MahaDAO

Smart Contract Audit Report

Genesis.sol

May 02, 2021

Introduction

1. About MahaDAO

ARTH is a new type of currency designed to not be pegged to government-owned currencies (like US Dollar, Euro, or Chinese Yuan), but still remain relatively stable (unlike Gold and Bitcoin).

Without being influenced by government-owned currencies, ARTH will be immune to inflation. Through stability, ARTH also becomes a superior choice of currency for means of trade. This is unlike Gold or Bitcoin, which are used more as a store of value rather than a medium of exchange.

Visit http://mahadao.com/ to learn 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 15+ 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 MahaDAO team has provided documentation for the purpose of conducting the audit. The documents are:

1. https://docs.arthcoin.com/

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: **ARTH v2**
- Contract Name: Genesis.sol
- Languages: Solidity(Smart contract)
- Github commit hash for audit:**95fac3e9dca2af67c974f2f87c2c385c4bc03df2**
- GitHub link: https://github.com/MahaDAO/arthcoin-v2/blob/develop/contracts/Genesis/Genesis.sol
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, 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	1	1	2
Closed	-	-	-

High severity issues

1. **Zero Maha Token amount is minted while calling _redeemARTHAndMAHA function**
   Line no - 314
   Description:
   As per the current design of the **_redeemARTHandMAHA** function, the Genesis token is being burnt while the ARTH and MAHA token is to be minted and transferred to the user. However, the function does not seem to be adequate as it mints ZERO MAHA tokens every time it is called. This will lead to an unwanted scenario where no MAHA Token is ever minted.

```
306     function _redeemARTHAndMAHA(uint256 amount) internal hasEnded {
307         require(balanceOf(msg.sender) >= amount, 'Genesis: balance < ar
308
309         _burn(msg.sender, amount);
310         _ARTH.poolMint(msg.sender, amount);
311
312         // TODO: distribute MAHA.
313         // HOW?
314         uint256 mahaAmount = 0;
315
316         // NOTE: need to be given and revoked MINTER ROLE accordingly.
317         MAHA.mint(msg.sender, mahaAmount);
```

   Recommendation:
   The distribution mechanism of MAHA tokens in the **redeemARTHAndMAHA function**, should be implemented in the function body to avoid ZERO tokens being minted.

Medium severity issues

1. Modifier hasStarted never used in the Genesis Contract
   Line no - 70
   Description:
   The Genesis contract includes the **hasStarted** modifier at Line 70 but never uses it throughout the contract.

```
70      modifier hasStarted() {
71          require(block.timestamp >= startTime, 'Genesis: not started');
72          _;
73      }
```

   While this consumes additional space in the contract, it also adversely affects the gas optimization as well as the readability of the smart contract code.
   Recommendation:

Adequate use of all State Variable, modifiers, mappings etc must be ensured in the contract. If the **hasStarted** modifier holds no significance it should be removed from the contract.

Low severity issues

1. **Absence of Zero Address Validation**
   Line no- 124, 139, 146, 153
   Description:
   The Genesis Contract includes quite a few functions that updates some of the imperative addresses in the contract like **arthWETHPoolAddres, arthETHPairAddress** etc. However, during the automated testing of the contact it was found that no Zero Address Validation is implemented on the following functions while updating the address state variables of the contract:
   - **setPoolAndPairs**
   - **setARTHWETHPoolAddress**
   - **setARTHETHPairAddress**
   - **setARTHXETHPairAddress**

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

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

   Therefore, the following function must be marked as external within the contract:
   - **mint #Line 178**
   - **redeem #Line 194**
   - **distribute #Line 203**
   - **getIsRaisedBetweenCaps #Line 219**

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

Recommendations

1. Coding Style Issues in the Contract
   Description:
   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 Genesis contract had quite a few code style issues.

```
Parameter Genesis.setDuration(uint256)._duration (contracts/Genesis/FLAT_Genesis.sol#1592) is not in mixedCase
Parameter Genesis.setPoolAndPairs(address,address,address)._arthETHPool (contracts/Genesis/FLAT_Genesis.sol#1597) is not in mixedCase
Parameter Genesis.setPoolAndPairs(address,address,address)._arthETHPair (contracts/Genesis/FLAT_Genesis.sol#1598) is not in mixedCase
Parameter Genesis.setPoolAndPairs(address,address,address)._arthxETHPair (contracts/Genesis/FLAT_Genesis.sol#1599) is not in mixedCase
Parameter Genesis.setCaps(uint256,uint256)._softCap (contracts/Genesis/FLAT_Genesis.sol#1606) is not in mixedCase
Parameter Genesis.setCaps(uint256,uint256)._hardCap (contracts/Genesis/FLAT_Genesis.sol#1606) is not in mixedCase
Variable Genesis._WETH (contracts/Genesis/FLAT_Genesis.sol#1496) is not in mixedCase
Variable Genesis._ARTH (contracts/Genesis/FLAT_Genesis.sol#1497) is not in mixedCase
Variable Genesis._ARTHX (contracts/Genesis/FLAT_Genesis.sol#1498) is not in mixedCase
Variable Genesis._CURVE (contracts/Genesis/FLAT_Genesis.sol#1499) is not in mixedCase
Variable Genesis._MAHA (contracts/Genesis/FLAT_Genesis.sol#1500) is not in mixedCase
Variable Genesis._ROUTER (contracts/Genesis/FLAT_Genesis.sol#1501) is not in mixedCase
```

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

2. NatSpec Annotations must be included
   Description:
   The smart contracts do not include the NatSpec annotations adequately.

