auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Waifu Grok Mirai

DATED : 24 Nov 23'

AUDIT SUMMARY

Project name – Waifu Grok Mirai

Date: 24 Nov, 2023

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	5	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.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0x97eb274d19b2e12f5a6b0780cbde8ff6babe53a1#code

Token Information

Token Name : Waifu Grok Mirai

Token Symbol: wGROK

Decimals: 9

Token Supply: 10000000000000000

Network: ETH

Token Type: ERC20

Token Address:
0x0255Fd7b33645C38559C124f6970343F0e23f9d1

Checksum:
d8994660845409af00a6d89ce082c533

Owner:
0xaffE0c6E9Ef2552433c645752c6fAA21B184C48c
(at time of writing the audit)

Deployer:
0xaffE0c6E9Ef2552433c645752c6fAA21B184C48c

TOKEN OVERVIEW

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0-10%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges:

Initial distribution of the tokens

Modifying fees

Enabling trades

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

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

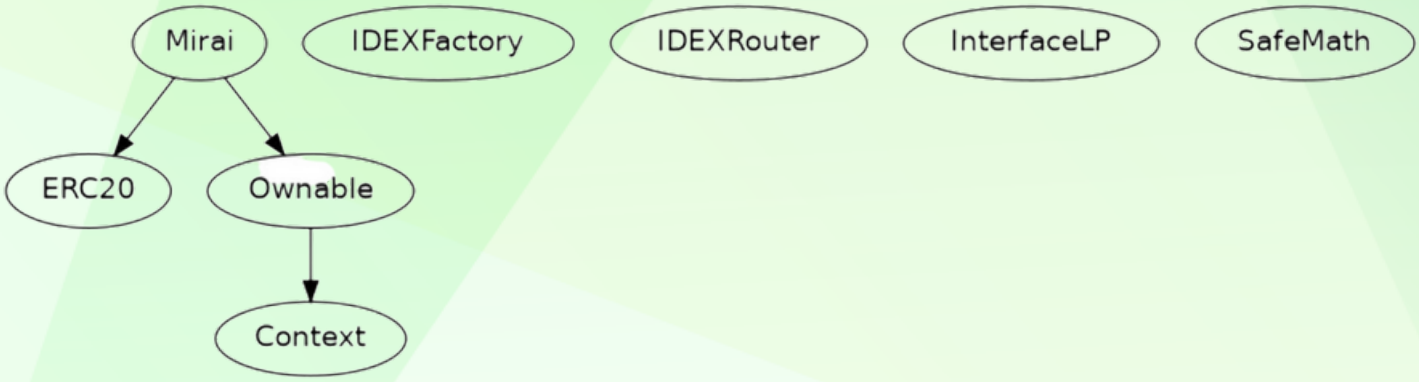
CLASSIFICATION OF RISK

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	1
◆ Medium-Risk	1
◆ Low-Risk	5
◆ Gas Optimization / Suggestions	2

INHERITANCE TREE



POINTS TO NOTE

- The owner can renounce the ownership.
- The owner can transfer the ownership.
- The Owner can set the max wallet to not less than 1%.
- The Owner can authorize.
- The Owner can white list wallets.
- The Owner can transfer.
- The owner can clear stuck tokens.
- The owner can set a fee multiplier of not more than 10%.
- The owner can set a tax of not more than 20%
- The owner can set a fee receiver.
- The owner can set swap back settings.

