QuillAudits

Audit Report
January, 2022

For

MONNFTS

Contents

Overview

During the Period of January 19, 2022 to January 31,2022 - QuillAudits team Performed a security audit for MonToken Smart Contract.

MONToken

MONToken is a BEP-20 compliant utility and governance token that is used in the MonsterNFT's Ecosystem

Scope of the Audit

The scope of this audit was to analyze MONToken smart contract's codebase for quality, security, and correctness.

MONToken Contracts: https://github.com/monnfts/contracts/blob/master/contracts/MONToken.sol

Branch: Master

Commit: 523d9eb3cbbd263b86220f325935d441bacb1912

https://bscscan.com/address/0x5E4f0f6FAC92E76E41a0d043DF565fe8fBFc1De3#code

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- BEP20 transfer() does not return boolean
- BEP20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of the smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of BEP20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Mythril, Slither, SmartCheck, Surya, Solhint, MythX.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	1
Closed	0	0	0	3

Functional Testing Results

Complete functional testing report has been attached below: Visit Link

- should be able to mint by owner role PASS
- should be able to burn by owner role PASS
- should follow BEP-20 specification PASS

Issues Found

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

- Floating pragma ^0.8.0. It is best practice to lock the pragma. Consider using version 0.8.4

  Status: Acknowledged

- Centralization risk due to the owner. Owner is critical to distributing tokens and if the assigned owner decides to be malicious this will harm the intended functioning. Consider a Multisig for the ownership of the token

  Status: Fixed
  The owner has now been replaced with a Gnosis Safe, with a threshold of 3 out of 3 owners

- **Critical address change**

  MONToken, which has ownership due to Ownable, may have ownership lost if a two-step process is not followed. Changing critical addresses in contracts should be a two-step process where the first transaction (from the old/current address) registers the new address (i.e. grants ownership) and the second transaction (from the new address) replaces the old address with the new one (i.e. claims ownership). This gives an opportunity to recover from incorrect addresses mistakenly used in the first step. If not, contract functionality might become inaccessible. Consider using Claimable contracts from OpenZeppelin if Multi-sig is not used.

  Status: Fixed
  The owner has now been replaced with a Gnosis Safe, with a threshold of 3 out of 3 owners

- **Renounce Ownership**

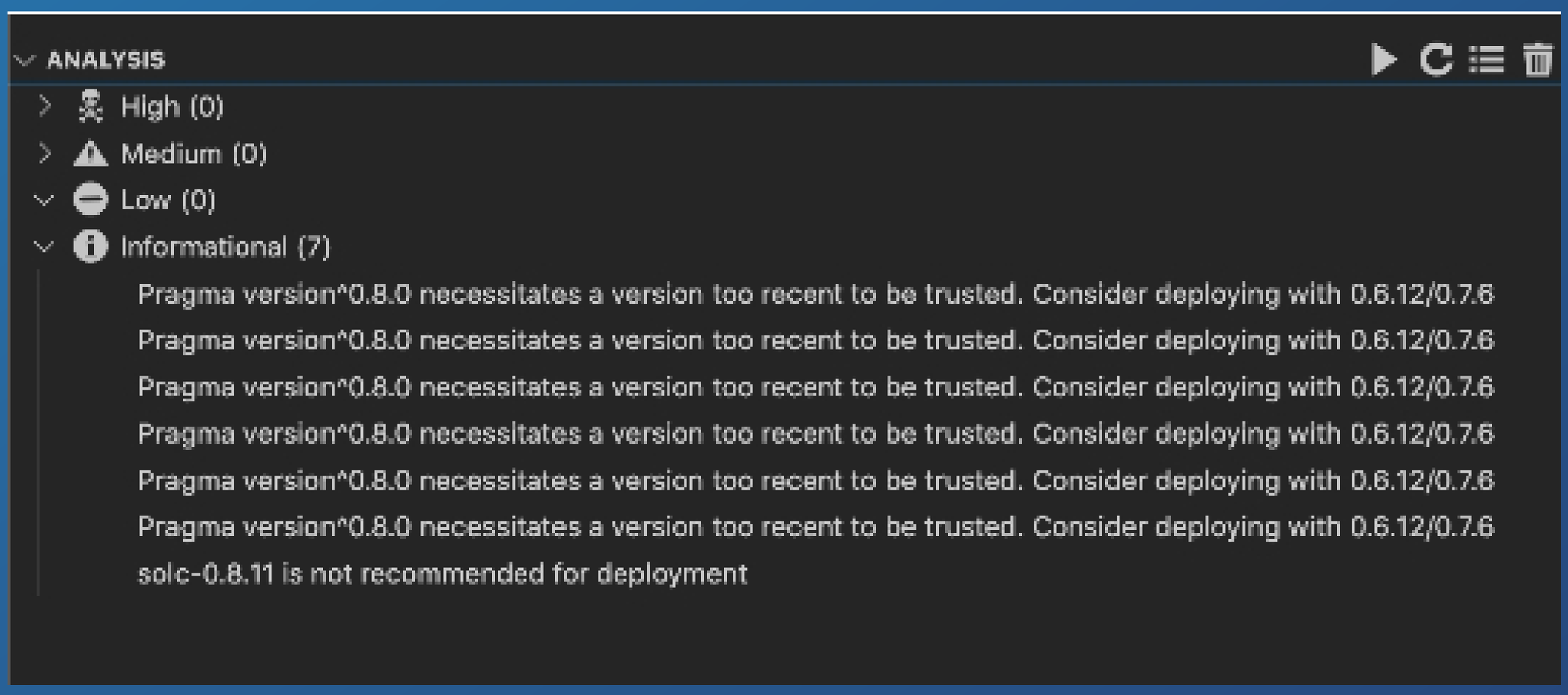
  Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The renounceOwnership function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

  It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the renounceOwnership method for two or more users should be confirmed. Alternatively, the Renounce Ownership functionality can be disabled by overriding it.

  Status: Fixed
  The owner has now been replaced with a Gnosis Safe, with a threshold of 3 out of 3 owners

Automation Tests

Slither Analysis Results



MythX Analysis Results

MythX

Analysis 1cb3becd-af67-4ce0-a5dd-4d9f330d493a MythX

Started	Wed Jan 19 2022 16:34:16 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jan 19 2022 16:36:24 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Token.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

/contracts/token.sol

Locations

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
```

Closing Summary

No issues were found. Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of MONToken. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the MONToken team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
January, 2022

For

MONNFTS

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com