Niftify

ERC20

Smart Contract Audit
Final Report

August 31, 2021

Introduction

1. About Niftify

Niftify is on a mission to create a strong community of NFT enthusiasts (content creators, traders, investors, etc.) wanting to buy, sell or trade NFTs, in a hassle free environment.

There are already several NFT marketplaces, most of which need programming expertise to make successful use of them, which introduces more difficulty and the lack of accessibility. They are typically complex to use and difficult to understand and require in-depth trading knowledge and programming skills, which alienates new customers and proves to be time-consuming for the more experienced traders.
Users can only buy Non-fungible tokens (NFTs) with cryptocurrencies. While on a worldwide scale a high percentage of users do not have access to cryptocurrencies or simply do not want to use them, not to talk of using to purchase a simple NFT.
Another issue NFTs are facing is the fact that only traditional media types such as art, music, videos as well as gaming assets are on the market.

Visit https://niftify.io/ to know more about.

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 75+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up 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 Niftify team has provided the following doc for the purpose of audit:

1. https://niftify.io/lightpaper
2. https://docs.google.com/document/d/1s6r7jG7lWeOIctJnCSBneg1ZC0xH2AtcyYwjQnAEI4o/edit

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: Niftify
- Contracts Name: NiftifyERC20.sol
- Languages: Solidity(Smart contract)
- Github commit for initial audit: 19d63f7369e3b3c61a860ea147ac7bfb8c689ad7
- Github commit for final audit: 775b8bd47cb7c1d5856c73d1bc05f8c5df63adef
- Platforms and Tools: Remix IDE, Truffle, Ganache, Solhint, 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 References

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

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	-	-	-
Closed	-	-	-

High Severity Issues

No Issues Found.

Medium severity issues

No issues found.

Low Severity Issues

No Issues Found.

Recommendations

1. **Inadequate Test Cases for NiftifyERC20 contract**

   Explanation:
   The test scripts for the NifitifyERC20.sol contract don't include the test cases for AccessControl procedures in the contract.

   Recommendation:
   Keeping in mind the immutable nature of Smart Contracts, it's always considered a better practice to include adequate test scripts that cover every aspect of a function in the contract.

   Amended (Aug 31st 2021): The issue has been fixed by the Niftify team and is no longer present in commit 775b8bd47cb7c1d5856c73d1bc05f8c5df63adef.

2. **NatSpec Annotations must be included**

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

   Recommendation:
   Cover by NatSpec all Contract methods.

   Amended (Aug 31st 2021): The issue has been fixed by the Niftify team and is no longer present in commit 775b8bd47cb7c1d5856c73d1bc05f8c5df63adef.

Unit Test

```
Pausable ERC20 Token
  ✓ allows to transfer when unpaused (53ms)
  ✓ allows to transfer when paused and then unpaused (59ms)
  ✓ reverts when trying to transfer when paused (150ms)
  ✓ Should be able to transfer with permit (147ms)
  transfer from
    ✓ allows to transfer from when unpaused
    ✓ allows to transfer when paused and then unpaused (43ms)
    ✓ reverts when trying to transfer from when paused
  Access control
    ✓ Admin should be able to assign operator role
    ✓ Operator should be able to pause/unpause contract
    ✓ Operator should not be able to assign roles (67ms)
    ✓ Admin should be able to revoke operator role
    ✓ Non operator should not be able to pause/unpause contract
    ✓ Admin should be able to assign/revoke admin role to anyone


13 passing (7s)
```

Automated Test Results

```
Compiled with solc
Number of lines: 1552 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 17 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 15
Number of informational issues: 27
Number of low issues: 5
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC165, ERC20

+--------------+------------+-------------+--------------------+--------------+-----------+
|     Name     | # functions |     ERCS    |     ERC20 info     | Complex code |  Features |
+--------------+------------+-------------+--------------------+--------------+-----------+
|   Strings    |      4     |             |                    |     Yes      |           |
|    ECDSA     |      9     |             |                    |      No      | Ecrecover |
|              |            |             |                    |              |  Assembly |
|   Counters   |      4     |             |                    |      No      |           |
| NiftifyERC20 |     71     | ERC20,ERC165 |      Pausable      |      No      | Ecrecover |
|              |            |             |     No Minting     |              |  Assembly |
|              |            |             | Approve Race Cond. |              |           |
|              |            |             |                    |              |           |
+--------------+------------+-------------+--------------------+--------------+-----------+
```

Concluding Remarks

While conducting the audits of the Niftify smart contract, it was observed that the contract contained only a few areas of recommendations. No High, Medium and Low severity is found.

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

- ***The Niftify team has fixed the recommendations.***

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 Niftify 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