FanslandNFT
Audit Report

Wed Mar 13 2024

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

FanslandNFT Audit Report

1 Executive Summary

1.1 Project Information

Description	The audit scope is the proxy address 0xf310C3f033AFe83038590c7671D4F0b3F8325850 with its implementation address 0x6613fC4e065fd6FEfB05a7f3855b670F432691b4. The production environment proxy address is 0xbf36ab3aed81bf8553b52c61041904d98ee882c2.
Type	NFT
Auditors	ScaleBit
Timeline	Mon Mar 11 2024 - Mon Mar 11 2024
Languages	Solidity
Platform	BSC
Methods	Architecture Review, Unit Testing, Manual Review

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
FNFT	FanslandNFT.sol	a6f3571f78ccaa59dafb44fd94e50fbd7f2ea763
FNFTU	FanslandNFT_Update.sol	7e9298f17b8524fc2d0c9b390fc918772cbd0b78

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	4	1
Informational	0	0	0
Minor	4	3	1
Medium	1	1	0
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by FanslandNFT to identify any potential issues and vulnerabilities in the source code of the FanslandNFT smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 5 issues of varying severity, listed below.

ID	Title	Severity	Status
FNF-1	May Cause `addNftType` Call to Fail	Medium	Fixed
FNF-2	Sensitive Operation Lacks Event	Minor	Acknowledged
FNF-3	Duplicate Checks	Minor	Fixed
FNF-4	Code Optimization	Minor	Fixed
FNF-5	Using the `safeMath` Library	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the FanslandNFT Smart Contract :

Admin

- The admin can change allowTransfer through `updateAllowTransfer()` .
- The admin can change uri through `updateNftTypeURI()` .
- The admin can change the name through `updateNftTypeName()` .
- The admin can change the maxSupply through `updateNftTypeMaxSupply()` .
- The admin can change the isSaleActive through `updateNftTypeSaleActive()` .
- The admin can change the price through `updateNftTypePrice()` .
- The admin can change the openSale through `setSaleActive()` .
- The admin can change the baseURI through `setBaseURI()` .
- The admin can add a new NftType through `addNftType()` .

User

- The user can mint different types of nft through `mintBatch()` .

4 Findings

FNF-1 May Cause `addNftType` Call to Fail

Severity: Medium

Status: Fixed

Code Location:

FanslandNFT.sol#159-176

Descriptions:

The `addNftType` function is utilized for creating an `NftType`, and it mandates that `nftTypeMap[id].maxSupply == 0 && bytes(nftTypeMap[id].name).length == 0`. However, `updateNftTypeName` and `updateNftTypeMaxSupply` can be used to modify the data of the `NftType` corresponding to the id without initializing it. Moreover, once modified, the values cannot be reverted to zero. This can result in the failure of the `addNftType` function when called.

Suggestion:

It is recommended to check that the `name` and `maxSupply` under the corresponding id are not set to zero when calling the `updateNftTypeName` and `updateNftTypeMaxSupply` functions.

Resolution:

This issue has been fixed.

FNF-2 Sensitive Operation Lacks Event

Severity: Minor

Status: Acknowledged

Code Location:

FanslandNFT.sol#85,125,155,159,167,178,182,317,321

Descriptions:

In the contract, some sensitive operations lack event listeners, making it difficult for external tracking of changes in related data within the contract. The functions affected by this issue include `updateAllowTransfer()` , `addNftType()` , `updateNftTypeURI()` , and `updateNftTypeName()` , among others.

Suggestion:

It is recommended to add events similar to other operations to facilitate monitoring changes within the contract.

Resolution:

This issue has been acknowledged by the client.

FNF-3 Duplicate Checks

Severity: Minor

Status: Fixed

Code Location:

FanslandNFT.sol#204-207

Descriptions:

In the contract, the `calculateTotal` function is exclusively called by the `mintBatch` function. Within the `calculateTotal` function, there is a check `nftTypeMap[typeId].maxSupply >= result`, and in the `mintBatch` function, the `_mintNFT` function is called, where another check is performed: `nftTypeMap[typeId].totalSupply <= nftTypeMap[typeId].maxSupply`. The effects of these two checks are identical. Moreover, there are some issues with the check in the `calculateTotal` function. If a user uses the same id in the `typeIds` parameter, the `totalSupply` won't be updated here, potentially allowing them to bypass the maximum minting limit.

Suggestion:

It is recommended to remove the redundant checks and ensure the correctness of the validation logic.

Resolution:

This issue has been fixed.

FNF-4 Code Optimization

Severity: Minor

Status: Fixed

Code Location:

FanslandNFT.sol#307-312

Descriptions:

In the for loop, you can directly use `index` as the index of the `result` array without redefining a `resultIndex` variable as the index.

Suggestion:

It is recommended to delete the `resultIndex` variable and use `index` directly as the index.

Resolution:

This issue has been fixed.

FNF-5 Using the `safeMath` Library

Severity: Minor

Status: Fixed

Code Location:

FanslandNFT.sol#187-225

Descriptions:

The `Math` library is used in the contract, which requires a lot of `require` judgments and reduces the readability of the code.

Suggestion:

It is recommended to use the `safeMath` library.

Resolution:

Client cancels use of Math library.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer