QuillAudits

Audit Report
July, 2022

For

MrwebFi

Table of Content

Executive Summary

Project Name	Mrweb Finance Audit Report
Overview	The scope of this audit covers two contracts, namely, MFTERC20.sol and TokensRecoverable.sol. MFTERC20 is the base and token contract that allows for the minting and burning of tokens, with an integration of snapshot to help query past balances of token. TokensRoverable.sol aids in the recovery of tokens and native currency of the contract by the owner.
Timeline	30 June, 2022 - 5 July, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze MFTERC20.sol and TokenRecoverable.sol. *https://testnet.bscscan.com/address/0x313F15c8019a3d523B636eC9Aea1899001971531*
Mainnet Address	*https://bscscan.com/address/0xa77d560e34bd6a8d7265f754b4fcd65d9a8e5619#code*
Fixed In	*https://github.com/huckhelp/ama_v2*
Commit Hash	17d2ebe7235bda83a0013609be5dd9e005dd1981

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	1	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
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility leve

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 Analysis

A. Contract - MFTERC20.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

No issues were found

Informational Issues

A.1 Absence of comments

Description

Contract has no proper comments that details the functionalities of the functions and the motive behind them. This is relevant to help users and developers interacting with the contract to understand the contract properly.

Remediation

Contract should be properly commented. The standard Natspec comment format will aid the structure of the contract and aid better understanding of contract logic.

Status

Fixed

B. Contract - TokensRecoverable.sol

High Severity Issues

No issues were found

Medium Severity Issues

No issues were found

Low Severity Issues

B.1 Transfer of Ownership Must be a Two-way process

Description

Contracts are integrated with the standard Openzeppelin ownable contract, however, when the owner mistakenly transfers ownership to an incorrect address, ownership is completely removed from the original owner and cannot be reverted. The transferOwnership() function in the ownable contract allows the current owner to transfer his privileges to another address. However, inside transferOwnership() , the newOwner is directly stored in the storage, owner, after validating the newOwner is a non-zero address, which may not be enough.

Remediation

It would be much safer if the transition is managed by implementing a two-step approach: _transferOwnership() and _updateOwnership() . Specifically, the _transferOwnership () function keeps the new address in the storage, _newOwner , instead of modifying the _owner() directly. The updateOwnership() function checks whether _newOwner is msg.sender, which means _newOwner signs the transaction and verifies himself as the new owner. After that, _newOwner could be set into _owner.

Status

Fixed

Informational Issues

B.2 Absence of comments

Description

Proper code commenting is required to help users understand the principal reason behind the withdrawal of tokens and BNB from the contracts.

Remediation

Contract should be properly commented. The standard Natspec comment format will aid the structure of the contract and aid users better understand the logic for withdrawal.

Status

Fixed

Functional Testing

Some of the tests performed are mentioned below

- Should get the name of the token
- Should get the symbol of the token
- Should get the owner of the contract
- Should get the decimal of the token
- Should get the total supply minted to owner
- Should transfer tokens between accounts
- Should fail if sender doesn’t have enough tokens
- Should update balances after transfers
- Should recoverTokens by the owner
- Should revert if token address is invalid
- Should recoverBNB by the owner
- Should revert if the caller of recoverTokens is not owner
- Should revert if the caller of recoverBNB is not owner

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.

Important Note To The Users

Users who mistakenly send other tokens into the contract, and these funds get stuck, can recover them with the help of the contract owner.

Closing Summary

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

Some issues of Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the end, Mrweb Finance team resolved all issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Mrweb Finance 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 Mrweb Finance 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.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
July, 2022

For

MrwebFi

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com