STATIC ANALYSIS

```
INFO:Detectors:
Mirai.takeFee(address,uint256,address) (Mirai.sol#371-395) performs a multiplication on the result of a division:
        - feeAmount = amount.mul(totalFee).mul(percent).div(feeDenominator * 100) (Mirai.sol#380)
        - burnTokens = feeAmount.mul(burnFee).div(totalFee) (Mirai.sol#381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Mirai.swapBack() (Mirai.sol#429-474) ignores return value by router.addLiquidityETH{value: amountETHLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (Mirai.sol#464-471)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Mirai.swapBack().tmpSuccess (Mirai.sol#457) is written in both
        (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
        (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
Mirai.swapBack().tmpSuccess (Mirai.sol#457) is written in both
        (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
        (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
Mirai.swapBack().tmpSuccess (Mirai.sol#457) is written in both
        (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
        tmpSuccess = false (Mirai.sol#461)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
INFO:Detectors:
Mirai.setFeeMultipliers(uint256,uint256,uint256) (Mirai.sol#418-427) should emit an event for:
        - sellpercent = _sell (Mirai.sol#419)
        - buypercent = _buy (Mirai.sol#420)
        - transferpercent = _trans (Mirai.sol#421)
Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256) (Mirai.sol#485-495) should emit an event for:
        - liquidityFee = _liquidityFee (Mirai.sol#486)
        - teamFee = _teamFee (Mirai.sol#487)
        - marketingFee = _marketingFee (Mirai.sol#488)
        - devFee = _devFee (Mirai.sol#489)
        - burnFee = _burnFee (Mirai.sol#490)
        - totalFee = _liquidityFee.add(_teamFee).add(_marketingFee).add(_devFee).add(_burnFee) (Mirai.sol#491)
        - feeDenominator = _feeDenominator (Mirai.sol#492)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Mirai.setFeeReceivers(address,address,address,address,address)._autoLiquidityReceiver (Mirai.sol#498) lacks a zero-check on :
                - autoLiquidityReceiver = _autoLiquidityReceiver (Mirai.sol#499)
Mirai.setFeeReceivers(address,address,address,address,address)._marketingFeeReceiver (Mirai.sol#498) lacks a zero-check on :
                - marketingFeeReceiver = _marketingFeeReceiver (Mirai.sol#500)
Mirai.setFeeReceivers(address,address,address,address,address)._devFeeReceiver (Mirai.sol#498) lacks a zero-check on :
                - devFeeReceiver = _devFeeReceiver (Mirai.sol#501)
Mirai.setFeeReceivers(address,address,address,address,address)._burnFeeReceiver (Mirai.sol#498) lacks a zero-check on :
                - burnFeeReceiver = _burnFeeReceiver (Mirai.sol#502)
Mirai.setFeeReceivers(address,address,address,address,address)._teamFeeReceiver (Mirai.sol#498) lacks a zero-check on :
                - teamFeeReceiver = _teamFeeReceiver (Mirai.sol#503)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Mirai._transferFrom(address,address,uint256) (Mirai.sol#331-354):
        External calls:
        - swapBack() (Mirai.sol#346)
```

```
INFO:Detectors:
Reentrancy in Mirai._transferFrom(address,address,uint256) (Mirai.sol#331-354):
        External calls:
        - swapBack() (Mirai.sol#346)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp) (Mirai.sol#440-446)
                - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
                - (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
                - (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
                - router.addLiquidityETH{value: amountETHLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (Mirai.sol#464-471)
        External calls sending eth:
        - swapBack() (Mirai.sol#346)
                - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
                - (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
                - (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
                - router.addLiquidityETH{value: amountETHLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (Mirai.sol#464-471)
        State variables written after the call(s):
        - amountReceived = takeFee(sender,amount,recipient) (Mirai.sol#349)
                - _totalSupply = _totalSupply.sub(burnTokens) (Mirai.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Mirai._transferFrom(address,address,uint256) (Mirai.sol#331-354):
        External calls:
        - swapBack() (Mirai.sol#346)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp) (Mirai.sol#440-446)
                - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
                - (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
                - (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
                - router.addLiquidityETH{value: amountETHLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (Mirai.sol#464-471)
        External calls sending eth:
        - swapBack() (Mirai.sol#346)
                - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
                - (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
                - (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
                - router.addLiquidityETH{value: amountETHLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (Mirai.sol#464-471)
        Event emitted after the call(s):
        - Transfer(sender,address(this),contractTokens) (Mirai.sol#385)
                - amountReceived = takeFee(sender,amount,recipient) (Mirai.sol#349)
        - Transfer(sender,ZERO,burnTokens) (Mirai.sol#390)
                - amountReceived = takeFee(sender,amount,recipient) (Mirai.sol#349)
        - Transfer(sender,recipient,amountReceived) (Mirai.sol#352)
Reentrancy in Mirai.swapBack() (Mirai.sol#429-474):
        External calls:
        - router.swapExactTokensForETHSupportingFeeOnTransferTokens(amountToSwap,0,path,address(this),block.timestamp) (Mirai.sol#440-446)
        - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
        - (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
        - (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
        - router.addLiquidityETH{value: amountETHLiquidity}(address(this),amountToLiquify,0,0,autoLiquidityReceiver,block.timestamp) (Mirai.sol#464-471)
        External calls sending eth:
        - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
```

