AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

GROK AI 2.0

DATED : 08 November 23'

AUDIT SUMMARY

Project name – GROK AI 2.0

Date: 08 November 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	0	2	0
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/0xd5673e47c6627522ea69e3412a283098118624 58

Token Information

Token Address: -
0x7a76573b4e0207f58921C7356B800B6cea590131

Name: GROK AI 2.0

Symbol: GROK20

Decimals: 9

Network: Binance smart chain

Token Type: ERC20

Owner: - 0xbfA9e00288D16aFb401f265DD50793874B4a84f8

Deployer: -
0xbfA9e00288D16aFb401f265DD50793874B4a84f8

Token Supply: 420000000000

Checksum: 0ff4376de461c7768aeb7a75b5a3f1ee

Testnet version: -
https://testnet.bscscan.com/address/0xd5673e47c6627522ea69e3412a283098118624 58

TOKEN OVERVIEW

buy fee: 0-5%

Sell fee: 0-15%

transfer fee: 0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

Other Privileges:

- Initial distribution of the tokens
- Modifying fees
- Enabling trades
-bulk exempts fee
-claim stuck tokens.
— Update deadline

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.

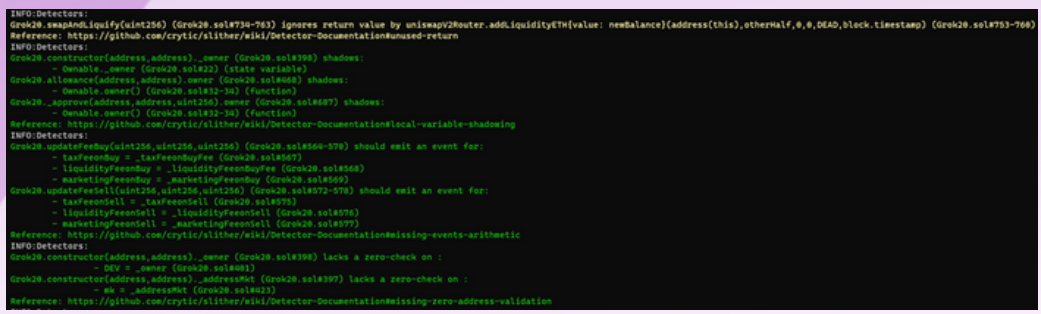
Findings

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

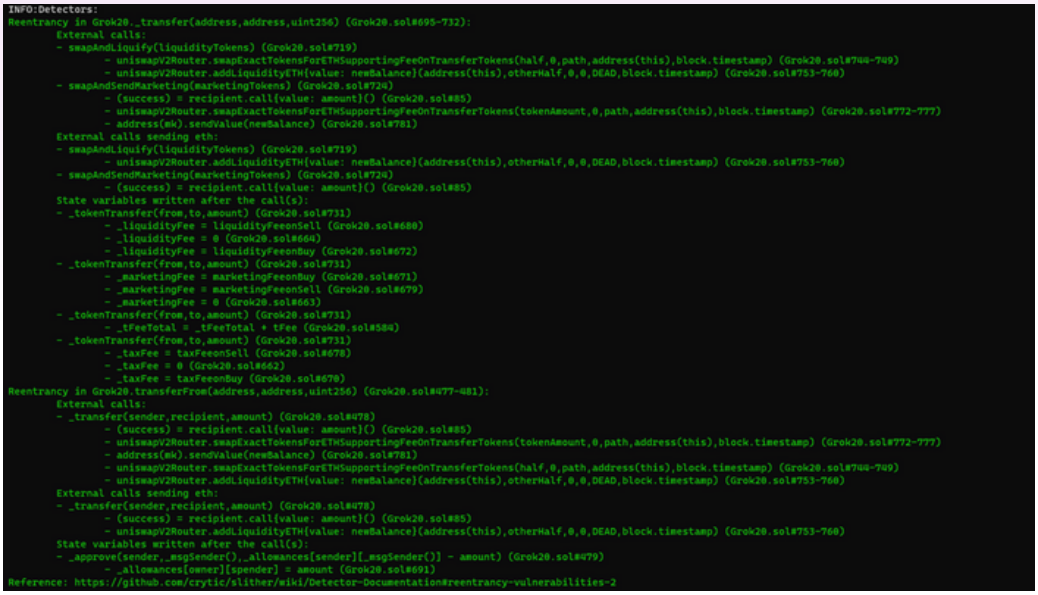
POINTS TO NOTE

- Owner can renounce the ownership.
- Owner can transfer the ownership.
- Owner can exclude wallets from rewards.
- Owner can set buy and sell fee not more than 10%.
- Owner can set the minimum swap amount not less than 0.001% of the total supply.
- Owner can set swapthreshold

