MahaDAO

MahaToken

Smart Contract Audit Report

January 09, 2023

IMMUNE BYTES

Introduction

1. About MahaDAO

MahaDAO is a mission to create a decentralized and stable economy. that is driven by the people, for the people.

MahaDAO is a community-powered, decentralized organization that aims to empower billions with a stable economy through the world's first value coin, ARTH.

To do this, MahaDAO uses two tokens to achieve this vision - the governance token MAHA and the value coin ARTH.

Visit http://mahadao.com/ to learn more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up to 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 MahaDAO team has provided the following doc for audit:

1. https://docs.mahadao.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 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

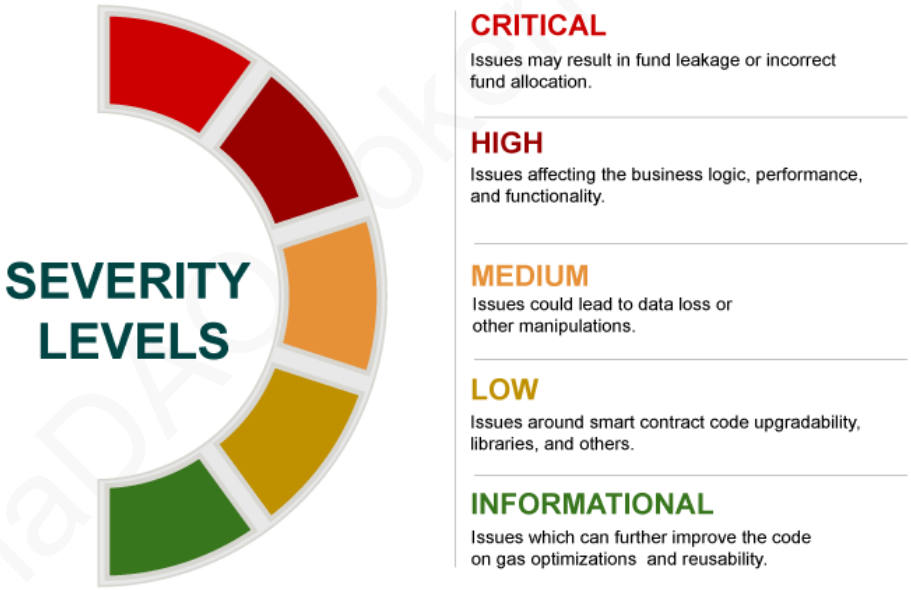
- Project Name: MahaDAO
- Contracts Name: MahaToken.sol
- Languages: Solidity(Smart contract)
- Smart Contract Address: https://etherscan.io/address/0x745407c86df8db893011912d3ab28e68b62e49b0#code
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, 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 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



IMMUNE BYTES

Audit Summary

Team ImmuneBytes have 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	-	-	1	1
Closed	-	-	-	-

Finding

#	Findings	Risk	Status
1	**setNameSymbol() tried updating private state variables of Parent contract. Leads to compilation error**	Medium	Pending
2	**Unlocked Pragma statements found in the contracts**	Low	Pending
3	**Use of large digits can be avoided**	Informational	Pending

Critical Severity Issues

No issues were found.

High Severity Issues

No issues were found.

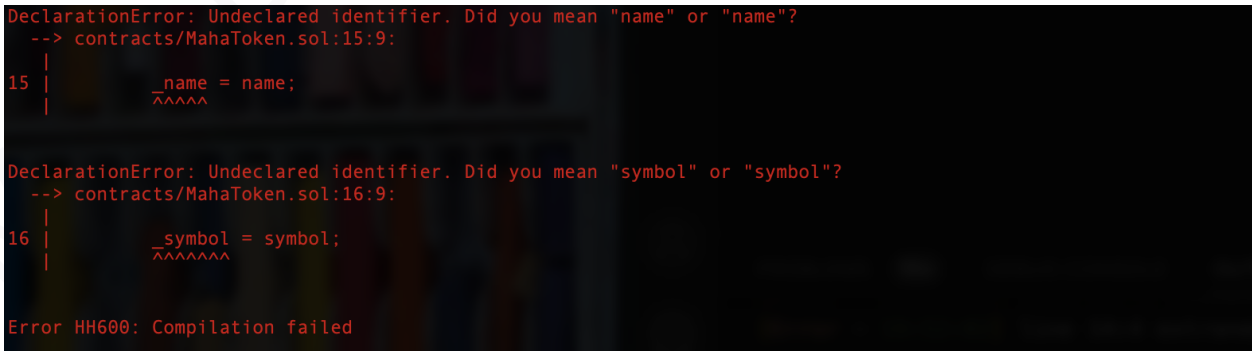
Medium severity issues

1. **setNameSymbol() tried updating private state variables of Parent contract. Leads to compilation error**
   Line no: 14-17

   Description
   The setNameSymbol() function tries to update the _name & _symobol state variable of the ERC20 contract.

