alethea.ai

NFT Factory / ALI Token

SMART CONTRACT AUDIT REPORT

IMMUNEBYTES

21 March 2023

Introduction

About Alethea AI

Alethea AI is building a decentralized protocol that will enable the creation of interactive and intelligent NFTs (iNFTs). As originators of the iNFT standard, Alethea AI is on the cutting edge of embedding AI animation, interaction, and generative AI capabilities into NFTs. Anyone can use the iNFT protocol to Create, Train and Earn from their iNFTs in the world's first Intelligent Metaverse known as Noah's Ark.

Visit https://alethea.ai/ to know more about it.

About ImmuneBytes

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

The team has secured 205+ blockchain projects by providing security services on different frameworks. The ImmuneBytes team helps start-ups with detailed system analysis, ensuring security and managing the overall project.

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

Documentation Details

The team has provided the following doc for audit:

1. https://github.com/AletheaAI/alethea-contracts#readme

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.

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

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 to find 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 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	Alethea AI
Platform	EVM
Languages	Solidity
GitHub Link	https://github.com/AletheaAI/alethea-contracts
Commit - Final Audit	4c6a911d541ea54933124e9b932032a5336ea61d
Platforms & Tools	Remix IDE, Truffle, VScode, Contract Library, Slither, SmartCheck, Fuzz

Security Level References

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

CRITICAL

Issues may result in fund leakage or incorrect fund allocation.

HIGH

Issues affecting the business logic, performance, and functionality.

MEDIUM

Issues could lead to data loss or other manipulations.

LOW

Issues around smart contract code upgradability, libraries, and others.

INFORMATIONAL

Issues which can further improve the code on gas optimizations and reusability.

Issues	Critical	High	Medium	Low	Informational
Open	-	-	-	-	-
Closed	-	-	-	-	2
Acknowledged	-	-	-	1	1

Finding

#	Findings	Risk	Status
1.	RoyalERC721: Absence of Zero Address Validation	Low	Acknowledged
2.	NFTFactoryV2: External Visibility could be used	Informational	Fixed
3.	NFTFactoryV2: Coding Style Issues in the Contract	Informational	Fixed
4.	NFTFactory, WhitableNFT, BinanceAliERC20v2: Unlocked Pragma statements found in the contracts	Informational	Acknowledged

Critical Severity Issues

No issues were found.

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

1. **RoyalERC721: Absence of Zero Address Validation**

   Description: The RoyalERC721 contract includes an ownership transfer function, i.e., transferOwnership(), that changes the owner's address in the contract.

   However, during the automated testing of the contract, it was found that no Zero Address Validation is implemented before updating the owner's address. Although the function includes an access control check, it's considered a better practice in Solidity smart contracts to validate the inputs passed to a function.

   Recommendation: A require statement should be included in such functions to ensure no zero address is passed in the arguments.

   Acknowledged (February 24th, 2023): This is an intentional functionality.

Informational

1. **NFTFactoryV2: External Visibility could be used**

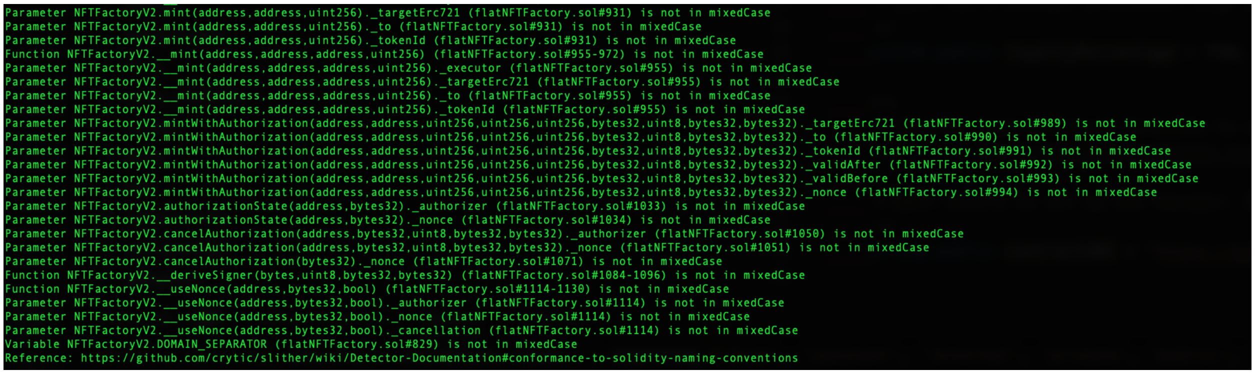
   Description: Functions that are never called within the contract should be marked as external visibility instead of public visibility.

