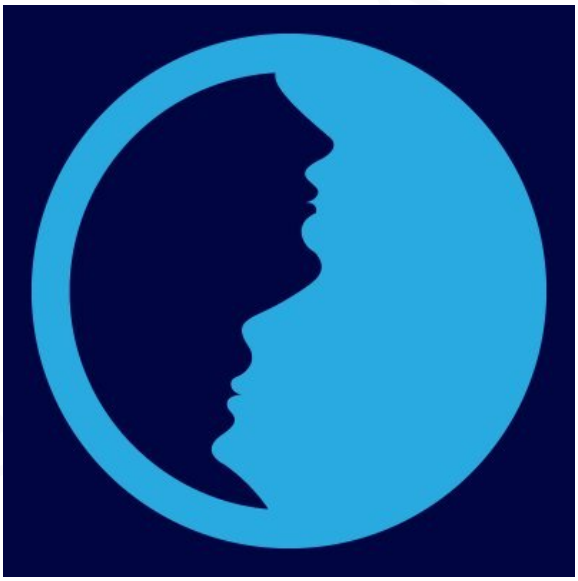
Alethea AI

Smart Contract Audit
Final Report



January 12, 2023

Introduction 3
1. About Alethea AI 3
2. About ImmuneBytes 3

Documentation Details 3

Audit Process & Methodology 4

Audit Details 4

Audit Goals 5

Security Level Reference 5

Audit Summary 6

Finding 6
Critical Severity Issues 7
High Severity Issues 7
Medium Severity Issues 7
Low Severity Issues 7
Informational 7

Automated Test Results 8
Maian 9

Concluding Remarks 10

Disclaimer 10

Introduction

1. 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.

2. 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://immunebytes.notion.site/Standard-Contract-Audit-AletheaAI-c69cc5bb7d0348fbb1def0090dfd839d

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, including -

1. Structural analysis of the smart contract is checked and verified.
2. An extensive automated testing of all the contracts under scope is conducted.
3. Line-by-line Manual Code review is conducted to evaluate, analyze and identify the potential security risks in the contract.
4. Evaluation of the contract's intended behavior and the documentation shared is imperative to verify the contract behaves as expected.
5. For complex and heavy contracts, adequate integration testing is conducted to ensure that contracts interact acceptably.
6. Storage layout verifications in the upgradeable contract are a must.
7. An important step in the audit procedure is highlighting and recommending better gas optimization techniques in the contract.

Audit Details

- Project Name: Alethea AI
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Smart Contracts: NFTFactory.sol, PolygonAliERC20v2.sol, WhitelabelNFT.sol
- GitHub Link: https://github.com/AletheaAI/alethea-contracts
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

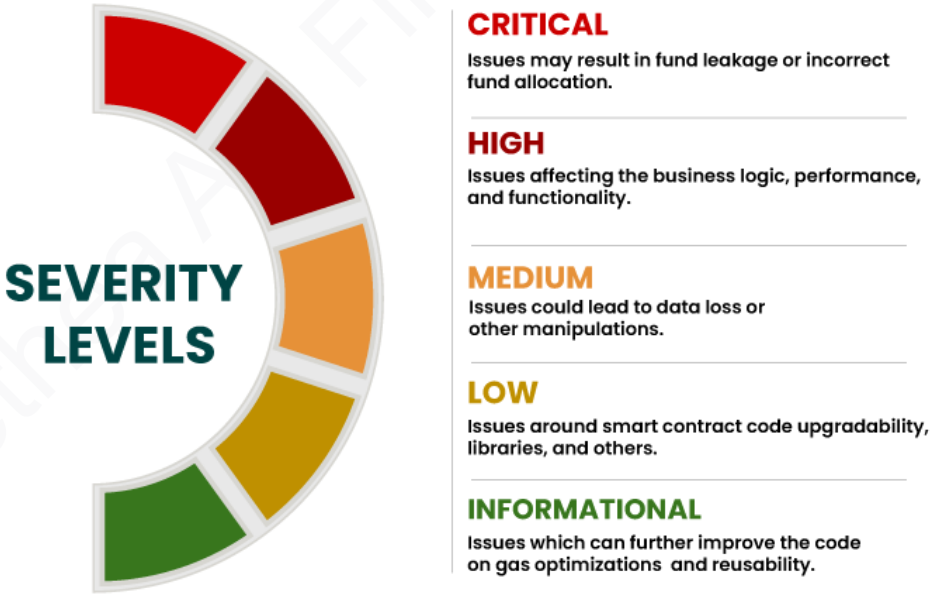
Audit Goals

The audit's focus 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 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 were assigned a severity level from the following:



Audit Summary

Team ImmuneBytes has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

Issues	Critical	High	Medium	Low
Open	-	-	-	-
Closed	-	-	-	-
Acknowledged	-	-	-	-

Finding

#	Findings	Risk	Status
1	NFTFactory, Coding Style Issues in the Contract	**Informatory**	Acknowledged
2	NFTFactory, WhitableNFT, PolygonAliERC20v2: Unlocked Pragma statements found in the contracts	**Informatory**	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
No issues were found.

