MicroMachines

Smart Contract Security Audit

Date 15/June/2021

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and ApeAudits and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (ApeAudits) owe no duty of care towards you or any other person, nor does ApeAudits make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and ApeAudits hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, ApeAudits hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against ApeAudits, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Purpose

This document details ApeAudits' findings and recommended solutions. This audit was performed on June 15, 2021.

At the time of our audit, ownership of the contract had been renounced.

Token Name	MicroMachines
Token Symbol	MMAC
Contract Address	0x250996383c16Aa22275FA9e64570ac3c4f78DEb2
Etherscan Link	https://bscscan.com/address/0x250996383c16aa22275fa9e64570ac3c4f78deb2
Ownership Renounce Transaction	https://bscscan.com/tx/0xf19fc0c840c1cfb170a78c201e1a70b3ceb43532a6fbd3779d56bc472885c142

Findings

NO.	Audit Items	Audit Subclass	Audit Subclass Result
1	Overflow Audit	N/A	Passed
2	Race Conditions Audit	N/A	Passed
3	Authority Control Audit	Permission Vulnerability Audit Excessive Auditing Authority	Passed Passed
4	Safe Design Audit	Zeppelin Module Safe Compiler Version Hard-coded Version Fallback Function Safeuse Show Coding Security Function Return Value Security Call Function Security	Passed Passed Passed Passed Passed Passed Passed
5	Denial of Service Audit	N/A	Passed
6	Gas Optimization Audit	N/A	Passed
7	Design Logic Audit	N/A	Passed
8	Malicious Event Log Audit	N/A	Passed
9	"False Deposit" Vulnerability Audit	N/A	Passed
10	Uninitialized Storage Pointers Audit	N/A	Passed

11	Arithmetic Accuracy Deviation Audit	N/A	Passed

MicroMachines.sol

High Severity Issues

- None.

Medium Severity Issues

- None.

Low Severity Issues

- **Issue:**
  Function unlock() can be used to reclaim ownership after contract has been renounced by calling in the following order: lock() -> renounceOwnership() -> unlock().

  Recommendation:
  Contract ownership was renounced properly and thus this is a non-issue for the contract in its current state.

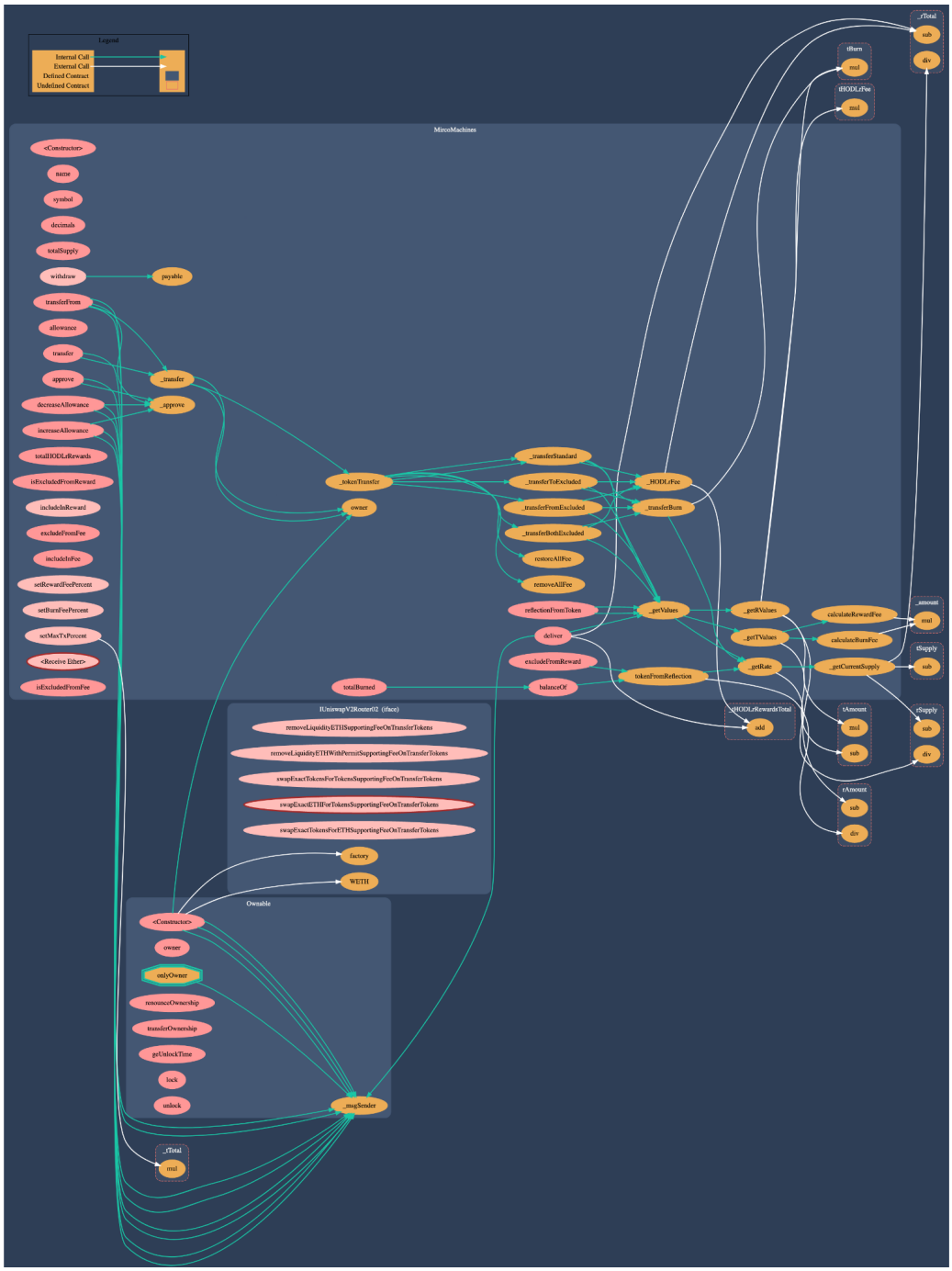
- **Issue:**
  There is no liquidity generation mechanism, which could hamper growth in the long-term.

