftifyNFTVouchers.ERC721Voucher,address)._receiver (myFlats/flaterc721.sol#2058) is not in mixedCase
Parameter NiftifyERC721.updateMetadataHash(NiftifyNFTVouchers.UpdateMetadataHashVoucher)._voucher (myFlats/flaterc721.sol#2066) is not in mixedCase
Parameter NiftifyERC721.supportsInterface(bytes4)._interfaceId (myFlats/flaterc721.sol#2098) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Recommendation:

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

B.2 NatSpec Annotations must be included

Description:

The smart contracts do not include the NatSpec annotations adequately.

Recommendation:

Cover by NatSpec all Contract methods.

C. Contract Name: NiftifyERC1155

Informational

C.1 Coding Style Issues in the Contract

Explanation:

Code readability of a Smart Contract is largely influenced by the Coding Style.

During the automated testing, it was found that the **NiftifyERC1155** contract had quite a few code-style issues.

```
Parameter NiftifyERC1155.redeem(NiftifyNFTVouchers.ERC1155Voucher)._voucher (myFlats/flaterc1155.sol#2131) is not in mixedCase
Parameter NiftifyERC1155.mint(NiftifyNFTVouchers.ERC1155Voucher)._voucher (myFlats/flaterc1155.sol#2157) is not in mixedCase
Parameter NiftifyERC1155.mintAndTransfer(NiftifyNFTVouchers.ERC1155Voucher,address)._voucher (myFlats/flaterc1155.sol#2174) is not in mixedCase
Parameter NiftifyERC1155.mintAndTransfer(NiftifyNFTVouchers.ERC1155Voucher,address)._receiver (myFlats/flaterc1155.sol#2174) is not in mixedCase
Parameter NiftifyERC1155.updateMetadataHash(NiftifyNFTVouchers.UpdateMetadataHashVoucher)._voucher (myFlats/flaterc1155.sol#2188) is not in mixedCase
Parameter NiftifyERC1155.supportsInterface(bytes4)._interfaceId (myFlats/flaterc1155.sol#2225) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Recommendation:

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

C.2 NatSpec Annotations must be included

Description:

The smart contracts do not include the NatSpec annotations adequately.

Recommendation:

Cover by NatSpec all Contract methods.

Contract Name: NiftifyNFTVouchers

No issues Found

Automated Test Results

1. Automated Test Results for TokenStorage

```
Compiled with solc
Number of lines: 568 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 8 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 4
Number of informational issues: 19
Number of low issues: 0
Number of medium issues: 1
Number of high issues: 0

ERCs: ERC165

+--------------+------------+--------+-----------+-------------+----------+
|     Name     | # functions |  ERCS  | ERC20 info | Complex code | Features |
+--------------+------------+--------+-----------+-------------+----------+
|   Strings    |      4     |        |           |     Yes     |          |
| TokenStorage |     34     | ERC165 |           |     No      |          |
+--------------+------------+--------+-----------+-------------+----------+
INFO:Slither:myFlats/flatTokenStorage.sol analyzed (8 contracts)
```

2. Automated Test Results for NiftifyNFTVoucher

```
Compiled with solc
Number of lines: 445 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 3 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 23
Number of low issues: 1
Number of medium issues: 0
Number of high issues: 0

+--------------------+-------------+------+-----------+-------------+-----------+
|        Name        | # functions | ERCS | ERC20 info | Complex code |  Features |
+--------------------+-------------+------+-----------+-------------+-----------+
|       ECDSA        |      9      |      |           |      No     | Ecrecover |
|                    |             |      |           |             |  Assembly |
| NiftifyNFTVouchers |      9      |      |           |      No     |           |
+--------------------+-------------+------+-----------+-------------+-----------+
INFO:Slither:myFlats/flatVouchers.sol analyzed (3 contracts)
```

3. Automated Test Results for NiftifyERC1155