   Recommendation: If public visibility of such functions isn’t intentional in the contract, they can be marked as external.

   Amended (March 21st, 2023): The team has fixed the issue.

2. NFTFactoryV2: Coding Style Issues in the Contract

Description: The code readability of a smart contract is primarily influenced by the Coding Style issues and in some specific scenarios may lead to bugs in the future.



```
Parameter NFTFactoryV2.mint(address,address,uint256)._targetErc721 (flatNFTFactory.sol#931) is not in mixedCase
Parameter NFTFactoryV2.mint(address,address,uint256)._to (flatNFTFactory.sol#931) is not in mixedCase
Parameter NFTFactoryV2.mint(address,address,uint256)._tokenId (flatNFTFactory.sol#931) is not in mixedCase
Function NFTFactoryV2.__mint(address,address,address,uint256) (flatNFTFactory.sol#955-972) is not in mixedCase
Parameter NFTFactoryV2.__mint(address,address,address,uint256)._executor (flatNFTFactory.sol#955) is not in mixedCase
Parameter NFTFactoryV2.__mint(address,address,address,uint256)._targetErc721 (flatNFTFactory.sol#955) is not in mixedCase
Parameter NFTFactoryV2.__mint(address,address,address,uint256)._to (flatNFTFactory.sol#955) is not in mixedCase
Parameter NFTFactoryV2.__mint(address,address,address,uint256)._tokenId (flatNFTFactory.sol#955) is not in mixedCase
Parameter NFTFactoryV2.mintWithAuthorization(address,address,uint256,uint256,uint256,bytes32,uint8,bytes32,bytes32)._targetErc721 (flatNFTFactory.sol#989) is not in mixedCase
Parameter NFTFactoryV2.mintWithAuthorization(address,address,uint256,uint256,uint256,bytes32,uint8,bytes32,bytes32)._to (flatNFTFactory.sol#990) is not in mixedCase
Parameter NFTFactoryV2.mintWithAuthorization(address,address,uint256,uint256,uint256,bytes32,uint8,bytes32,bytes32)._tokenId (flatNFTFactory.sol#991) is not in mixedCase
Parameter NFTFactoryV2.mintWithAuthorization(address,address,uint256,uint256,uint256,bytes32,uint8,bytes32,bytes32)._validAfter (flatNFTFactory.sol#992) is not in mixedCase
Parameter NFTFactoryV2.mintWithAuthorization(address,address,uint256,uint256,uint256,bytes32,uint8,bytes32,bytes32)._validBefore (flatNFTFactory.sol#993) is not in mixedCase
Parameter NFTFactoryV2.mintWithAuthorization(address,address,uint256,uint256,uint256,bytes32,uint8,bytes32,bytes32)._nonce (flatNFTFactory.sol#994) is not in mixedCase
