auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Hulk

DATED : 08 Mar 23'

AUDIT SUMMARY

Project name – Hulk

Date: 08 March, 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	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 done on BSC Test network, each test has its transaction has attached to it.

3- Slither : Static Analysis

Testnet Link: all tests were done using this contract, tests are done on BSC Testnet

https://testnet.bscscan.com/token/0xbBc688FD6B78C4ce076197eBcD48D2ec28739Fe9

Token Information

Token Name : Hulk Token

Token Symbol: Hulk

Decimals: 18

Token Supply: 200,000,000

Token Address:
0x85D128F0415AE50683a972732f98C168CE3ced6f

Checksum:
cb8957a126d502f6f5c9e873a5ecf32e8941eb1c

Owner:
0xEf0c5850ADCdc4520F004B1Da31B64Aba4a8801F

TOKEN OVERVIEW

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees: 10%

Fees Privilige: Owner

Ownership : Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: setting max buy/sell/transfer - changing fee - manual buybacks

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

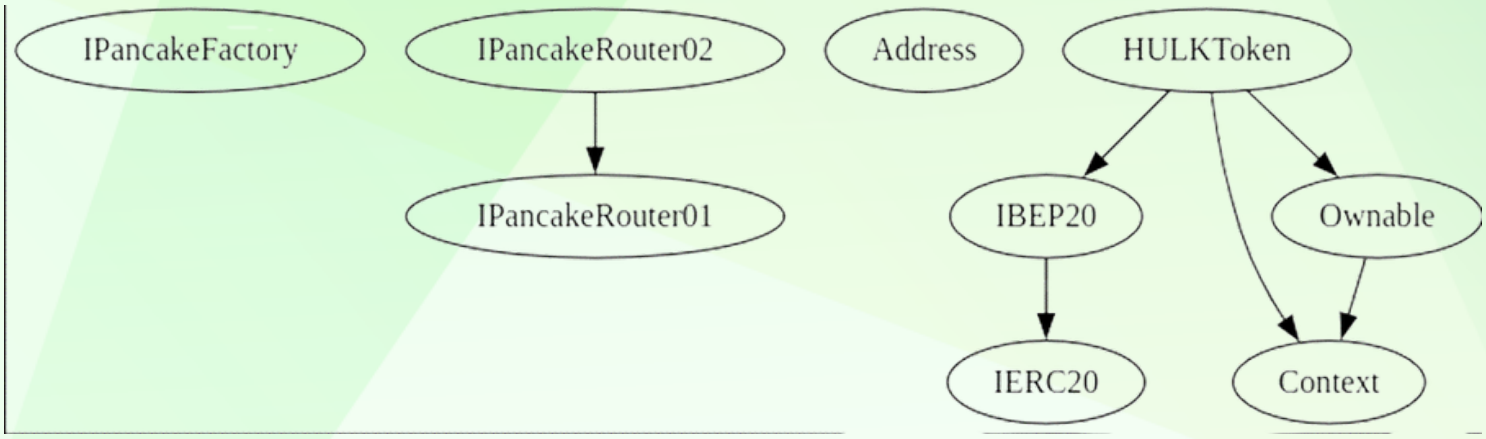
CLASSIFICATION OF RISK

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	0
Low-Risk	2
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to disable trades
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to mint new tokens
- Owner is able to set buy/sell/transfer fees each one up to 15% (30% buy + sell max)
- Owner is able to set max buy/sell/transfer amount to 1/1000 of total supply

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IPancakeFactory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
IPancakeRouter01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IPancakeRouter02	Interface	IPancakeRouter01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗

CONTRACT ASSESMENT

Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	_verifyCallResult	Private 🔐		
IERC20	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IBEP20	Interface	IERC20		
└	getOwner	External ❗		NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
HULKToken	Implementation	Context, IBEP20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	getOwner	External ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗

CONTRACT ASSESMENT

└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	isExcludedFromReward	Public ❗		NO❗
└	totalFees	Public ❗		NO❗
└	minimumTokensBeforeSwapAmount	Public ❗		NO❗
└	reflectionFromToken	Public ❗		NO❗
└	tokenFromReflection	Public ❗		NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	isExcludedFromAntiWhale	Public ❗		NO❗
└	setExcludedFromAntiWhale	Public ❗	🛑	onlyOwner
└	isIncludedInHulkLpList	Public ❗		NO❗
└	setIncludeInHulkLpList	Public ❗	🛑	onlyOwner
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	setMinimumBalanceRequired	Public ❗	🛑	onlyOwner
└	swapTokens	Private 🔐	🛑	lockTheSwap
└	buyBackTokens	Private 🔐	🛑	lockTheSwap
└	swapTokensForEth	Private 🔐	🛑	
└	swapETHForTokens	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	_transferStandard	Private 🔐	🛑	
└	_transferToExcluded	Private 🔐	🛑	
└	_transferFromExcluded	Private 🔐	🛑	
└	_transferBothExcluded	Private 🔐	🛑	
└	_reflectFee	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_takeLiquidity	Private 🔐	🛑	
└	calculateTaxFee	Private 🔐		
└	calculateLiquidityFee	Private 🔐		

CONTRACT ASSESMENT

└	removeAllFee	Private 🔐	🛑	
└	restoreAllFee	Private 🔐	🛑	
└	isExcludedFromFee	Public ❗		NO❗
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	setTaxFeePercent	External ❗	🛑	onlyOwner
└	setLiquidityFeePercent	External ❗	🛑	onlyOwner
└	setMaxTxAmount	External ❗	🛑	onlyOwner
└	setNumTokensSellToAddToLiquidity	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└	setBuyBackEnabled	Public ❗	🛑	onlyOwner
└	buyBackAndBurn	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Address._verifyCallResult(bool,bytes,string) (contracts/Token.sol#474-495) is never used and should be removed
Address.functionCall(address,bytes) (contracts/Token.sol#331-336) is never used and should be removed
Address.functionCall(address,bytes,string) (contracts/Token.sol#344-350) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/Token.sol#363-375) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/Token.sol#383-400) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/Token.sol#444-454) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/Token.sol#462-472) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/Token.sol#408-418) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/Token.sol#426-436) is never used and should be removed
Address.isContract(address) (contracts/Token.sol#270-281) is never used and should be removed
Address.sendValue(address,uint256) (contracts/Token.sol#299-311) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#11) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/Token.sol#45) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/Token.sol#197) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.17 (contracts/Token.sol#247) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.18 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#299-311):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#306)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/Token.sol#383-400):
        - (success,returndata) = target.call{value: value}(data) (contracts/Token.sol#396-398)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/Token.sol#426-436):
        - (success,returndata) = target.staticcall(data) (contracts/Token.sol#434)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/Token.sol#462-472):
        - (success,returndata) = target.delegatecall(data) (contracts/Token.sol#470)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IPancakeRouter01.WETH() (contracts/Token.sol#50) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#55) is too similar to IPancakeRouter01.addLiquidit
y(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#56)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
```

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

No major issues were found in the output

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All functionalities of the token appear to be working as intended. We have examined the smart contract's code and implemented various test cases for:

- Buying (excluded and not excluded from fees)
- Selling (excluded and not excluded from fees)
- Transferring (excluded and not excluded from fees)
- Internal Swap (if contract balance is more than threshold)
- BuyBack
- Reflections

our analysis did not reveal any potential issues or security flaws in process of this scenarios.

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x53098417d1007ffeddc1058e832f3c3f53aee7dbc3f9a186bf7a0ecc38c482a3

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x53098417d1007ffeddc1058e832f3c3f53aee7dbc3f9a186bf7a0ecc38c482a3

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x16fa9056b478b17632cc3487203d094b371071dae695178002ac6f25ae2a0eef

4- Transferring when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x66934a6507b4b779f8a962c930e613d071b91afa8f8129e63c71df6ac6b73d7d

FUNCTIONAL TESTING

5- Buying when not excluded (10% tax) (passed):
https://testnet.bscscan.com/tx/0xf3ae70b667301c25ec9b2a8279c3f473729f214a5bbe892b3619d3b3f1ba0881

6- Selling when not excluded (10% tax) (passed):
https://testnet.bscscan.com/tx/0xa0c3ade3bb28143eb83eb160b3ef8c404137dbbecfbb8b5c12218c6e64675090

7- Transferring when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xbb0e172b8e1e7ca8b5a69a46b6dc3e969b94a0bbb9a888b10968ad8fc07fa714

8-Internal swap (passed):
Contract received BNB during above sell, this means internal swap works
https://testnet.bscscan.com/address/0xbBc688FD6B78C4ce076197eBcD48D2ec28739Fe9#internaltx

9- Buyback using contract's BNB balance (passed):
Successfully bought, the tokens were sent to dead wallet (burning)
https://testnet.bscscan.com/tx/0x70082d61fae12dda998abb78020436c42f34c7829d74ec12eaf86fbf5cf7a723

10- Reflections (passed):
for testing this feature, we sent tokens to:
0xfc794e63f9677fa1fc499b42e8f135750fa833a1
after each sell from another wallet, balance of this account increased (received reflections), until we excluded it from rewards.
https://testnet.bscscan.com/tx/0xe202448f72ecd88879d7d81d5b0706dc10124f8ae72a5ff1539b9f6b1c14d852

MANUAL TESTING

Low Risk Issue

Issue: modifiable fees

Type : Centralization

Function: **setLiquidityFeePercent - setTaxFeePercent**

Line: 1072-1084

Severity: Low

Overview:

Overview: setLiquidityFeePercent , setTaxFeePercent functions allow the owner of the contract to change the tax fee and liquidity fee up to 5% and 10%, respectively. The token design can include a max 15% fee on buy and a 15% fee on sell, totaling a 30% fee. This is non-compliant with the SAFU (Secure Asset Fund for Users) criteria of Pinksale, which recommends a maximum total fee of 12.5% (25% on a buy+ sell max).

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner {
    require(taxFee <= MAX_TAX_FEE, "taxFee is too high");
    _taxFee = taxFee;

    emit UpdateTaxFee(taxFee);
}

ftrace | funcSig
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner {
    require(liquidityFee <= MAX_LIQUIDITY_FEE, "liquidityFee is too high");
    _liquidityFee = liquidityFee;

    emit UpdateLiquidityFee(liquidityFee);
}
```

