auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

KNUCKLES

DATED : 14 MAY 23'

AUDIT SUMMARY

Project name – KNUCKLES

Date: 14 May, 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	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1.Manual Review: The code has undergone a line-by-line review by the **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/token/0x5d7fd00ebdd22efdaece3fd6df212202144cc1e3

Token Information

Name : Knuckles Inu

Symbol : KNUCKLES

Decimals: 9

Network: BSC

Token Type: BEP20

Token Address : ---

Owner: ---**(at time of writing the audit)**

Deployer:---

Token Information

Fees:

Buy Fees:

Sell Fees:

Transfer Fees:

Fees Privilige:

Ownership :

Minting: None

Max Tx Amount/ Max Wallet Amount:

Blacklist:

Other Priviliges:

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.

auditace.tech

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

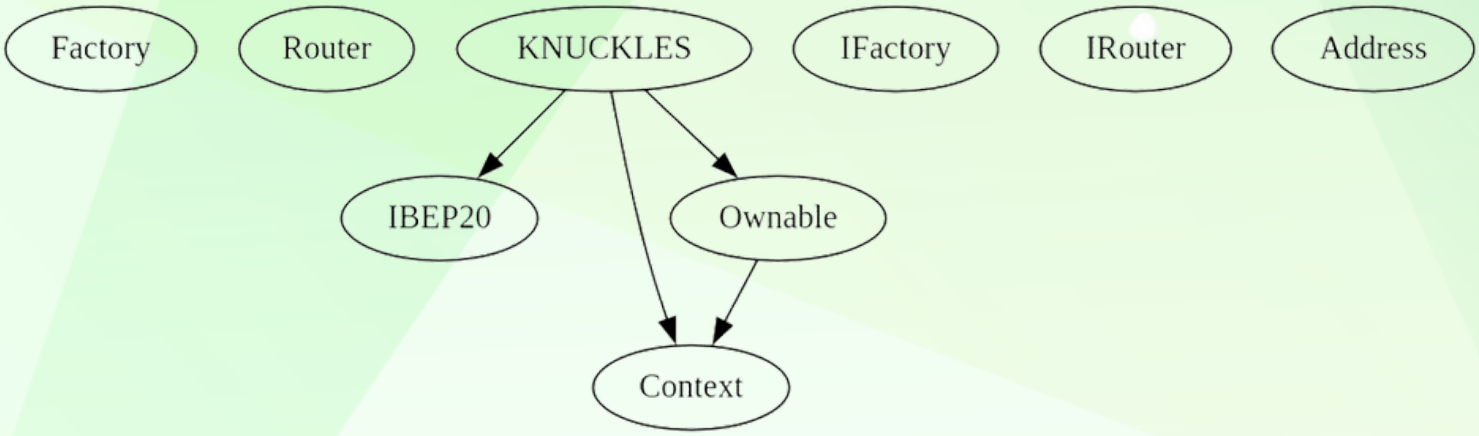
CLASSIFICATION OF RISK

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	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change buy/sell fees over 12.5% and transfer fee over 5%**
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens
- Owner must enable trades manually

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Factory	Interface		
└	createPair	External ❗	🛑	NO ❗
Router	Interface		
└	WETH	External ❗		NO ❗
└	factory	External ❗		NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
IBEP20	Interface		
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Context	Implementation		
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context	
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IFactory	Interface		
└	createPair	External ❗	🛑	NO ❗
IRouter	Interface		
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
Address	Library		
└	sendValue	Internal 🔒	🛑	

CONTRACT ASSESMENT

KNUCKLES	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	isExcludedFromReward	Public ❗		NO ❗
└	reflectionFromToken	Public ❗		NO ❗
└	EnableTrading	External ❗	🛑	onlyOwner
└	updateBuyTaxes	Public ❗	🛑	onlyOwner
└	updateSellTaxes	Public ❗	🛑	onlyOwner
└	updateTransferTaxes	Public ❗	🛑	onlyOwner
└	tokenFromReflection	Public ❗		NO ❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		NO ❗
└	_reflectRfi	Private 🔐	🛑	
└	_takeBuyback	Private 🔐	🛑	
└	_takeMarketing	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues1	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	InternalSwap	Internal 🔒	🛑	LockSwap
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Address.sendValue(address,uint256) (contracts/Token.sol#143-153) is never used and should be removed
Context._msgData() (contracts/Token.sol#63-66) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

KNUCKLES._rTotal (contracts/Token.sol#173) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#143-153):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#148)
Low level call in KNUCKLES.InternalSwap() (contracts/Token.sol#654-694):
        - (success) = address(owner()).call{value: ethReceived}() (contracts/Token.sol#675)
        - (success) = address(marketingWallet).call{value: marketingShare}() (contracts/Token.sol#684-686)
        - (success) = address(buyBackWallet).call{value: buybackShare}() (contracts/Token.sol#690-692)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Router.WETH() (contracts/Token.sol#13) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#119) is not in mixedCase
Struct KNUCKLES.valuesFromGetValues (contracts/Token.sol#202-212) is not in CapWords
Function KNUCKLES.EnableTrading() (contracts/Token.sol#363-366) is not in mixedCase
Function KNUCKLES.InternalSwap() (contracts/Token.sol#654-694) is not in mixedCase
Parameter KNUCKLES.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#710) is not in mixedCase
Parameter KNUCKLES.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#711) is not in mixedCase
Parameter KNUCKLES.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#712) is not in mixedCase
Constant KNUCKLES._decimals (contracts/Token.sol#169) is not in UPPER_CASE_WITH_UNDERSCORES
Constant KNUCKLES._name (contracts/Token.sol#178) is not in UPPER_CASE_WITH_UNDERSCORES
Constant KNUCKLES._symbol (contracts/Token.sol#179) is not in UPPER_CASE_WITH_UNDERSCORES
Modifier KNUCKLES.LockSwap() (contracts/Token.sol#220-224) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#64)" inContext (contracts/Token.sol#58-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable KNUCKLES.InternalSwap().success_scope_0 (contracts/Token.sol#684) is too similar to KNUCKLES.InternalSwap().success_scope_1 (contracts/Token.sol#690)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

KNUCKLES._getTValues(uint256,bool,address,address) (contracts/Token.sol#523-546) uses literals with too many digits:
        - s.tRfi = (tAmount * temp.rfi) / 100000 (contracts/Token.sol#541)
KNUCKLES._getTValues(uint256,bool,address,address) (contracts/Token.sol#523-546) uses literals with too many digits:
        - s.tBuyback = (tAmount * temp.buyback) / 100000 (contracts/Token.sol#542)
KNUCKLES._getTValues(uint256,bool,address,address) (contracts/Token.sol#523-546) uses literals with too many digits:
        - s.tMarketing = (tAmount * temp.marketing) / 100000 (contracts/Token.sol#543)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

KNUCKLES._tTotal (contracts/Token.sol#172) should be constant
KNUCKLES.buyBackWallet (contracts/Token.sol#175) should be constant
KNUCKLES.marketingWallet (contracts/Token.sol#176) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

KNUCKLES.pair (contracts/Token.sol#226) should be immutable
KNUCKLES.swapRouter (contracts/Token.sol#227) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Static Analysis

an static analysis of the code were performed using slither. No issues were found

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0xd49fecaa8572d99b96a1951024011b6942b0f5e3e986ea690a37d166f770f583

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x6b4a7ae38895a2e95bb9b505b59bcb9c8efb85986a698f9cf6937f1338b24e71

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf8af3ba997e4a1f03bf6d9fe4dbe5036563f63bc456b8e21b83565e33671d13c

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xd5521dc30ac3f01df497088943bf15a2af8e1633b2f733faf480683a1a9ef854

5- Buying when not excluded from fees (0-12.5% tax) (passed):
https://testnet.bscscan.com/tx/0x836e679faa39b6d024e19e7d597dca8b4f5ee2e6e3abb07a0745ee66a280c65f

6- Selling when not excluded from fees (0-12.5% tax) (passed):
https://testnet.bscscan.com/tx/0x3200b3fe1ff4161070391d1e29a7388eae0f5052716f59dc6a6dfc399d3abd42

7- Transferring when not excluded from fees (0-5% tax) (passed):
https://testnet.bscscan.com/tx/0x61870d3f8806e5410d01c137212a22db3f75235f9d587defc6880e42a337f66c

FUNCTIONAL TESTING

8- Internal swap (marketing + buyback) (passed):
https://testnet.bscscan.com/tx/0xdf6598276ed65dd54273a1ec76e9350cc4f00ffac2a38acb07267ca4576c6ccf

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: Medium

function: EnableTrading

Status: Not Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    swapEnabled = true;
    genesis_block = block.number;
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

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