  Recommendation:
  It is too late to do anything about this, and this is not a security issue.

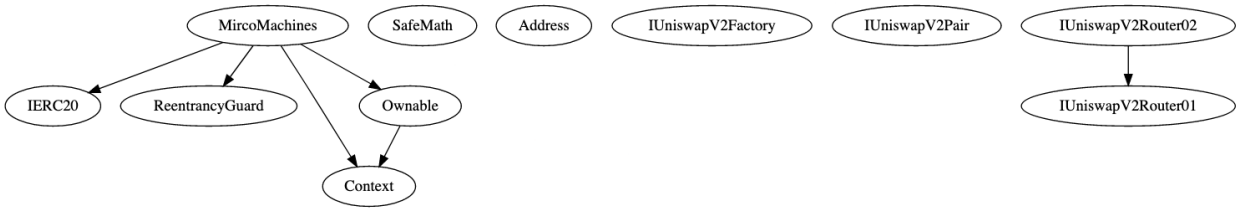
Conclusion

No severe issues found. Functions unlock() and renounceOwnership() have a bug that would allow ownership to be reclaimed, but it is impossible for the owner to exploit this because ownership has already been renounced. The token lacks liquidity generation, which could possibly hurt long-term growth.

Appendix A - Call Graph



Appendix B - Inheritance Graph



Appendix C - Function Information

+ [Int] IERC20
- **[Ext] totalSupply**
- **[Ext] balanceOf**
- **[Ext] transfer #**
- **[Ext] allowance**
- **[Ext] approve #**
- **[Ext] transferFrom #**

+ ReentrancyGuard
- **[Pub] <Constructor> #**

+ [Lib] SafeMath
- **[Int] add**
- **[Int] sub**
- **[Int] sub**
- **[Int] mul**
- **[Int] div**
- **[Int] div**
- **[Int] mod**
- **[Int] mod**

+ Context
- **[Int] _msgSender**
- **[Int] _msgData**

+ [Lib] Address
- **[Int] isContract**
- **[Int] sendValue #**
- **[Int] functionCall #**
- **[Int] functionCall #**
- **[Int] functionCallWithValue #**
- **[Int] functionCallWithValue #**
- **[Prv] _functionCallWithValue #**

+ Ownable (Context)
- **[Pub] <Constructor> #**
- **[Pub] owner**
- **[Pub] renounceOwnership #**
- **modifiers: onlyOwner**
- **[Pub] transferOwnership #**
- **modifiers: onlyOwner**
- **[Pub] geUnlockTime**
- **[Pub] lock #**
- **modifiers: onlyOwner**
- **[Pub] unlock #**

