



Smart Contract Audit

FOR
GOKUSWAP

DATED : 09 January 2024



AUDIT SUMMARY

Project name – GOKUSWAP

Date: 09 January, 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	0	3	1
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/0x25CEb7468042A1a7B9Dc0098b9111Cd742CF21D9#code>



Token Information

Token Address:

0x87429B114315E8DBfA8b9611BEf07EcAD9a13742

Name: GOKUSWAP

Symbol: GOKU

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner:

0x13e0ACd34331E682F99658b4Bf418d1296e586D6

Deployer:

0x8fE091c76D372204715D1819747cb4b41baDD49C

Token Supply: 1000000000

Checksum: Ae1c3a4fbb6e83e8393a57617b5a5b17

Testnet:

<https://testnet.bscscan.com/address/0x25CEb7468042A1a7B9Dc0098b9111Cd742CF21D9#code>



TOKEN OVERVIEW

Buy Fee: 3-5%

Sell Fee: 3-5%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes

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

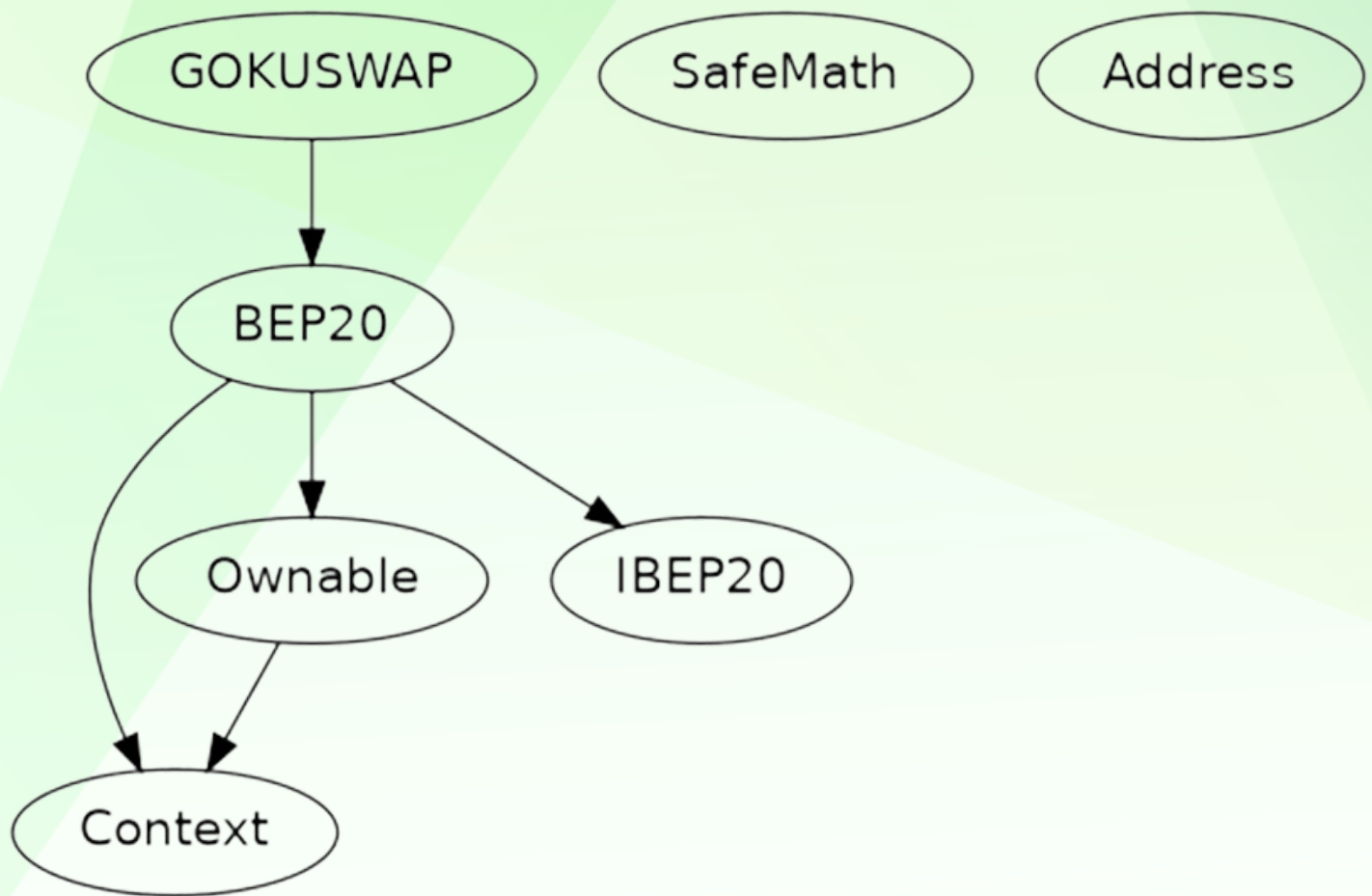
Findings

Severity

Found

◆ Critical	0
◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	3
◆ Gas Optimization / Suggestions	1

INHERITANCE TREE





POINTS TO NOTE

- The owner can renounce ownership.
 - The owner can transfer ownership.
 - The owner can change the buy and sell tax not more than 25%.
 - The owner can change the marketing Pool address.
 - The owner can set liquidity pool status.
-



STATIC ANALYSIS

```
INFO:Detectors:
BEP20.constructor(string,string).name (GOKUSWAP.sol#212) shadows:
  - BEP20.name() (GOKUSWAP.sol#222-224) (function)
  - IBEP20.name() (GOKUSWAP.sol#62) (function)
BEP20.constructor(string,string).symbol (GOKUSWAP.sol#212) shadows:
  - BEP20.symbol() (GOKUSWAP.sol#226-228) (function)
  - IBEP20.symbol() (GOKUSWAP.sol#60) (function)
BEP20.allowance(address,address).owner (GOKUSWAP.sol#247) shadows:
  - Ownable.owner() (GOKUSWAP.sol#31-33) (function)
BEP20._approve(address,address,uint256).owner (GOKUSWAP.sol#304) shadows:
  - Ownable.owner() (GOKUSWAP.sol#31-33) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
GOKUSWAP.setMarketingPool(address)._marketingPool (GOKUSWAP.sol#355) lacks a zero-check on :
  - marketingPool = _marketingPool (GOKUSWAP.sol#356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Address.isContract(address) (GOKUSWAP.sol#132-139) uses assembly
  - INLINE ASM (GOKUSWAP.sol#135-137)
Address._functionCallWithValue(address,bytes,uint256,string) (GOKUSWAP.sol#177-197) uses assembly
  - INLINE ASM (GOKUSWAP.sol#189-192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