   Recommendation:
   Cover by NatSpec all Contract methods.

IMMUNE BYTES

Automated Test Result

```
Genesis.setPoolAndPairs(address,address,address)._arthETHPool (contracts/Genesis/FLAT_Genesis.sol#1597) lacks a zero-check on :
        - arthWETHPoolAddres = _arthETHPool (contracts/Genesis/FLAT_Genesis.sol#1601)
Genesis.setPoolAndPairs(address,address,address)._arthETHPair (contracts/Genesis/FLAT_Genesis.sol#1598) lacks a zero-check on :
        - arthETHPairAddress = _arthETHPair (contracts/Genesis/FLAT_Genesis.sol#1602)
Genesis.setPoolAndPairs(address,address,address)._arthxETHPair (contracts/Genesis/FLAT_Genesis.sol#1599) lacks a zero-check on
        - arthxETHPairAddress = _arthxETHPair (contracts/Genesis/FLAT_Genesis.sol#1603)
Genesis.setARTHWETHPoolAddress(address).poolAddress (contracts/Genesis/FLAT_Genesis.sol#1611) lacks a zero-check on :
        - arthWETHPoolAddres = poolAddress (contracts/Genesis/FLAT_Genesis.sol#1615)
Genesis.setARTHETHPairAddress(address).pairAddress (contracts/Genesis/FLAT_Genesis.sol#1618) lacks a zero-check on :
        - arthETHPairAddress = pairAddress (contracts/Genesis/FLAT_Genesis.sol#1622)
Genesis.setARTHXETHPairAddress(address).pairAddress (contracts/Genesis/FLAT_Genesis.sol#1625) lacks a zero-check on :
        - arthxETHPairAddress = pairAddress (contracts/Genesis/FLAT_Genesis.sol#1629)
```

```
Parameter Genesis.setDuration(uint256)._duration (contracts/Genesis/FLAT_Genesis.sol#1592) is not in mixedCase
Parameter Genesis.setPoolAndPairs(address,address,address)._arthETHPool (contracts/Genesis/FLAT_Genesis.sol#1597) is not in mixedCase
Parameter Genesis.setPoolAndPairs(address,address,address)._arthETHPair (contracts/Genesis/FLAT_Genesis.sol#1598) is not in mixedCase
Parameter Genesis.setPoolAndPairs(address,address,address)._arthxETHPair (contracts/Genesis/FLAT_Genesis.sol#1599) is not in mixedCase
Parameter Genesis.setCaps(uint256,uint256)._softCap (contracts/Genesis/FLAT_Genesis.sol#1606) is not in mixedCase
Parameter Genesis.setCaps(uint256,uint256)._hardCap (contracts/Genesis/FLAT_Genesis.sol#1606) is not in mixedCase
Variable Genesis._WETH (contracts/Genesis/FLAT_Genesis.sol#1496) is not in mixedCase
Variable Genesis._ARTH (contracts/Genesis/FLAT_Genesis.sol#1497) is not in mixedCase
Variable Genesis._ARTHX (contracts/Genesis/FLAT_Genesis.sol#1498) is not in mixedCase
```

```
mint(uint256) should be declared external:
        - Genesis.mint(uint256) (contracts/Genesis/FLAT_Genesis.sol#1650-1664)
redeem(uint256) should be declared external:
        - Genesis.redeem(uint256) (contracts/Genesis/FLAT_Genesis.sol#1666-1673)
distribute() should be declared external:
        - Genesis.distribute() (contracts/Genesis/FLAT_Genesis.sol#1675-1685)
getIsRaisedBelowSoftCap() should be declared external:
        - Genesis.getIsRaisedBelowSoftCap() (contracts/Genesis/FLAT_Genesis.sol#1687-1689)
getIsRaisedBetweenCaps() should be declared external:
        - Genesis.getIsRaisedBetweenCaps() (contracts/Genesis/FLAT_Genesis.sol#1691-1694)
getPercentRaised() should be declared external:
        - Genesis.getPercentRaised() (contracts/Genesis/FLAT_Genesis.sol#1696-1698)
```

Concluding Remarks

While conducting the audits of MahaDAO smart contract - Genesis.sol, it was observed that the contracts contain High, Medium and Low severity issues, along with a few areas of recommendations.

Our auditors suggest that High, Medium and Low severity issues should be resolved by MahaDAO developers. Resolving the areas of recommendations are up to the team's discretion. 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 MahaDAO platform or its product neither 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 Pvt Ltd.