auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

# Smart Contract Audit

FOR

# Myro Grok

DATED : 17 Jan, 2024

# AUDIT SUMMARY

**Project name** – Myro Grok

**Date**: 17 Jan, 2024

**Scope of Audit-** Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

**Audit Status:** Passed

## Issues Found

| Status | Critical | High | Medium | Low | Suggestion |
|---|---|---|---|---|---|
| Open | 0 | 0 | 1 | 1 | 2 |
| Acknowledged | 0 | 0 | 0 | 0 | 0 |
| Resolved | 0 | 0 | 0 | 0 | 0 |

# USED TOOLS

## Tools:

**1- Manual Review:**
A line by line code review has been performed by audit ace team.

**2- BSC Test Network:** All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

**3- Slither :**
The code has undergone static analysis using Slither.

# Token Information

**Token Name** : Myro Grok

**Token Symbol**: MyroG

**Decimals:** 18

**Token Supply**: 420,000,000,000,000

**Network:** BscScan

**Token Type:** BEP-20

**Token Address:**
0xf17fed96e995Beac0726972F41E80903Fd5b5f0a

**Checksum:**
A67acbefe2a12642d388659dffd20112

**Owner:**
0xAd79F47D6E03DF9F6ADbD6635c98392Cc62835b6
(at time of writing the audit)

**Deployer:**
0xAd79F47D6E03DF9F6ADbD6635c98392Cc62835b6

# TOKEN OVERVIEW

**Fees:**

**Buy Fee:** 0-0%

**Sell Fee:** 0-0%

**Transfer Fee:** 0-0%

**Fees Privilege:** Owner

**Ownership**: Owned

**Minting:** No mint function

**Max Tx Amount/ Max Wallet Amount: No**

**Blacklist: No**

# AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

# VULNERABILITY CHECKLIST

- Return values of low-level calls
- Private modifier
- Multiple Sends
- Using Suicide
- Gas Limitand Loops
- Address hardcoded
- Exception Disorder
- Using inline assembly
- Divide before multiply
- Missing Zero Address Validation
- Compiler version not fixed
- **Gasless Send**
- Using block.timestamp
- Re-entrancy
- Tautology or contradiction
- Timestamp Dependence
- Revert/require functions
- Use of tx.origin
- Integer overflow/underflow
- Dangerous strict equalities
- Using SHA3
- Using throw

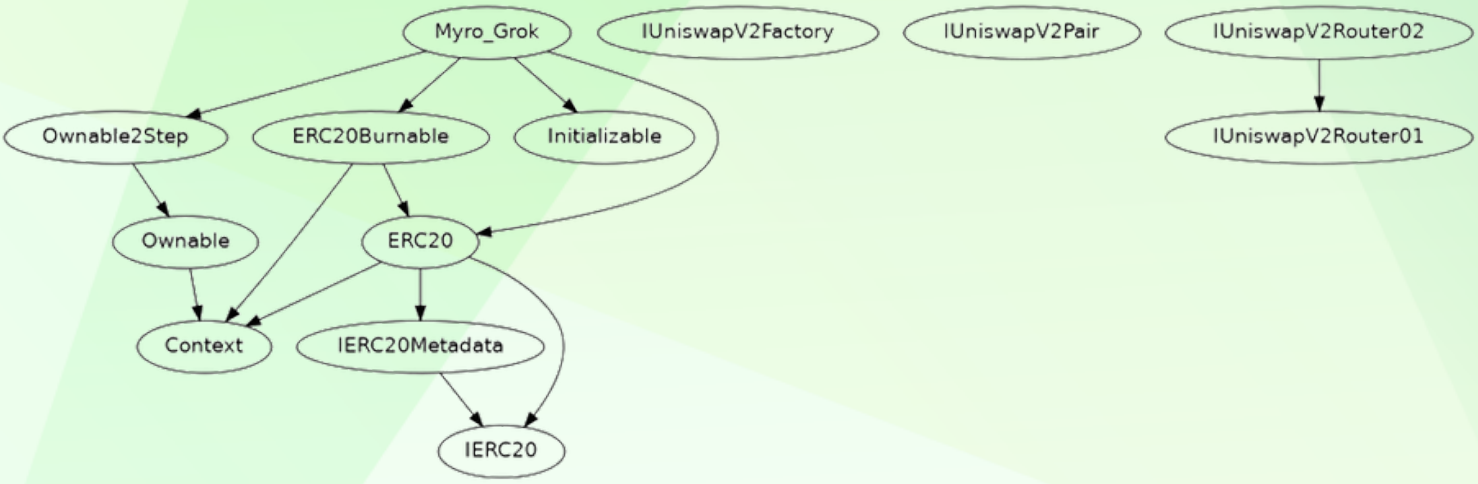
# CLASSIFICATION OF RISK

| Severity | Description |
| --- | --- |
| Critical | These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away. |
| High-Risk | A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way. |
| Medium-Risk | A vulnerability that could affect the desired outcome of executing the contract in a specific scenario. |
| Low-Risk | A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective. |
| Gas Optimization /Suggestion | A vulnerability that has an informational character but is not affecting any of the code. |