Parameter NFTFactoryV2.authorizationState(address,bytes32)._authorizer (flatNFTFactory.sol#1033) is not in mixedCase
Parameter NFTFactoryV2.authorizationState(address,bytes32)._nonce (flatNFTFactory.sol#1034) is not in mixedCase
Parameter NFTFactoryV2.cancelAuthorization(address,bytes32,uint8,bytes32,bytes32)._authorizer (flatNFTFactory.sol#1050) is not in mixedCase
Parameter NFTFactoryV2.cancelAuthorization(address,bytes32,uint8,bytes32,bytes32)._nonce (flatNFTFactory.sol#1051) is not in mixedCase
Parameter NFTFactoryV2.cancelAuthorization(bytes32)._nonce (flatNFTFactory.sol#1071) is not in mixedCase
Function NFTFactoryV2.__deriveSigner(bytes,uint8,bytes32,bytes32) (flatNFTFactory.sol#1084-1096) is not in mixedCase
Function NFTFactoryV2.__useNonce(address,bytes32,bool) (flatNFTFactory.sol#1114-1130) is not in mixedCase
Parameter NFTFactoryV2.__useNonce(address,bytes32,bool)._authorizer (flatNFTFactory.sol#1114) is not in mixedCase
Parameter NFTFactoryV2.__useNonce(address,bytes32,bool)._nonce (flatNFTFactory.sol#1114) is not in mixedCase
Parameter NFTFactoryV2.__useNonce(address,bytes32,bool)._cancellation (flatNFTFactory.sol#1114) is not in mixedCase
Variable NFTFactoryV2.DOMAIN_SEPARATOR (flatNFTFactory.sol#829) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

During the automated testing, it was found that the NFTFactory contract had quite a few code-style issues. Please follow this link to find details on naming conventions in solidity code.

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

Amended (March 21st, 2023): The team has fixed the issue.

3. NFTFactory, WhitableNFT, BinanceAliERC20v2: Unlocked Pragma statements found in the contracts

Description: During the code review, it was found that the contracts included unlocked pragma solidity version statements.

It's not considered a better practice in Smart contract development to do so, as it might lead to accidental deployment to a version with unfixed bugs.

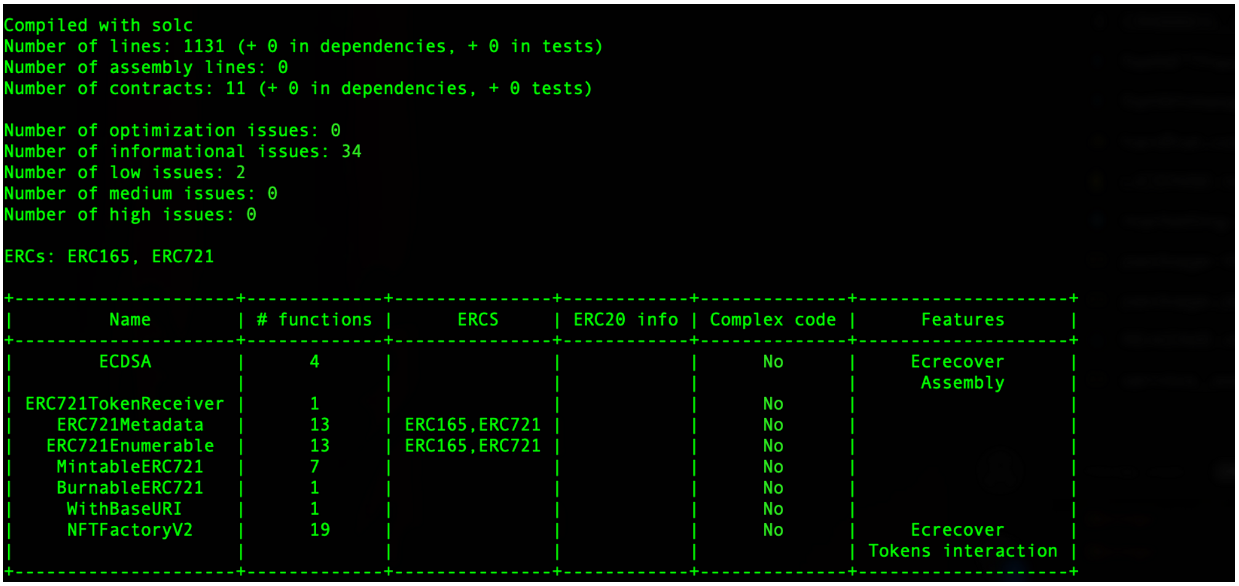
Recommendation: It's always recommended to lock pragma statements to a specific version while writing contracts.

Acknowledged (February 24th, 2023): This is an intentional functionality.

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.

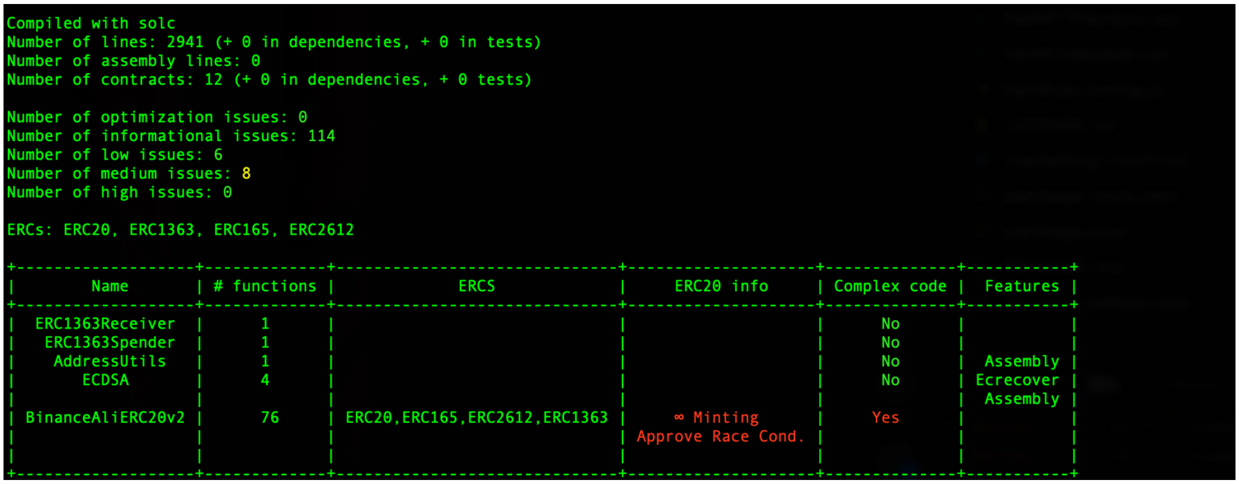
Automated Test Results

1. **NFTFactoryV2**