MANUAL TESTING

Recommendations

To address the non-compliance with Pinksale SAFU criteria, we recommend:

- Reducing fees: The token's governance should consider reducing the fees to comply with **Pinksale's SAFU** criteria. This will help improve the token's attractiveness to investors and increase its liquidity.
- Increase transparency: To increase transparency, we recommend adding a function to the contract that allows anyone to view the current tax fee and liquidity fee. This could be done by adding public functions that return the current values of the _taxFee and _liquidityFee variables.

MANUAL TESTING

Low Risk Issue

IIssue: max tx

Type : Centralization

Function:**setMaxTxAmount**

Line: 1075-1080

Severity: Low

Overview: Owner is able to set a max for buy/sell/transfer amount, but this amount has a minimum limit of 1/1000 of total supply.

```
function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner {
    require(
        maxTxAmount >= MIN_TX_AMOUNT_HARD_CAP,
        "maxTxAmount have to be greater than 200000 HULK"
    );
    _maxTxAmount = maxTxAmount;

    emit UpdateMaxTxAmount(maxTxAmount);
}
```

Recommendations

To address the non-compliance with Pinksale SAFU criteria, we recommend:

- Reducing fees: The token's governance should consider reducing the fees to comply with **Pinksale's SAFU** criteria. This will help improve the token's attractiveness to investors and increase its liquidity.
- Increase transparency: To increase transparency, we recommend adding a function to the contract that allows anyone to view the current tax fee and liquidity fee. This could be done by adding public functions that return the current values of the _taxFee and _liquidityFee variables.

MANUAL TESTING

Suggestions & Recommendations

Issue: Stuck BNB in the contract

Type : Suggestion

Function: ---

Line: ---

Severity: Informational

Overview:

token collects fees in **BNB** and sends them to the token's address. The collected fees cannot be withdrawn by any means, and they can only be used to buyback the token. The purpose of this report is to assess the security implications of this design decision, taking into account the possibility that it might be intentional.

Recommendation:

- **Implement a withdrawal mechanism**: If the token's team wishes to allow for greater flexibility in the use of the collected fees, we recommend implementing a withdrawal mechanism that allows the token owner to withdraw the collected **BNB** from the contract. This could be done by adding a function to the contract that allows the owner to transfer the collected BNB to a designated wallet address.

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