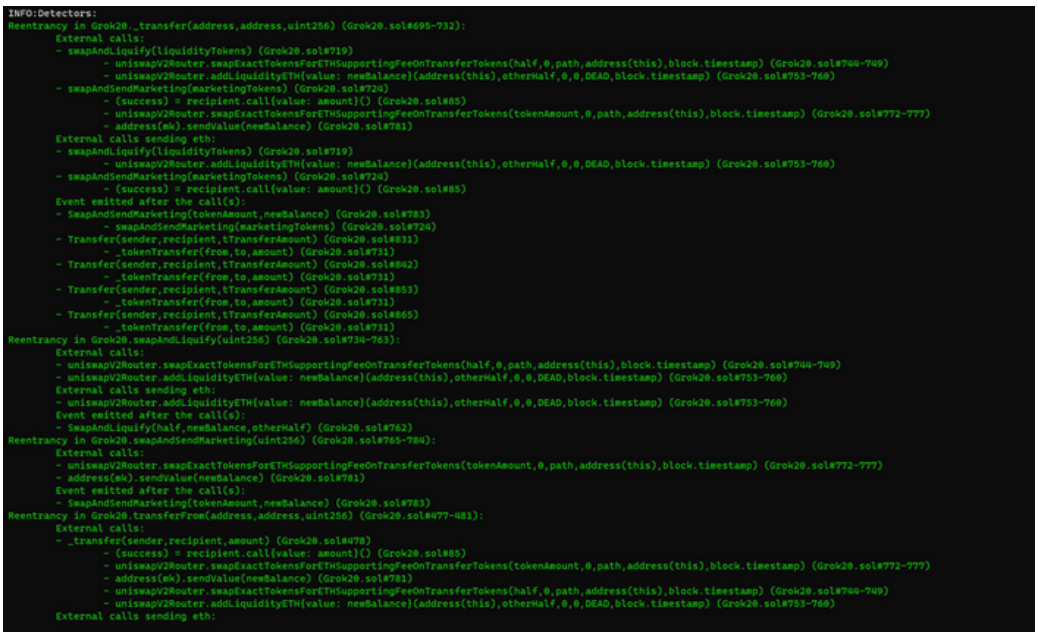
STATIC ANALYSIS



```
INFO:Detectors:
Grok20.swapAndLiquify(uint256) (Grok20.sol#734-763) ignores return value by uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Grok20.constructor(address,address)._owner (Grok20.sol#398) shadows:
        - Ownable._owner (Grok20.sol#22) (state variable)
Grok20.allowance(address,address).owner (Grok20.sol#468) shadows:
        - Ownable.owner() (Grok20.sol#32-34) (function)
Grok20._approve(address,address,uint256).owner (Grok20.sol#687) shadows:
        - Ownable.owner() (Grok20.sol#32-34) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Grok20.updateFeeBuy(uint256,uint256,uint256) (Grok20.sol#564-570) should emit an event for:
        - taxFeeonBuy = _taxFeeonBuyFee (Grok20.sol#567)
        - liquidityFeeonBuy = _liquidityFeeonBuyFee (Grok20.sol#568)
        - marketingFeeonBuy = _marketingFeeonBuy (Grok20.sol#569)
Grok20.updateFeeSell(uint256,uint256,uint256) (Grok20.sol#572-578) should emit an event for:
        - taxFeeonSell = _taxFeeonSell (Grok20.sol#575)
        - liquidityFeeonSell = _liquidityFeeonSell (Grok20.sol#576)
        - marketingFeeonSell = _marketingFeeonSell (Grok20.sol#577)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Grok20.constructor(address,address)._owner (Grok20.sol#398) lacks a zero-check on :
                - DEV = _owner (Grok20.sol#401)
Grok20.constructor(address,address)._addressMkt (Grok20.sol#397) lacks a zero-check on :
                - mk = _addressMkt (Grok20.sol#421)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