```
Compiled with solc
Number of lines: 1131 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 11 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 34
Number of low issues: 2
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC165, ERC721

+---------------------+-------------+---------------+------------+--------------+--------------------+
|         Name        | # functions |      ERCS     | ERC20 info | Complex code |      Features      |
+---------------------+-------------+---------------+------------+--------------+--------------------+
|        ECDSA        |      4      |               |            |      No      |     Ecrecover      |
|                     |             |               |            |              |      Assembly      |
| ERC721TokenReceiver |      1      |               |            |      No      |                    |
|    ERC721Metadata   |      13     | ERC165,ERC721 |            |      No      |                    |
|   ERC721Enumerable  |      13     | ERC165,ERC721 |            |      No      |                    |
|    MintableERC721   |      7      |               |            |      No      |                    |
|    BurnableERC721   |      1      |               |            |      No      |                    |
|     WithBaseURI     |      1      |               |            |      No      |                    |
|     NFTFactoryV2    |      19     |               |            |      No      |     Ecrecover      |
|                     |             |               |            |              | Tokens interaction |
+---------------------+-------------+---------------+------------+--------------+--------------------+
```

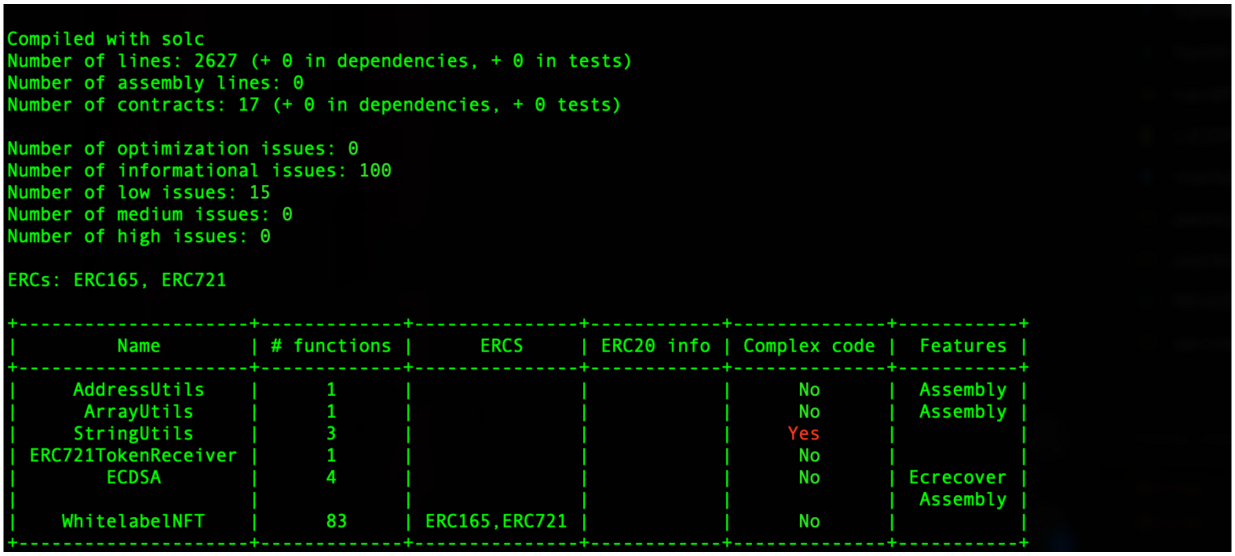
2. **BinanceAliERC20v2.sol**



