QuillAudits

Audit Report
June, 2023

For

MONWU

Table of Content

Executive Summary

Project Name	Monwu
Overview	Monwu contracts encompasses an ERC20 token creation designed to distribute these tokens to different department vesting contract addresses in accordance to their allocations. The token has a maximum cap of 1,000,000,000 and allows token holders to burn their tokens until it reaches its minimal cap of 500,000,000. Both the Development and Marketing vesting contracts share similar features which allows vesting to last for 2 years and the releasable tokens will be half its total allocation for each year. While founders and private sale have a one year cliff period where allocated tokens get locked until cliff ends and vesting follows. Releasable tokens are alloted to 5 phases. Unclaimed private investors tokens are given a grace of one year after vesting period before the owner burns the tokens.
Timeline	31st May, 2023 - 23th June, 2023
Method	Manual Review, Functional Testing, Automated Testing etc.
Language	Solidity
Blockchain	Ethereum
Scope of Audit	The scope of this audit was to analyze the Monwu codebase for quality, security, and correctness. *https://github.com/BlackH3art/monwu-token*
Branch	Main
Commit	be46d069eab73004c1d5c10fb5d02ff94f93ea2c
Contracts in Scope	TokenMonwu.sol MonwuDevelopmentVesting.sol MonwuMarketingVesting.sol MonwuFoundersVesting.sol MonwuPrivateSaleVesting.sol MonwuPrivateSaleWhitelist.sol
Fixed in	ae62801d024af428d8857cde2fe8ab1346834893

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	2	2

Types of Severities

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.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- 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 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 ERC-20 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

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Common Issues

High Severity Issues

No issues found

Medium Severity Issues

No issues found

Low Severity Issues

A.1 Missing Event Emission for Significant Actions

Contracts Affected - MonwuPrivateSaleVesting.sol, MonwuPrivateSaleWhitelist.sol

Description

Whenever certain significant privileged actions are performed within the contract, it is recommended to emit an event about it. Significant actions such as releasing tokens, adding and removing addresses from whitelist and also withdrawing ethers from the contract are crucial hence it is recommended to emit an event. Although, this won't be the case for functions that make external functions with emitted events.

In MonwuPrivateSaleVesting.sol:

- withdrawEther
- buyPrivateSaleMonwu

In MonwuPrivateSaleWhitelist.sol:

- addToWhitelist
- editWhitelististedInvestor
- removeFromWhitelist

Remediation

Consider emitting an event whenever certain significant changes are made in the contracts.

Status

Resolved

Informational Issues

A.2 Floating Solidity Version (pragma solidity ^0.8.10)

Description

Contract has a floating solidity pragma version. This is present also in inherited contracts. Locking the pragma helps to ensure that the contract does not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. The recent solidity pragma version also possesses its own unique bugs.

Remediation

Making the contract use a stable solidity pragma version prevents bugs occurrence that could be ushered in by prospective versions. It is recommended, therefore, to use a fixed solidity pragma version while deploying to avoid deployment with versions that could expose the contract to attack.

Status

Resolved

A.3 Absence of Proper Code Comment in All Contracts

Description

Proper code comments explain the purpose of functions in the contracts and also about the kind of parameters a function expects as an input value. This is why it is recommended that contracts should have a NATSPEC code format to give a comprehensive meaning to the functions. While there are few comments that classify the data type, it is not substantial.

Remediation

Add a proper natspec comment format across the contracts.

Reference

Status

Resolved

B. MonwuPrivateSaleVesting.sol

High Severity Issues

No issues found

Medium Severity Issues

B.1 Reverts when Private Sale Owner Attempts to Withdraw an Amount Less Than Contract Ether Balance - POC

```
ftrace | funcSig
function withdrawEther(uint256 amount⬆) external onlyOwner {
    //@audit-issue amount should be less than or equal to the available
    balance in the contract.
    require(amount⬆ >= address(this).balance, "Not enough ether");

    (bool success,) = owner().call{value: amount⬆}("");
    require(success, "Transfer failed");
}
```

Description

When private whitelisted investors buy their allocated tokens, there is a withdrawal function created for the contract owner to withdraw these ethers. However, there is a logic error with the first required check which will prevent withdrawing less than what is in the contract. While this will still allow the contract owner to withdraw when the owner passes an input amount which is equal to the contract balance, this will hamper appropriate control flow.

Remediation

To achieve better flexibility which allows contract owner to withdraw less than or equal to the balance of ethers in the contract, we recommend

Status

Resolved

Low Severity Issues

No issues found

Informational Issues

No issues found

C. General Recommendation

All common issues highlighted in the common section are issues that cut across two or more contracts in the audit scope. Floating solidity version and inadequate code comment are common issues in all contracts and recommended to be addressed. It is important to also remediate the issue of hampered withdrawal control flow.

Functional Tests

Some of the tests performed are mentioned below

- Should create tokens, transfer ownership to the contract owner, and make distribution of allocated tokens possible
- Should allow only whitelist owner add, edit and remove addresses to the whitelist
- Should allow investors to claim their allocations when malicious owner removes them during vesting period
- Should make releasable tokens for individual allocations phased into 5 unlocks
- Should allow releasable tokens greater than zero after the cliff period is over
- Should make investors withdraw their allocated tokens in fractions - 20% of each 5 unlocks
- Should allow the contract owner to burn the left over tokens not claimed by an investor when burn time reaches
- Should allow the contract owner to withdraw exact amount in the contract

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Summary

In this report, we have considered the security of Monwu. We performed our audit according to the procedure described above.

Some issues of medium, low and informational severity were found. Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Monwu Platform. 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 Monwu Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

850+
Audits Completed

$16B
Secured

800K
Lines of Code Audited

Follow Our Journey

Audit Report
June, 2023

For

MONWU

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com