+ [Int] IUniswapV2Factory
- **[Ext] feeTo**
- **[Ext] feeToSetter**
- **[Ext] getPair**
- **[Ext] allPairs**
- **[Ext] allPairsLength**

- [Ext] createPair #
- [Ext] setFeeTo #
- [Ext] setFeeToSetter #

+ [Int] IUniswapV2Pair
  - [Ext] name
  - [Ext] symbol
  - [Ext] decimals
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] allowance
  - [Ext] approve #
  - [Ext] transfer #
  - [Ext] transferFrom #
  - [Ext] DOMAIN_SEPARATOR
  - [Ext] PERMIT_TYPEHASH
  - [Ext] nonces
  - [Ext] permit #
  - [Ext] MINIMUM_LIQUIDITY
  - [Ext] factory
  - [Ext] token0
  - [Ext] token1
  - [Ext] getReserves
  - [Ext] price0CumulativeLast
  - [Ext] price1CumulativeLast
  - [Ext] kLast
  - [Ext] mint #
  - [Ext] burn #
  - [Ext] swap #
  - [Ext] skim #
  - [Ext] sync #
  - [Ext] initialize #

+ [Int] IUniswapV2Router01
  - [Ext] factory
  - [Ext] WETH
  - [Ext] addLiquidity #
  - [Ext] addLiquidityETH ($)
  - [Ext] removeLiquidity #
  - [Ext] removeLiquidityETH #
  - [Ext] removeLiquidityWithPermit #
  - [Ext] removeLiquidityETHWithPermit #
  - [Ext] swapExactTokensForTokens #
  - [Ext] swapTokensForExactTokens #
  - [Ext] swapExactETHForTokens ($)
  - [Ext] swapTokensForExactETH #
  - [Ext] swapExactTokensForETH #
  - [Ext] swapETHForExactTokens ($)
  - [Ext] quote
  - [Ext] getAmountOut
  - [Ext] getAmountIn
  - [Ext] getAmountsOut
  - [Ext] getAmountsIn

+ [Int] IUniswapV2Router02 (IUniswapV2Router01)

- [Ext] removeLiquidityETHSupportingFeeOnTransferTokens #
- [Ext] removeLiquidityETHWithPermitSupportingFeeOnTransferTokens #
- [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
- [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens ($)
- [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

+ MircoMachines (Context, IERC20, Ownable, ReentrancyGuard)
  - [Pub] <Constructor> #
  - [Pub] name
  - [Pub] symbol
  - [Pub] decimals
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Ext] withdraw #
  - modifiers: onlyOwner,nonReentrant
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] totalHODLrRewards
  - [Pub] totalBurned
  - [Pub] deliver #
  - [Pub] reflectionFromToken
  - [Pub] tokenFromReflection
  - [Pub] isExcludedFromReward
  - [Pub] excludeFromReward #
  - modifiers: onlyOwner
  - [Ext] includeInReward #
  - modifiers: onlyOwner
  - [Pub] excludeFromFee #
  - modifiers: onlyOwner
  - [Pub] includeInFee #
  - modifiers: onlyOwner
  - [Ext] setRewardFeePercent #
  - modifiers: onlyOwner
  - [Ext] setBurnFeePercent #
  - modifiers: onlyOwner
  - [Ext] setMaxTxPercent #
  - modifiers: onlyOwner
  - [Ext] <Fallback> ($)
  - [Prv] _HODLrFee #
  - [Prv] _getValues
  - [Prv] _getTValues
  - [Prv] _getRValues
  - [Prv] _getRate
  - [Prv] _getCurrentSupply
  - [Prv] calculateRewardFee
  - [Prv] calculateBurnFee
  - [Prv] removeAllFee #
  - [Prv] restoreAllFee #
  - [Pub] isExcludedFromFee
  - [Prv] _approve #
  - [Prv] _transfer #

- [Prv] _tokenTransfer #
- [Prv] _transferBurn #
- [Prv] _transferFromExcluded #
- [Prv] _transferToExcluded #
- [Prv] _transferStandard #
- [Prv] _transferBothExcluded #

($) = payable function
= non-constant function