# Findings

| Severity | Found |
| --- | --- |
| Critical | 0 |
| High-Risk | 0 |
| Medium-Risk | 1 |
| Low-Risk | 1 |
| Gas Optimization / Suggestions | 2 |

# INHERITANCE TREE



# POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can update the swap threshold.
- The owner can update the marketing, dev, and liquidity fees not more than 25%.
- The owner can update the marketing/dev wallet address.

# STATIC ANALYSIS

**A static analysis of the code was performed using Slither.**

**No issues were found.**

```
INFO:Detectors:
Myro_Grok._transfer(address,address,uint256) (MyroGrok.sol#1457-1535) uses a Boolean constant improperly:
        -false || _marketingPending > 0 || _devPending > 0 (MyroGrok.sol#1498)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant
INFO:Detectors:
Myro_Grok._transfer(address,address,uint256) (MyroGrok.sol#1457-1535) performs a multiplication on the result of a division:
        - fees = amount * totalFees[txType] / 10000 (MyroGrok.sol#1476)
        - _marketingPending += fees * marketingFees[txType] / totalFees[txType] (MyroGrok.sol#1479)
Myro_Grok._transfer(address,address,uint256) (MyroGrok.sol#1457-1535) performs a multiplication on the result of a division:
        - fees = amount * totalFees[txType] / 10000 (MyroGrok.sol#1476)
        - _devPending += fees * devFees[txType] / totalFees[txType] (MyroGrok.sol#1481)
Myro_Grok._transfer(address,address,uint256) (MyroGrok.sol#1457-1535) performs a multiplication on the result of a division:
        - fees = amount * totalFees[txType] / 10000 (MyroGrok.sol#1476)
        - _liquidityPending += fees * liquidityFees[txType] / totalFees[txType] (MyroGrok.sol#1483)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Myro_Grok._addLiquidity(uint256,uint256) (MyroGrok.sol#1426-1430) ignores return value by routerV2.addLiquidityETH{value: coinAmount}(address(this),tokenAmo
unt,0,0,address(0),block.timestamp) (MyroGrok.sol#1429)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Ownable2Step.transferOwnership(address).newOwner (MyroGrok.sol#795) lacks a zero-check on :
                - _pendingOwner = newOwner (MyroGrok.sol#796)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Myro_Grok._swapAndLiquify(uint256) (MyroGrok.sol#1406-1424):
        External calls:
        - _swapTokensForCoin(halfAmount) (MyroGrok.sol#1411)
                - routerV2.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (MyroGrok.sol#1348)
        - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,coinBalance) (MyroGrok.sol#1416)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (MyroGrok.sol#1429)
        External calls sending eth:
        - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,coinBalance) (MyroGrok.sol#1416)
                - routerV2.addLiquidityETH{value: coinAmount}(address(this),tokenAmount,0,0,address(0),block.timestamp) (MyroGrok.sol#1429)
        State variables written after the call(s):
        - (amountToken,amountCoin,liquidity) = _addLiquidity(otherHalf,coinBalance) (MyroGrok.sol#1416)
                - _allowances[owner][spender] = value (MyroGrok.sol#594)
Reentrancy in Myro_Grok._transfer(address,address,uint256) (MyroGrok.sol#1457-1535):
        External calls:
```

```
INFO:Detectors:
Initializable._getInitializableStorage() (MyroGrok.sol#1043-1047) uses assembly
        - INLINE ASM (MyroGrok.sol#1044-1046)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Myro_Grok._transfer(address,address,uint256) (MyroGrok.sol#1457-1535) has a high cyclomatic complexity (16).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Context._contextSuffixLength() (MyroGrok.sol#144-146) is never used and should be removed
Context._msgData() (MyroGrok.sol#140-142) is never used and should be removed
ERC20._afterTokenTransfer(address,address,uint256) (MyroGrok.sol#420) is never used and should be removed
ERC20._beforeTokenTransfer(address,address,uint256) (MyroGrok.sol#415) is never used and should be removed
ERC20._mint(address,uint256) (MyroGrok.sol#529-534) is never used and should be removed
Initializable._checkInitializing() (MyroGrok.sol#998-1002) is never used and should be removed
Initializable._disableInitializers() (MyroGrok.sol#1012-1023) is never used and should be removed
Initializable._getInitializedVersion() (MyroGrok.sol#1028-1030) is never used and should be removed
Initializable._isInitializing() (MyroGrok.sol#1035-1037) is never used and should be removed
Myro_Grok._afterTokenTransfer(address,address,uint256) (MyroGrok.sol#1567-1571) is never used and should be removed
Myro_Grok._beforeTokenTransfer(address,address,uint256) (MyroGrok.sol#1561-1565) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.19 (MyroGrok.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (MyroGrok.sol#1085) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (MyroGrok.sol#1086) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (MyroGrok.sol#1103) is not in mixedCase
Function IUniswapV2Router01.WETH() (MyroGrok.sol#1125) is not in mixedCase
Contract Myro_Grok (MyroGrok.sol#1262-1572) is not in CapWords
Event Myro_Grok.marketingAddressUpdated(address) (MyroGrok.sol#1289) is not in CapWords
Event Myro_Grok.marketingFeesUpdated(uint16,uint16,uint16) (MyroGrok.sol#1290) is not in CapWords
Event Myro_Grok.marketingFeeSent(address,uint256) (MyroGrok.sol#1291) is not in CapWords
Event Myro_Grok.devAddressUpdated(address) (MyroGrok.sol#1293) is not in CapWords
Event Myro_Grok.devFeesUpdated(uint16,uint16,uint16) (MyroGrok.sol#1294) is not in CapWords
Event Myro_Grok.devFeeSent(address,uint256) (MyroGrok.sol#1295) is not in CapWords
Event Myro_Grok.liquidityFeesUpdated(uint16,uint16,uint16) (MyroGrok.sol#1297) is not in CapWords
Event Myro_Grok.liquidityAdded(uint256,uint256,uint256) (MyroGrok.sol#1298) is not in CapWords
```