```
INFO:Detectors:
Reentrancy in Grok20._transfer(address,address,uint256) (Grok20.sol#695-732):
        External calls:
        - swapAndLiquify(liquidityTokens) (Grok20.sol#719)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (Grok20.sol#744-749)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        - swapAndSendMarketing(marketingTokens) (Grok20.sol#724)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Grok20.sol#772-777)
                - address(mk).sendValue(newBalance) (Grok20.sol#781)
        External calls sending eth:
        - swapAndLiquify(liquidityTokens) (Grok20.sol#719)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        - swapAndSendMarketing(marketingTokens) (Grok20.sol#724)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount) (Grok20.sol#731)
                - _liquidityFee = liquidityFeeonSell (Grok20.sol#680)
                - _liquidityFee = 0 (Grok20.sol#666)
                - _liquidityFee = liquidityFeeonBuy (Grok20.sol#672)
        - _tokenTransfer(from,to,amount) (Grok20.sol#731)
                - _marketingFee = marketingFeeonBuy (Grok20.sol#671)
                - _marketingFee = marketingFeeonSell (Grok20.sol#679)
                - _marketingFee = 0 (Grok20.sol#663)
        - _tokenTransfer(from,to,amount) (Grok20.sol#731)
                - _tFeeTotal = _tFeeTotal + tFee (Grok20.sol#584)
        - _tokenTransfer(from,to,amount) (Grok20.sol#731)
                - _taxFee = taxFeeonSell (Grok20.sol#678)
                - _taxFee = 0 (Grok20.sol#662)
                - _taxFee = taxFeeonBuy (Grok20.sol#670)
Reentrancy in Grok20.transferFrom(address,address,uint256) (Grok20.sol#477-481):
        External calls:
        - _transfer(sender,recipient,amount) (Grok20.sol#478)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Grok20.sol#772-777)
                - address(mk).sendValue(newBalance) (Grok20.sol#781)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (Grok20.sol#744-749)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (Grok20.sol#478)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()] - amount) (Grok20.sol#479)
                - _allowances[owner][spender] = amount (Grok20.sol#691)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```



```
INFO:Detectors:
Reentrancy in Grok20._transfer(address,address,uint256) (Grok20.sol#695-732):
        External calls:
        - swapAndLiquify(liquidityTokens) (Grok20.sol#719)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (Grok20.sol#744-749)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        - swapAndSendMarketing(marketingTokens) (Grok20.sol#724)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Grok20.sol#772-777)
                - address(mk).sendValue(newBalance) (Grok20.sol#781)
        External calls sending eth:
        - swapAndLiquify(liquidityTokens) (Grok20.sol#719)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        - swapAndSendMarketing(marketingTokens) (Grok20.sol#724)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
        Event emitted after the call(s):
        - SwapAndSendMarketing(tokenAmount,newBalance) (Grok20.sol#783)
                - swapAndSendMarketing(marketingTokens) (Grok20.sol#724)
        - Transfer(sender,recipient,tTransferAmount) (Grok20.sol#811)
                - _tokenTransfer(from,to,amount) (Grok20.sol#731)
        - Transfer(sender,recipient,tTransferAmount) (Grok20.sol#842)
                - _tokenTransfer(from,to,amount) (Grok20.sol#731)
        - Transfer(sender,recipient,tTransferAmount) (Grok20.sol#853)
                - _tokenTransfer(from,to,amount) (Grok20.sol#731)
        - Transfer(sender,recipient,tTransferAmount) (Grok20.sol#865)
                - _tokenTransfer(from,to,amount) (Grok20.sol#731)
Reentrancy in Grok20.swapAndLiquify(uint256) (Grok20.sol#734-763):
        External calls:
        - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (Grok20.sol#744-749)
        - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        External calls sending eth:
        - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        Event emitted after the call(s):
        - SwapAndLiquify(half,newBalance,otherHalf) (Grok20.sol#762)
Reentrancy in Grok20.swapAndSendMarketing(uint256) (Grok20.sol#765-784):
        External calls:
        - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Grok20.sol#772-777)
        - address(mk).sendValue(newBalance) (Grok20.sol#781)
        Event emitted after the call(s):
        - SwapAndSendMarketing(tokenAmount,newBalance) (Grok20.sol#783)
Reentrancy in Grok20.transferFrom(address,address,uint256) (Grok20.sol#477-481):
        External calls:
        - _transfer(sender,recipient,amount) (Grok20.sol#478)
                - (success) = recipient.call{value: amount}() (Grok20.sol#85)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (Grok20.sol#772-777)
                - address(mk).sendValue(newBalance) (Grok20.sol#781)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(half,0,path,address(this),block.timestamp) (Grok20.sol#744-749)
                - uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,DEAD,block.timestamp) (Grok20.sol#753-760)
        External calls sending eth:
```