Informational

1. ***NFTFactory, Coding Style Issues in the Contract***

   Description

   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 NFTFactory.authorizationState(address,bytes32)._authorizer (myFlats/flatNftFactory.sol#786) is not in mixedCase
Parameter NFTFactory.authorizationState(address,bytes32)._nonce (myFlats/flatNftFactory.sol#787) is not in mixedCase
Parameter NFTFactory.cancelAuthorization(address,bytes32,uint8,bytes32,bytes32)._authorizer (myFlats/flatNftFactory.sol#803) is not in mixedCase
Parameter NFTFactory.cancelAuthorization(address,bytes32,uint8,bytes32,bytes32)._nonce (myFlats/flatNftFactory.sol#804) is not in mixedCase
Parameter NFTFactory.cancelAuthorization(bytes32)._nonce (myFlats/flatNftFactory.sol#824) is not in mixedCase
Function NFTFactory.__deriveSigner(bytes,uint8,bytes32,bytes32) (myFlats/flatNftFactory.sol#837-849) is not in mixedCase
Function NFTFactory.__useNonce(address,bytes32,bool) (myFlats/flatNftFactory.sol#867-883) is not in mixedCase
Parameter NFTFactory.__useNonce(address,bytes32,bool)._authorizer (myFlats/flatNftFactory.sol#867) is not in mixedCase
Parameter NFTFactory.__useNonce(address,bytes32,bool)._nonce (myFlats/flatNftFactory.sol#867) is not in mixedCase
Parameter NFTFactory.__useNonce(address,bytes32,bool)._cancellation (myFlats/flatNftFactory.sol#867) is not in mixedCase
Variable NFTFactory.DOMAIN_SEPARATOR (myFlats/flatNftFactory.sol#594) is not in mixedCase
```

   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.

2. ***NFTFactory, WhitableNFT, PolygonAliERC20v2: 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.

Automated Test Results

1. NFTFactory

```
Compiled with solc
Number of lines: 884 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 6 (+ 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


+-----------------+-------------+------+------------+--------------+------------+
|      Name       | # functions | ERCS | ERC20 info | Complex code |  Features  |
+-----------------+-------------+------+------------+--------------+------------+
| MintableERC721  |      7      |      |            |      No      |            |
| BurnableERC721  |      1      |      |            |      No      |            |
|  WithBaseURI    |      1      |      |            |      No      |            |
|     ECDSA       |      4      |      |            |      No      | Ecrecover  |
|                 |             |      |            |              |  Assembly  |
|   NFTFactory    |      19     |      |            |      No      | Ecrecover  |
+-----------------+-------------+------+------------+--------------+------------+
```

2. PolygonAliERC20v2.sol

```
Compiled with solc
Number of lines: 2983 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 13 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 115
Number of low issues: 6
Number of medium issues: 13
Number of high issues: 0

ERCs: ERC2612, ERC20, ERC165, ERC1363

+-------------------+-------------+-----------------------------+--------------------+--------------+------------+
|        Name       | # functions |            ERCS             |     ERC20 info     | Complex code |  Features  |
+-------------------+-------------+-----------------------------+--------------------+--------------+------------+
|  ERC1363Receiver  |      1      |                             |                    |      No      |            |
|   ERC1363Spender  |      1      |                             |                    |      No      |            |
|    AddressUtils   |      1      |                             |                    |      No      |  Assembly  |
|       ECDSA       |      4      |                             |                    |      No      | Ecrecover  |
|                   |             |                             |                    |              |  Assembly  |
|     AliERC20v2    |      76     | ERC20,ERC165,ERC2612,ERC1363 |    ∞ Minting       |     Yes      |            |
|                   |             |                             | Approve Race Cond. |              |            |
|                   |             |                             |                    |              |            |
| PolygonAliERC20v2 |      78     | ERC20,ERC165,ERC2612,ERC1363 |    ∞ Minting       |     Yes      |            |
|                   |             |                             | Approve Race Cond. |              |            |
|                   |             |                             |                    |              |            |
+-------------------+-------------+-----------------------------+--------------------+--------------+------------+
```

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: ERC721, ERC165

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

Maian

```
root@dee4ed2f2831:/MAIAN/tool# python3 maian.py -s /share/NFTFactory_flat.sol NFTFactory -c 0

================================================================================
=====
[ ] Compiling Solidity contract from the file /share/NFTFactory_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    7222  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/NFTFactory.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052348015610010576000080fd5b506004361061012c...
[ ] Bytecode length    : 14444
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@dee4ed2f2831:/MAIAN/tool# python3 maian.py -s /share/NFTFactory_flat.sol NFTFactory -c 1

================================================================================
=====
[ ] Compiling Solidity contract from the file /share/NFTFactory_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Sending Ether to contract 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306  .. tx[0] mined  Sent!

[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    7222  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/NFTFactory.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052348015610010576000080fd5b506004361061012c...
[ ] Bytecode length    : 14444
[ ] Blockchain contract: True
[ ] Debug              : False

[ ] Search with call depth: 1   : Unknown operation  at pos 15

[ ] Search with call depth: 2   : Unknown operation  at pos 15

[ ] Search with call depth: 3   : Unknown operation  at pos 15

[+] No prodigal vulnerability found
root@dee4ed2f2831:/MAIAN/tool# python3 maian.py -s /share/NFTFactory_flat.sol NFTFactory -c 2

================================================================================
=====
[ ] Compiling Solidity contract from the file /share/NFTFactory_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    7222  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/NFTFactory.address
[ ] Check if contract is GREEDY

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052348015610010576000080fd5b506004361061012c...
[ ] Bytecode length    : 14444
[ ] Debug              : False
[-] Contract can receive Ether

[-] No lock vulnerability found because the contract cannot receive Ether
```

Concluding Remarks

While conducting the audits of the Alethea AI smart contracts, it was observed that the contracts contain no Critical, High, Medium, and Low severity issues.

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 complementing this audit is strongly recommended.

Our team does not endorse the Alethea AI platform or its product, nor is this audit investment advice.

Notes:

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