GOKUSWAP._transfer(address,address,uint256) (GOKUSWAP.sol#377-388) compares to a boolean constant:
  - liquidityPool[sender] == true (GOKUSWAP.sol#379)
GOKUSWAP._transfer(address,address,uint256) (GOKUSWAP.sol#377-388) compares to a boolean constant:
  - liquidityPool[recipient] == true (GOKUSWAP.sol#381)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Different versions of Solidity are used:
  - Version used: ['0.6.12', '>=0.4.0', '>=0.6.0<0.8.0', '>=0.6.4']
  - 0.6.12 (GOKUSWAP.sol#317)
  - >=0.4.0 (GOKUSWAP.sol#200)
  - >=0.6.0<0.8.0 (GOKUSWAP.sol#6)
  - >=0.6.0<0.8.0 (GOKUSWAP.sol#19)
  - >=0.6.0<0.8.0 (GOKUSWAP.sol#81)
  - >=0.6.4 (GOKUSWAP.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
INFO:Detectors:
Pragma version>=0.6.0<0.8.0 (GOKUSWAP.sol#6) is too complex
Pragma version>=0.6.0<0.8.0 (GOKUSWAP.sol#19) is too complex
Pragma version>=0.6.4 (GOKUSWAP.sol#52) allows old versions
Pragma version>=0.6.0<0.8.0 (GOKUSWAP.sol#81) is too complex
Pragma version>=0.4.0 (GOKUSWAP.sol#200) allows old versions
Pragma version0.6.12 (GOKUSWAP.sol#317) allows old versions
solc-0.6.12 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) (GOKUSWAP.sol#141-145):
  - (success) = recipient.call{value: amount}() (GOKUSWAP.sol#143)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (GOKUSWAP.sol#177-197):
  - (success,returndata) = target.call{value: weiValue}(data) (GOKUSWAP.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter GOKUSWAP.setMarketingPool(address)._marketingPool (GOKUSWAP.sol#355) is not in mixedCase
Parameter GOKUSWAP.updateMaxTransferAmountRate(uint16)._maxTransferAmountRate (GOKUSWAP.sol#360) is not in mixedCase
Parameter GOKUSWAP.setLiquidityPoolStatus(address,bool)._lpAddress (GOKUSWAP.sol#372) is not in mixedCase
Parameter GOKUSWAP.setLiquidityPoolStatus(address,bool)._status (GOKUSWAP.sol#372) is not in mixedCase
Parameter GOKUSWAP.setTaxes(uint8,uint8)._sellTax (GOKUSWAP.sol#390) is not in mixedCase
Parameter GOKUSWAP.setTaxes(uint8,uint8)._buyTax (GOKUSWAP.sol#390) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (GOKUSWAP.sol#14)" inContext (GOKUSWAP.sol#8-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
GOKUSWAP.constructor() (GOKUSWAP.sol#338-345) uses literals with too many digits:
  - _mint(_msgSender(),1000000000 * 1000000000000000000) (GOKUSWAP.sol#344)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:GOKUSWAP.sol analyzed (7 contracts with 93 detectors), 40 result(s) found
```