STATIC ANALYSIS

```
INFO:Detectors:
Context._msgData() (Mirai.sol#52-55) is never used and should be removed
Mirai.shouldTakeFee(address) (Mirai.sol#367-369) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Mirai._maxTxAmount (Mirai.sol#217) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.mul(100).div(100)
Mirai._maxWalletToken (Mirai.sol#218) is set pre-construction with a non-constant function or state variable:
        - _totalSupply.mul(2).div(100)
Mirai.totalFee (Mirai.sol#230) is set pre-construction with a non-constant function or state variable:
        - teamFee + marketingFee + liquidityFee + devFee + burnFee
Mirai.swapThreshold (Mirai.sol#252) is set pre-construction with a non-constant function or state variable:
        - _totalSupply * 50 / 10000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Pragma version0.8.20 (Mirai.sol#29) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Mirai.swapBack() (Mirai.sol#429-474):
        - (tmpSuccess) = address(marketingFeeReceiver).call{value: amountETHMarketing}() (Mirai.sol#457)
        - (tmpSuccess,None) = address(devFeeReceiver).call{value: amountETHdev}() (Mirai.sol#458)
        - (tmpSuccess,None) = address(teamFeeReceiver).call{value: amountETHteam}() (Mirai.sol#459)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Variable Ownable._owner (Mirai.sol#59) is not in mixedCase
Function IDEXRouter.WETH() (Mirai.sol#98) is not in mixedCase
Event Mirai.user_exemptfromfees(address,bool) (Mirai.sol#207) is not in CapWords
Event Mirai.user_TxExempt(address,bool) (Mirai.sol#208) is not in CapWords
Event Mirai.set_Receivers(address,address,address,address) (Mirai.sol#211) is not in CapWords
Event Mirai.set_MaxWallet(uint256) (Mirai.sol#212) is not in CapWords
Event Mirai.set_SwapBack(uint256,bool) (Mirai.sol#213) is not in CapWords
Parameter Mirai.setFeeMultipliers(uint256,uint256,uint256)._buy (Mirai.sol#418) is not in mixedCase
Parameter Mirai.setFeeMultipliers(uint256,uint256,uint256)._sell (Mirai.sol#418) is not in mixedCase
Parameter Mirai.setFeeMultipliers(uint256,uint256,uint256)._trans (Mirai.sol#418) is not in mixedCase
Function Mirai.set_fees() (Mirai.sol#477-483) is not in mixedCase
Parameter Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256)._liquidityFee (Mirai.sol#485) is not in mixedCase
Parameter Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256)._teamFee (Mirai.sol#485) is not in mixedCase
Parameter Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256)._marketingFee (Mirai.sol#485) is not in mixedCase
Parameter Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256)._devFee (Mirai.sol#485) is not in mixedCase
Parameter Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256)._burnFee (Mirai.sol#485) is not in mixedCase
Parameter Mirai.setTax(uint256,uint256,uint256,uint256,uint256,uint256)._feeDenominator (Mirai.sol#485) is not in mixedCase
Parameter Mirai.setFeeReceivers(address,address,address,address,address)._autoLiquidityReceiver (Mirai.sol#498) is not in mixedCase
Parameter Mirai.setFeeReceivers(address,address,address,address,address)._marketingFeeReceiver (Mirai.sol#498) is not in mixedCase
Parameter Mirai.setFeeReceivers(address,address,address,address,address)._devFeeReceiver (Mirai.sol#498) is not in mixedCase
Parameter Mirai.setFeeReceivers(address,address,address,address,address)._burnFeeReceiver (Mirai.sol#498) is not in mixedCase
Parameter Mirai.setFeeReceivers(address,address,address,address,address)._teamFeeReceiver (Mirai.sol#498) is not in mixedCase
Parameter Mirai.setSwapBackSettings(bool,uint256)._enabled (Mirai.sol#508) is not in mixedCase
```

```
INFO:Detectors:
Redundant expression "this (Mirai.sol#53)" inContext (Mirai.sol#46-56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Mirai.sol#103) is too similar to IDEXRouter.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (Mirai.sol#104)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Mirai.slitherConstructorVariables() (Mirai.sol#193-520) uses literals with too many digits:
        - _totalSupply = 100000000 * 10 ** _decimals (Mirai.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Mirai.TradingOpen (Mirai.sol#249) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Mirai.WETH (Mirai.sol#196) should be immutable
Mirai._maxTxAmount (Mirai.sol#217) should be immutable
Mirai.pair (Mirai.sol#248) should be immutable
Mirai.pairContract (Mirai.sol#247) should be immutable
Mirai.router (Mirai.sol#246) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:Mirai.sol analyzed (8 contracts with 93 detectors), 67 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x7c343b169d188c1a216e7a4659f8a92a10e7d11eb75e02f9b8729116bb84dd66

2- Approve Max (passed):

https://testnet.bscscan.com/tx/0x313c71ab11c07e68756f32967c3aa552bc33be3cc1b7b0c28745048c7f7f4356

3- Authorize (passed):

https://testnet.bscscan.com/tx/0x5810d21a44ceb05ca3251bd669161ccda4b6567707fbd57c03f77afa84272bad

4- Set Fee Receivers (passed):

https://testnet.bscscan.com/tx/0xe84a7e514143281eb6794d266f11f697999cd30aa1012f0fce7181b92c4da942

5- Set Max Wallet (passed):

https://testnet.bscscan.com/tx/0x7ae48d0d654a9022a6577e8a0601d2e1dcaf8e71044b95b9067511fc59881542

6- Set Swap Back Settings (passed):

https://testnet.bscscan.com/tx/0xa987d9b4e5e7903d5bccd86085c49ae6bbe21a7d8234317cd963826728480e6a

FUNCTIONAL TESTING

7- Set Whitelist Address (passed):

https://testnet.bscscan.com/tx/0x7e5d9159dc53a70bee180cb4f06f0516ace5c257aaeac7579322e3ee8a6aa964

8- Transfer (passed):

https://testnet.bscscan.com/tx/0xc91776998d3c9c09653f792ac2fe2c8837a6ec639d52d73ad0696259b7a826df

9- clear Stuck ETH (passed):

https://testnet.bscscan.com/tx/0x6c8a11b805bfcb038d8245cc5036693306199cebab75143d14fed9566ab2cc08

MANUAL TESTING

Centralization – Liquidity is added to EOA

Severity: Medium

function: swapBack

Status: Open

Overview:

Liquidity is adding to EOA. It may be drained by the autoliquidity receiver.