   However, MahaToken is a derived contract that inherits the features of the ERC20 token contract. Therefore, as per the rules of visibility keywords in solidity, any private state variable shall only be accessible for the base contract and not the derived contract. This also leads to an Undeclared Identifier error during the compilation of the contract.

   function setNameSymbol(string memory name, string memory symbol) external onlyRole(DEFAULT_ADMIN_ROLE) {
   _name = name;
   _symbol = symbol;
   }



Recommendation:
The name and symbol of the token are already initialized in the constructor of the MahaToken contract while initializing the ERC20PresetMinterPauser("MahaDAO", "MAHA") contract.

Therefore, it's recommended to modify the current implementation of the contract to avoid compilation warnings as well as the violation of visibility rules of solidity.

Low severity issues

1. **Unlocked Pragma statements found in the contracts**
   Line no: 2

   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.

Informational

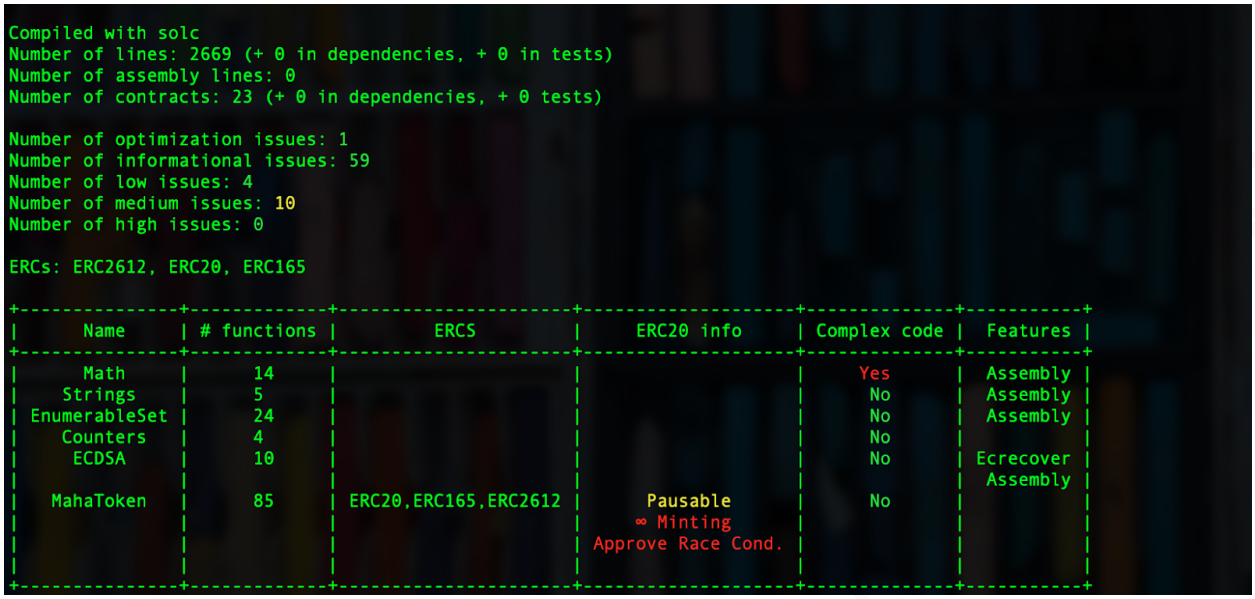
1. **Unlocked Pragma statements found in the contracts**
   Line no: 2

   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 Audit Result



```
Compiled with solc
Number of lines: 2669 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 23 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 1
Number of informational issues: 59
Number of low issues: 4
Number of medium issues: 10
Number of high issues: 0

ERCs: ERC2612, ERC20, ERC165

+---------------+-------------+----------------------+--------------------+--------------+-----------+
|      Name     | # functions |         ERCS         |     ERC20 info     | Complex code |  Features |
+---------------+-------------+----------------------+--------------------+--------------+-----------+
|      Math     |      14     |                      |                    |     Yes      |  Assembly |
|    Strings    |      5      |                      |                    |      No      |  Assembly |
| EnumerableSet |      24     |                      |                    |      No      |  Assembly |
|    Counters   |      4      |                      |                    |      No      |           |
|     ECDSA     |      10     |                      |                    |      No      | Ecrecover |
|               |             |                      |                    |              |  Assembly |
|   MahaToken   |      85     | ERC20,ERC165,ERC2612 |      Pausable      |      No      |           |
|               |             |                      |     ∞ Minting      |              |           |
|               |             |                      | Approve Race Cond. |              |           |
|               |             |                      |                    |              |           |
+---------------+-------------+----------------------+--------------------+--------------+-----------+
```

Concluding Remarks

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

Our auditors suggest that Medium and Low severity issues should be resolved by MahaDAO 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 MahaDAO 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