# STATIC ANALYSIS

**A static analysis of the code was performed using Slither.**

**No issues were found.**

```
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (MyroGrok.sol#1130) is too similar
to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (MyroGrok.sol#1131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Myro_Grok.constructor() (MyroGrok.sol#1306-1326) uses literals with too many digits:
        - _mint(supplyRecipient,4200000000000000 * (10 ** decimals()) / 10) (MyroGrok.sol#1324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:MyroGrok.sol analyzed (16 contracts with 93 detectors), 66 result(s) found
```

# MANUAL TESTING

## Centralization – Missing Require Check

**Severity:** Medium

**Function:** marketingAddressSetup/devAddressSetup

**Status: Open**

**Overview:**

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function marketingAddressSetup(address _newAddress) public onlyOwner {
        require(_newAddress != address(0), "TaxesDefaultRouterWallet: Wallet tax
recipient cannot be a 0x0 address");

        marketingAddress = _newAddress;
        excludeFromFees(_newAddress, true);

        emit marketingAddressUpdated(_newAddress);
    }
 function devAddressSetup(address _newAddress) public onlyOwner {
        require(_newAddress != address(0), "TaxesDefaultRouterWallet: Wallet tax
recipient cannot be a 0x0 address");

        devAddress = _newAddress;
        excludeFromFees(_newAddress, true);

        emit devAddressUpdated(_newAddress);
    }
```

**Suggestion:**

It is recommended that the address should not be able to be set as a contract address.

# MANUAL TESTING

## Centralization – Missing Events

## Severity: Low

## Function: Missing Events

## Status: Open

**Overview:**

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setAMMPair(address pair, bool isPair) external onlyOwner {
        require(pair != pairV2, "DefaultRouter: Cannot remove initial pair from
list");

        _setAMMPair(pair, isPair);
    }
```

# MANUAL TESTING

## Optimization

**Severity: Informational**

**Subject: Floating Pragma**

**Status: Open**

**Overview:**

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.19;
```

**Suggestion:**

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

# MANUAL TESTING

## Optimization

**Severity: Optimization**

**Subject: Remove unused code**

**Status: Open**

**Overview:**

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them.

```
function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
```

# DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

# ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.


**https://auditace.tech/**

**https://t.me/Audit_Ace**

**https://twitter.com/auditace_**

**https://github.com/Audit-Ace**