```
if(amountToLiquify > 0){
        router.addLiquidityETH{value: amountETHLiquidity}(
            address(this),
            amountToLiquify,
            0,
            0,
            autoLiquidityReceiver,
            block.timestamp
        );
        emit AutoLiquify(amountETHLiquidity, amountToLiquify);
    }
```

Suggestion:

It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Subject: Zero address

Status: Open

Overview:

functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
 function setWhitelistAddresss(address holder, bool exempt) external
onlyOwner {
      isFeeExempt[holder] = exempt;
      isTxLimitExempt[holder] = exempt;
  }

function setFeeReceivers(address _autoLiquidityReceiver, address
_marketingFeeReceiver, address _devFeeReceiver, address
_burnFeeReceiver, address _teamFeeReceiver) external onlyOwner {
      autoLiquidityReceiver = _autoLiquidityReceiver;
      marketingFeeReceiver = _marketingFeeReceiver;
      devFeeReceiver = _devFeeReceiver;
     burnFeeReceiver = _burnFeeReceiver;
      teamFeeReceiver = _teamFeeReceiver;

      emit set_Receivers(marketingFeeReceiver, teamFeeReceiver,
burnFeeReceiver, devFeeReceiver);
  }
```

Suggestion:

It is suggested that the address should not be zero or dead.

MANUAL TESTING

Optimization

Severity: Low

subject: 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 setWhitelistAddresss(address holder, bool exempt) external onlyOwner {
        isFeeExempt[holder] = exempt;
        isTxLimitExempt[holder] = exempt;
    }
```

MANUAL TESTING

Centralization – Missing Visibility

Severity: Low

subject: Visibility

Status: Open

Overview:

It's simply saying that no visibility was specified, so it's going with the default. This has been related to security issues in contracts.

```
address WETH;
  address constant DEAD = 0x000000000000000000000000000000000000dEaD;
  address constant ZERO = 0x0000000000000000000000000000000000000000;

string constant _name = "Waifu Grok Mirai";
  string constant _symbol = "wGROK";
  uint8 constant _decimals = 9;

uint256 _totalSupply = 100000000 * 10**_decimals;

mapping (address => uint256) _balances;
  mapping (address => mapping (address => uint256)) _allowances;
  mapping (address => bool) isexemptfromfees;
  mapping (address => bool) isexemptfrommaxTX;

uint256 sellpercent = 100;
  uint256 buypercent = 100;
  uint256 transferpercent = 0;

mapping (address => bool) isFeeExempt;
  mapping (address => bool) isTxLimitExempt;

bool inSwap;
```

Suggestion:

You can easily silence the warning by adding the public/private.

MANUAL TESTING

Centralization – unused Event
Severity: Low
subject: Event
Status: Open

Overview:
Events are unused.

event user_exemptfromfees(address Wallet, bool Exempt);
event user_TxExempt(address Wallet, bool Exempt);
event ClearStuck(uint256 amount);
event ClearToken(address TokenAddressCleared, uint256 Amount);

Suggestion:
It is suggested to remove unused events.

MANUAL TESTING

Centralization – Unnecessary calculation
Severity: Low
subject: Calculation
Status: Open

Overview:

uint256 public _maxTxAmount = _totalSupply.mul(100).div(100);

Suggestion:
To reduce high gas fees. It is suggested to remove the unnecessary calculation.

MANUAL TESTING

Optimization

Severity: Informational

subject: Remove Safe Math

Status: Open

Line: 149-184

Overview:

compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow, It is recommended to remove the unwanted code in order to avoid high gas fees.

MANUAL TESTING

Optimization

Severity: Informational

subject: Missing requires error message

Status: Open

function: setMaxwallet

Overview:

```
  function setMaxWallet(uint256 maxWallPercent) external onlyOwner {
        require(maxWallPercent >= 1);
      _maxWalletToken = (_totalSupply * maxWallPercent ) / 1000;
      emit set_MaxWallet(_maxWalletToken);

   }
```

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