```
Compiled with solc
Number of lines: 2941 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 12 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 114
Number of low issues: 6
Number of medium issues: 8
Number of high issues: 0

ERCs: ERC20, ERC1363, ERC165, ERC2612

+-------------------+-------------+------------------------------+--------------------+--------------+-----------+
|        Name       | # functions |             ERCS             |     ERC20 info     | Complex code | Features  |
+-------------------+-------------+------------------------------+--------------------+--------------+-----------+
|  ERC1363Receiver  |      1      |                              |                    |      No      |           |
|   ERC1363Spender  |      1      |                              |                    |      No      |           |
|    AddressUtils   |      1      |                              |                    |      No      |  Assembly |
|       ECDSA       |      4      |                              |                    |      No      | Ecrecover |
|                   |             |                              |                    |              |  Assembly |
| BinanceAliERC20v2 |      76     | ERC20,ERC165,ERC2612,ERC1363 |     ∞ Minting      |     Yes      |           |
|                   |             |                              | Approve Race Cond. |              |           |
|                   |             |                              |                    |              |           |
+-------------------+-------------+------------------------------+--------------------+--------------+-----------+
```

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.

3. WhiteableNFT.sol



```
Compiled with solc
Number of lines: 2627 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 17 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 100
Number of low issues: 15
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC165, ERC721

+---------------------+-------------+---------------+-----------+--------------+-----------+
|         Name        | # functions |      ERCS     | ERC20 info | Complex code |  Features |
+---------------------+-------------+---------------+-----------+--------------+-----------+
|     AddressUtils    |      1      |               |           |      No      |  Assembly |
|      ArrayUtils     |      1      |               |           |      No      |  Assembly |
|     StringUtils     |      3      |               |           |     Yes      |           |
| ERC721TokenReceiver |      1      |               |           |      No      |           |
|        ECDSA        |      4      |               |           |      No      | Ecrecover |
|                     |             |               |           |              |  Assembly |
|    WhitelabelNFT    |      83     | ERC165,ERC721 |           |      No      |           |
+---------------------+-------------+---------------+-----------+--------------+-----------+
```

Concluding Remarks

While conducting the audits of the Alethea AI - NFTFactory and ALIToken, it was observed that the contracts contain Low severity issues along with a few recommendations.

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 Alethea 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 refactoring by the team on critical issues.

IMMUNEBYTES

AUDITS