STATIC ANALYSIS

```
INFO:Detectors:
Grok20._tTotal (Grok20.sol#347) is set pre-construction with a non-constant function or state variable:
        - 100_000_000_000 * (10 ** _decimals)
Grok20._rTotal (Grok20.sol#348) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Grok20.totalBuyFees (Grok20.sol#364) is set pre-construction with a non-constant function or state variable:
        - taxFeeonBuy + liquidityFeeonBuy + marketingFeeonBuy
Grok20.totalSellFees (Grok20.sol#365) is set pre-construction with a non-constant function or state variable:
        - taxFeeonSell + liquidityFeeonSell + marketingFeeonSell
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state
INFO:Detectors:
Pragma version0.8.17 (Grok20.sol#8) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Grok20.sol#81-87):
        - (success) = recipient.call{value: amount}() (Grok20.sol#85)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Grok20.sol#106-127):
        - (success,returndata) = target.call{value: weiValue}(data) (Grok20.sol#110)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (Grok20.sol#161) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (Grok20.sol#162) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (Grok20.sol#178) is not in mixedCase
Function IUniswapV2Router01.WETH() (Grok20.sol#197) is not in mixedCase
Event Grok20.mkChanged(address) (Grok20.sol#380) is not in CapWords
Parameter Grok20.updateFeeBuy(uint256,uint256,uint256)._taxFeeonBuyFee (Grok20.sol#564) is not in mixedCase
Parameter Grok20.updateFeeBuy(uint256,uint256,uint256)._liquidityFeeonBuyFee (Grok20.sol#564) is not in mixedCase
Parameter Grok20.updateFeeBuy(uint256,uint256,uint256)._marketingFeeonBuy (Grok20.sol#564) is not in mixedCase
Parameter Grok20.updateFeeSell(uint256,uint256,uint256)._taxFeeonSell (Grok20.sol#572) is not in mixedCase
Parameter Grok20.updateFeeSell(uint256,uint256,uint256)._liquidityFeeonSell (Grok20.sol#572) is not in mixedCase
Parameter Grok20.updateFeeSell(uint256,uint256,uint256)._marketingFeeonSell (Grok20.sol#572) is not in mixedCase
Parameter Grok20.calculateTaxFee(uint256)._amount (Grok20.sol#647) is not in mixedCase
Parameter Grok20.calculateLiquidityFee(uint256)._amount (Grok20.sol#651) is not in mixedCase
Parameter Grok20.calculateMarketingFee(uint256)._amount (Grok20.sol#655) is not in mixedCase
Parameter Grok20.setSwapEnabled(bool)._enabled (Grok20.sol#792) is not in mixedCase
Variable Grok20.DEAD (Grok20.sol#369) is not in mixedCase
Variable Grok20.DEV (Grok20.sol#370) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Grok20.balances (Grok20.sol#389) is never used in Grok20 (Grok20.sol#330-879)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
Loop condition i < _excluded.length (Grok20.sol#618) should use cached array length instead of referencing `length` member of the storage array.
 Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cache-array-length
INFO:Detectors:
Grok20.DEAD (Grok20.sol#369) should be constant
Grok20._decimals (Grok20.sol#344) should be constant
Grok20._name (Grok20.sol#342) should be constant
Grok20._symbol (Grok20.sol#343) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Grok20.DEV (Grok20.sol#370) should be immutable
Grok20._tTotal (Grok20.sol#347) should be immutable
Grok20.mk (Grok20.sol#367) should be immutable
Grok20.totalBuyFees (Grok20.sol#364) should be immutable
Grok20.totalSellFees (Grok20.sol#365) should be immutable
Grok20.uniswapV2Pair (Grok20.sol#373) should be immutable
Grok20.uniswapV2Router (Grok20.sol#372) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:Grok20.sol analyzed (9 contracts with 93 detectors), 109 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

Exclude from reward(passed) -
https://testnet.bscscan.com/tx/0xc6674448d0cf10dc379782f4ad647581d4068c41190ca320d4bfd35a2a312d5e

Include in reward(passed) -
https://testnet.bscscan.com/tx/0xbb35d8a1244a3331e4d3b34a3f22623ccad0375476ed2fd559224089eff592f9

Update Fee Buy(Passed) -
https://testnet.bscscan.com/tx/0x10fc53b6f70330f330602980a28c5b2aa2b8c803f176eb435afaf2f265158898

Update Fee Sell(Passed) -
https://testnet.bscscan.com/tx/0xe28aeb6cd350494a862347ddff35ea83a64d721f686613de7fda6cb1f3778dfd

Set Swap Token At Amount(Passed) -
https://testnet.bscscan.com/tx/0x4f31488f7ae3b03c520feec636a15ce33d418af658c69f155cc573757ce94d38

Set Swap Enabled(Passed) -
https://testnet.bscscan.com/tx/0xa331d6d73c8b254fbc6d3557267b4b951bec786401e9f4aa92e7c466ad8da6fe

Exclude From Fees(Passed) -
https://testnet.bscscan.com/tx/0x23743d823aa84233eb2f918e31b45d4ba6dcf28155a6e2e15210086e2d713774

MANUAL TESTING

Severity: Low
function: excludeFromReward
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 excludeFromReward(address account) public onlyOwner() {
  require(!_isExcluded[account], "Account is already excluded");
  if(_rOwned[account] > 0) {
    _tOwned[account] = tokenFromReflection(_rOwned[account]);
  }
  _isExcluded[account] = true;
  _excluded.push(account);
}
```

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

MANUAL TESTING

Severity: Low
function: mapping
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.

mapping (address => uint256) balances;

Suggestion:
You can easily silence the warning by adding the mapping public:

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