auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

BarbiePEPE

DATED : 6 May 23'

AUDIT SUMMARY

Project name – BarbiePEPE

Date: 6 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	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/address/0x1a1040bf0a182aa4e85944d4b713361d8fef37a6#code

Token Information

Token Name : BarbiePEPE

Token Symbol: BPEPE

Decimals: 9

Token Supply: 1,000,000,000

Token Address:
0x55A02AD006CfaF71FEF8D825720A76DFA9fb82F6

Checksum:
80b4b14b6f2ec91de8765d5ba8fe52cf73411863

Owner:
0x0000000000000000000000000000000000000000
(Renounced)

Deployer:
0xb66ea3B473B484a94a7A013C7B834805540E0A63

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: None

Ownership: renounced

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: 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 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

- 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	0
Gas Optimization / Suggestions	0

INHERITANCE TREE

POINTS TO NOTE

- Owner is not able to set buy/sell/transfer taxes (0% all)
- Owner is not able to set a max buy/transfer/wallet/sell amount
- Owner is able to blacklist an arbitrary wallet
- Owner is able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
BarbiePepe	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	approve	Public ❗	🛑	NO ❗
└	destroyTokens	Public ❗	🛑	NO ❗
└	renounceOwnership	Public ❗	🛑	NO ❗
IUniswapV2Router01	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 ❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
UniswapV2Caller	Implementation			
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗

CONTRACT ASSESMENT

ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🛑	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	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 ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_spendAllowance	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
BarbiePepe.totalSupply (contracts/Token.sol#26) is set pre-construction with a non-constant function or state variable:
        - numberOfCoins * 10 ** decimalls
BarbiePepe.decimals (contracts/Token.sol#27) is set pre-construction with a non-constant function or state variable:
        - decimalls
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#7) 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

BarbiePepe.slitherConstructorVariables() (contracts/Token.sol#9-90) uses literals with too many digits:
        - numberOfCoins = 1000000000 (contracts/Token.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

BarbiePepe.decimalls (contracts/Token.sol#17) should be constant
BarbiePepe.name (contracts/Token.sol#13) should be constant
BarbiePepe.numberOfCoins (contracts/Token.sol#16) should be constant
BarbiePepe.symbol (contracts/Token.sol#14) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

BarbiePepe.decimals (contracts/Token.sol#27) 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

Router (PCS V2):

0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x20d61dc0c214336c7dbe3fab7f793dc40f7169a72e543415d99740d8f6e7661a

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

https://testnet.bscscan.com/tx/0xbf1477580da02618d567d32ace81aeadbd3f5f46f09dc9260b64f57bf25cfcd7

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

https://testnet.bscscan.com/tx/0x3469bab8f6027b2ac3d47ed60fb233b3bcfc9c6c3755e2a567a99611a574ae66

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

https://testnet.bscscan.com/tx/0x97923e55aec7982990e58a7732cbaf228c5fb9fe36936844793530 1f8e784ee4

5- Buying when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x64073d8493bac01dfd0b339e09ca0c545ee86c15b2040c090db2a46c8592d880

6- Selling when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x4367193abe04670b6b3b7109171037c4344833a7ebf84f080f1ba9a8a8ed34fb

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xcda0ae42f9436f333160d2a7d6343308e14becf83f2202434856856224ead51c

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