auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Beagle Inu

DATED : 20 July 23'

MANUAL TESTING

Logical – 0 swap threshold can disable trades

Severity: High
function: updateSwapTokensAtAmount
Status: Not Resolved
Overview:
if swapTokensAtAmount is set to 0, internal swap would be failed due to a division by zero error. (swapAndLiquify function)

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(amount <= 42e14, "Cannot set swap threshold amount higher than 1% of tokens");
    swapTokensAtAmount = amount * 10 ** _decimals;
}
```

Suggestion

To mitigate this Logical issue, make sure that swapTokensAtAmount is always greater than 0

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(amount <= 42e14, "Cannot set swap threshold amount higher than 1% of tokens");
    require(amount >= 1, "Cannot set swap threshold amount higher than 1% of tokens");
    swapTokensAtAmount = amount * 10 ** _decimals;
}
```

MANUAL TESTING

Centralization – swaps are disabled by default

Severity: High
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

AUDIT SUMMARY

Project name – Beagle Inu

Date: 20 July, 2023

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed with High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	0	0	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/token/0x7ddc83e0abFDdDB2d7c0f8c93f1056166ecd88b3

Token Information

Token Name : Beagle Inu

Token Symbol: BEA

Decimals: 9

Token Supply: 420,000,000,000,000,000

Token Address:
0x46C370bfefC7d1edb811B59aC59687471b95809F

Checksum:
573bf4eed1106f9b77a1a8ef24d9cac562ed5992

Owner:
0xf3dC32eef17F860E1B428103f108dC749D4384aC
(at time of writing the audit)

Deployer:
0xf3dC32eef17F860E1B428103f108dC749D4384aC

TOKEN OVERVIEW

Fees:

Buy Fees: 5%

Sell Fees: 5%

Transfer Fees: 5%

Fees Privilege: static fees

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: no

Blacklist: No

Other Privileges: Initial distribution of the tokens 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

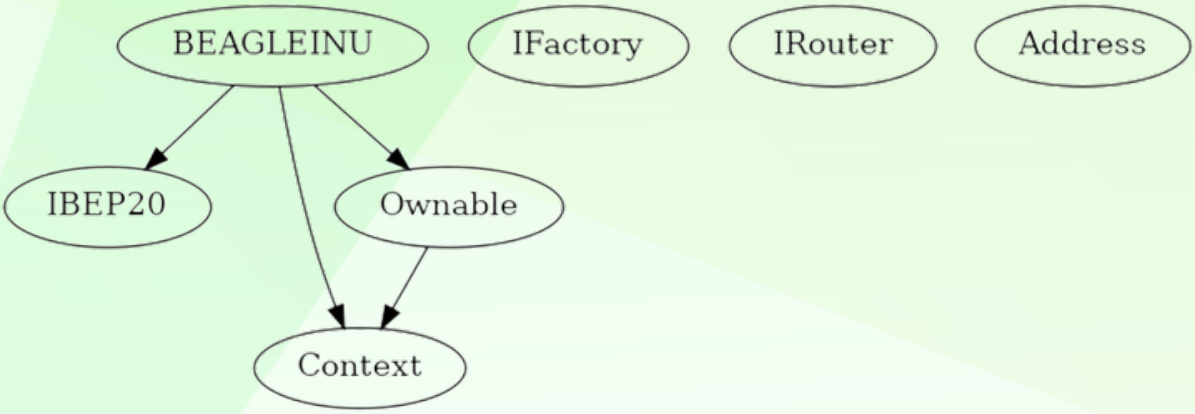
CLASSIFICATION OF RISK

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	2
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change fees (5% for each type of tax)**
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to mint new tokens
- **Owner must enable trades manually**

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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 ❗

CONTRACT ASSESMENT

Address	Library			
└	sendValue	Internal 🔒	🛑		
BEAGLEINU	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
└	updatedeadline	External ❗	🛑	onlyOwner
└	tokenFromReflection	Public ❗	NO ❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗	NO ❗
└	_reflectRfi	Private 🔐	🛑		
└	_takeLiquidity	Private 🔐	🛑		
└	_takeMarketing	Private 🔐	🛑		
└	_takeOps	Private 🔐	🛑		
└	_takeDev	Private 🔐	🛑		

CONTRACT ASSESMENT

└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues1	Private 🔐		
└	_getRValues2	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	addLiquidity	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateOpsWallet	External ❗	🛑	onlyOwner
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateSwapEnabled	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Context._msgData() (contracts/Token.sol#38-41) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

BEAGLEINU._rTotal (contracts/Token.sol#139) 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#13) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 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#106-111):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#85) is not in mixedCase
Struct BEAGLEINU.valuesFromGetValues (contracts/Token.sol#176-190) is not in CapWords
Function BEAGLEINU.EnableTrading() (contracts/Token.sol#299-304) is not in mixedCase
Parameter BEAGLEINU.updatedeadline(uint256)._deadline (contracts/Token.sol#306) is not in mixedCase
Parameter BEAGLEINU.updateSwapEnabled(bool)._enabled (contracts/Token.sol#642) is not in mixedCase
Parameter BEAGLEINU.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#653) is not in mixedCase
Parameter BEAGLEINU.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#653) is not in mixedCase
Parameter BEAGLEINU.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#653) is not in mixedCase
Constant BEAGLEINU._decimals (contracts/Token.sol#135) is not in UPPER_CASE_WITH_UNDERSCORES
Variable BEAGLEINU.genesis_block (contracts/Token.sol#143) is not in mixedCase
Constant BEAGLEINU._name (contracts/Token.sol#151) is not in UPPER_CASE_WITH_UNDERSCORES
Constant BEAGLEINU._symbol (contracts/Token.sol#152) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#39)" inContext (contracts/Token.sol#33-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

BEAGLEINU.slitherConstructorVariables() (contracts/Token.sol#114-659) uses literals with too many digits:
        - _tTotal = 420000000000000000 * 10 ** _decimals (contracts/Token.sol#138)
BEAGLEINU.slitherConstructorVariables() (contracts/Token.sol#114-659) uses literals with too many digits:
        - swapTokensAtAmount = 210000000000000 * 10 ** 9 (contracts/Token.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

BEAGLEINU._lastSell (contracts/Token.sol#130) is never used in BEAGLEINU (contracts/Token.sol#114-659)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

BEAGLEINU._tTotal (contracts/Token.sol#138) should be constant
BEAGLEINU.deadWallet (contracts/Token.sol#146) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

BEAGLEINU.pair (contracts/Token.sol#133) should be immutable
BEAGLEINU.router (contracts/Token.sol#132) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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

No major issues were found in the output

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xf400bbc671e5bb48fd3416ebf8e575b53ed1e04e01350157550988 3d63faab45

2- Buying when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x77c43a31fa1c94e699c448ca3780df9cb031f6349dd0ca068b44ab724fb78b0b

3- Selling when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x91ddd4a93a0fc91906f89b026c65f97c30cf87fd190f43a8effd5b72e0643595

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xf7de43faaf30f95ea31b99199891ca1507775a3c477972684cf09c9c82e2d09d

5- Buying when not excluded from fees (5% tax) (passed):

https://testnet.bscscan.com/tx/0xf809fed5434a79de423c461e76508082bd3c300c274dbb065cbd5700aa2cc58a

6- Selling when not excluded from fees (5% tax) (passed):

https://testnet.bscscan.com/tx/0x8dde84ed9c779fac88fd59b8da83919cddf59adef390318d5a77b8a8218de566

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (5% tax) (passed):

https://testnet.bscscan.com/tx/0xf5acde523c5ec3463020588a7c28ab46ca7aecada161ca71b5e41391f574c8fd

8- Internal swap (marketing wallet received BNB) (passed):

https://testnet.bscscan.com/address/0xb1ca3e7993f3d1ad8e2af28aa0de6fd49dc4a32b#internaltx

MANUAL TESTING

Logical – 0 swap threshold can disable trades

Severity: High

function: updateSwapTokensAtAmount

Status: Not Resolved

Overview:

if swapTokensAtAmount is set to 0, internal swap would be failed due to a division by zero error. (swapAndLiquify function)

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(amount <= 42e14, "Cannot set swap threshold amount higher than 1% of tokens");
    swapTokensAtAmount = amount * 10 ** _decimals;
}
```

Suggestion

To mitigate this Logical issue, make sure that swapTokensAtAmount is always greater than 0

```
function updateSwapTokensAtAmount(uint256 amount) external onlyOwner {
    require(amount <= 42e14, "Cannot set swap threshold amount higher than 1% of tokens");
    require(amount >= 1, "Cannot set swap threshold amount higher than 1% of tokens");
    swapTokensAtAmount = amount * 10 ** _decimals;
}
```

MANUAL TESTING

Centralization – swaps are disabled by default

Severity: High
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