```
Compiled with solc
Number of lines: 2236 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 21 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 9
Number of informational issues: 59
Number of low issues: 11
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   |
|      ECDSA      |      9      |        |           |      No      |  Ecrecover   |
|                 |             |        |           |              |   Assembly   |
|     Strings     |      4      |        |           |     Yes      |              |
|  NiftifyERC1155 |      93     | ERC165 |           |      No      |  Ecrecover   |
+-----------------+-------------+--------+-----------+-------------+--------------+
INFO:Slither:myFlats/flaterc1155.sol analyzed (21 contracts)
```

4. Automated Test Results for NiftifyERC721

```
Compiled with solc
Number of lines: 2109 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 21 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 56
Number of low issues: 4
Number of medium issues: 2
Number of high issues: 0

ERCs: ERC165, ERC721

+-----------------+-------------+---------------+-----------+-------------+--------------+
|       Name      | # functions |      ERCS     | ERC20 info | Complex code |   Features   |
+-----------------+-------------+---------------+-----------+-------------+--------------+
| IERC721Receiver |      1      |               |           |      No      |              |
|     Address     |      11     |               |           |      No      |   Send ETH   |
|                 |             |               |           |              | Delegatecall |
|                 |             |               |           |              |   Assembly   |
|     Strings     |      4      |               |           |     Yes      |              |
|      ECDSA      |      9      |               |           |      No      |  Ecrecover   |
|                 |             |               |           |              |   Assembly   |
|  NiftifyERC721  |      99     | ERC165,ERC721 |           |      No      |  Ecrecover   |
|                 |             |               |           |              |   Assembly   |
+-----------------+-------------+---------------+-----------+-------------+--------------+
INFO:Slither:myFlats/flaterc721.sol analyzed (21 contracts)
```

Test Cases Results

1. Test Cases for ERC1155

```
NiftifyERC1155
  Vouchers
    ✓ Should redeem NFT from signature (87ms)
    ✓ Should fail to redeem voucher, that was already redeemed (167ms)
  Minting
    ✓ Anyone should be able to mint any token with admins signature (87ms)
    ✓ Anyone should be able to mint and transfer any token with admin signature (85ms)
    ✓ Should not be able to mint token with non operator signature (43ms)
  Storage
    ✓ Should be able to get hash of any token
    ✓ Should be able to get creator address of any token
    ✓ Should be able to get royalty of any token
  UpdateHash
    ✓ Should be able to update hash with owner & creator signatures for token with supply==1 (55ms)
    ✓ Should not be able to update hash with signatures not from owner or creator (77ms)
    ✓ Should not be able to update metadata hash for token with supply > 1 (59ms)
  Block tokens
    ✓ Operator should be able to block any token (42ms)
    ✓ Operator should be able to unblock any token
  Pausable
    ✓ Operator should be able to pause/unpause contract
    ✓ Non operator should not be able to pause/unpause contract (53ms)
    ✓ Transactions should not work while paused (73ms)
  Royalty calculation
    ✓ Should calculate royalty correctly
    ✓ Shouldn't overflow when calculating royalty


18 passing (9s)
```

2. Test Cases for ERC721

```
NiftifyERC721
  Vouchers
    ✓ Should redeem an NFT from signature (101ms)
    ✓ Should fail to redeem voucher, that was already redeemed (155ms)
  Minting
    ✓ Anyone should be able to mint any token with admin signature (92ms)
    ✓ Anyone should be able to mint and transfer any token with admin signature (76ms)
    ✓ Should not be able to mint token with non operator signature (45ms)
  Storage
    ✓ Should be able to get hash of any token
    ✓ Should be able to get creator address of any token
    ✓ Should be able to get royalty of any token
  UpdateHash
    ✓ Should be able to update hash with owner & creator signatures (76ms)
    ✓ Should not be able to update hash with signatures not from owner or creator (100ms)
  Block tokens
    ✓ Operator should be able to block any token (40ms)
    ✓ Operator should be able to unblock any token
  Pausable
    ✓ Operator should be able to pause/unpause contract (46ms)
    ✓ Non operator should not be able to pause/unpause contract (80ms)
    ✓ Transactions/minting should not work while paused (64ms)
  Royalty calculation
    ✓ Should calculate royalty correctly
    ✓ Shouldn't overflow when calculating royalty


17 passing (3s)
```