FUNCTIONAL TESTING

1- **Approve** (passed):

<https://testnet.bscscan.com/tx/0x123fd8b38d2bccbc11de125e31a646de3978adbc83c283ea24e2f2377ef3c3a0>

2- **Increase Allowance** (passed):

<https://testnet.bscscan.com/tx/0x25937ff1b0b0873369dae3ff8283d8aa9562f31818061d6ecad4c8682ca53132>

3- **Decrease Allowance** (passed):

<https://testnet.bscscan.com/tx/0x9be572f03a14c97629e44303e2edc52da3ae1ba574cca69d0ad54e0f9e70181e>

4- **Set Taxes** (passed):

<https://testnet.bscscan.com/tx/0x5fc6bcfa3aea53239a8323666369229a3d3d7f94ceecb3a686c1d565afc9f9cc>

MANUAL TESTING

Centralization - Missing Zero Address

Severity: Low

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 setMarketingPool(address _marketingPool)
public onlyOperator {
    marketingPool = _marketingPool;
    emit ChangeMarketingPool(_marketingPool);
}
```

Suggestion:

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

MANUAL TESTING

Centralization - Local Variable Shadowing

Severity: Low

Function: _approve and allowance

Status: Open

Overview:

```
function allowance(address owner, address spender)
public override view returns (uint256) {
return _allowances[owner][spender];
}
function _approve (address owner, address spender,
uint256 amount) internal {
require(owner != address(0), 'BEP20: approve from the
zero address');
require(spender != address(0), 'BEP20: approve to the
zero address');
    _allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}
```

Suggestion:

Rename the local variable that shadows another component.

MANUAL TESTING

Optimization

Severity: Low

Subject: Old Pragma Solidity version

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 to avoid them.

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

Suggestion:

To reduce high gas fees. It is suggested to remove unused code from the contract.



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>
