auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Pepe Shrek

DATED : 18 MAY 23'

AUDIT SUMMARY

Project name – PEPESHREK

Date: 18 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	1	1
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/0x0a780a84865c72b21dd74dfddd6ae4ba58729d55

Token Information

Name : Pepe Shrek

Symbol : PEPESHREK

Decimals: 9

Network: BSC

Token Type: BEP20

Token Address:
0xe6CBFE6cce0925Eb3BFacF62cF7876fC38074a28

Owner:
0x79b78ccE0655FD3F18692a12f5239D8292278fC6
(at time of writing the audit)

Deployer:0x79b78ccE0655FD3F18692a12f5239D8292278fC6

Token Information

Fees:

Buy Fees: 4%

Sell Fees: 4%

Transfer Fees: 4%

Fees Privilige: Static Fees

Ownership :

0xe6CBFE6cce0925Eb3BFacF62cF7876fC38074a28

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: - initial distribution of the tokens

- including in fees

- excluding from fees

- changing swap threshold

- withdrawing stuck BNB or tokens from the contract (except native token)

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

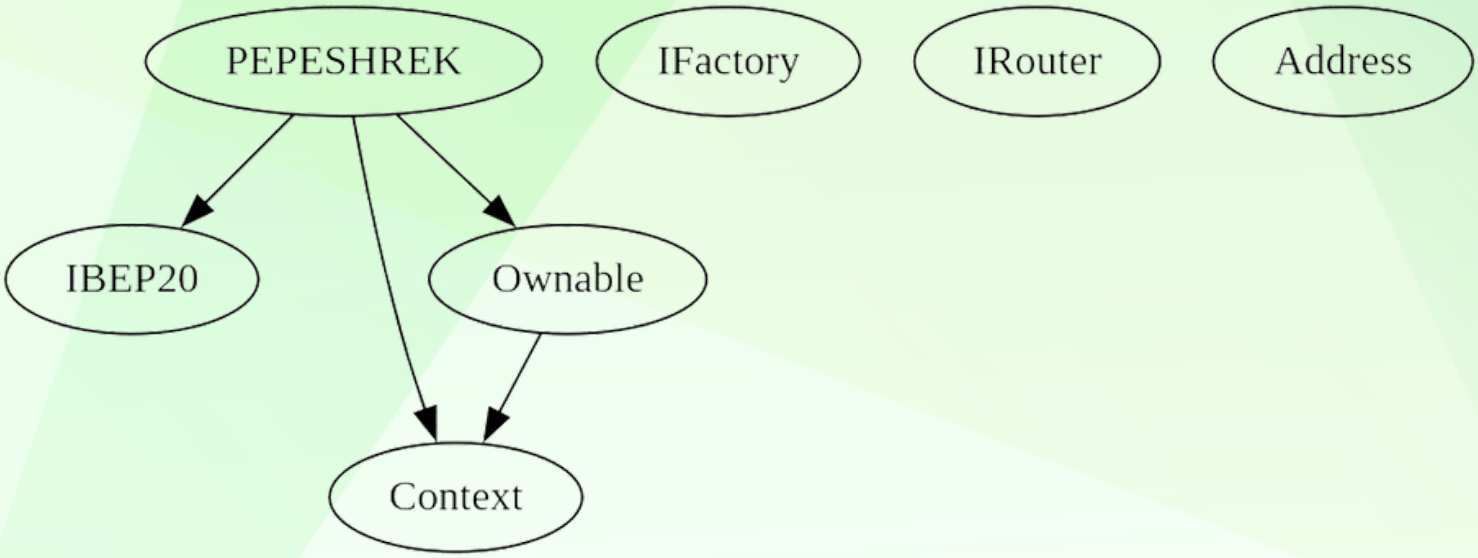
CLASSIFICATION OF RISK

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	1
Low-Risk	1
Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to set limit for buy/sell/transfer/holding amounts
- Owner is not able to mint new tokens
- Owner is not able to disable trades
- Contract uses a static 2% fee for **Reflections** and a 2% fee for **Marketing**
- Contract has a function to swap tokens for BNB and send the BNB to the **marketing wallet**
- Contract has a function to exclude or include addresses from fees
- Contract has a function to update the marketing wallet address
- Contract has a function to update the swap tokens at amount threshold
- Contract has a function to rescue any BEP20 tokens sent to the contract by mistake (except native token)
- Contract has a function to rescue BNB sent to the contract by mistake
- Contract has a function to bulk exclude or include addresses from fees
- Contract has a receive() function to accept BNB payments

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 ❗
Address	Library			
└	sendValue	Internal 🔒	🛑	
PEPESHREK	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 ❗

CONTRACT ASSESMENT

└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	isExcludedFromReward	Public ❗		NO ❗
└	reflectionFromToken	Public ❗		NO ❗
└	tokenFromReflection	Public ❗		NO ❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	isExcludedFromFee	Public ❗		NO ❗
└	_reflectRfi	Private 🔐	🛑	
└	_takeMarketing	Private 🔐	🛑	
└	_getValues	Private 🔐		
└	_getTValues	Private 🔐		
└	_getRValues	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForBNB	Private 🔐	🛑	
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	updateMarketingWallet	External ❗	🛑	onlyOwner
└	updateSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Reentrancy in PEPESHREK.transferFrom(address,address,uint256) (contracts/Token.sol#274-289):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#279)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#131)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (contracts/Token.sol#560-566)
                - address(marketingWallet).sendValue(deltaBalance) (contracts/Token.sol#547)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#279)
                - (success) = recipient.call{value: amount}() (contracts/Token.sol#131)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#485)
                - _approve(sender,_msgSender(),currentAllowance - amount) (contracts/Token.sol#286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

PEPESHREK.includeInReward(address) (contracts/Token.sol#364-376) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#372)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Context._msgData() (contracts/Token.sol#45-48) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

PEPESHREK._rTotal (contracts/Token.sol#159) is set pre-construction with a non-constant function or state variable:
        - (MAX - (MAX % _tTotal))
PEPESHREK.swapTokensAtAmount (contracts/Token.sol#161) is set pre-construction with a non-constant function or state variable:
        - _tTotal / 5000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#6) 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#125-136):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IRouter.WETH() (contracts/Token.sol#101) is not in mixedCase
Struct PEPESHREK.valuesFromGetValues (contracts/Token.sol#188-196) is not in CapWords
Parameter PEPESHREK.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#603) is not in mixedCase
Parameter PEPESHREK.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#604) is not in mixedCase
Parameter PEPESHREK.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#605) is not in mixedCase
Constant PEPESHREK._decimals (contracts/Token.sol#155) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPESHREK._name (contracts/Token.sol#171) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PEPESHREK._symbol (contracts/Token.sol#172) 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#46)" inContext (contracts/Token.sol#40-49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

PEPESHREK._tTotal (contracts/Token.sol#158) should be constant
PEPESHREK.deadWallet (contracts/Token.sol#168) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PEPESHREK.pair (contracts/Token.sol#153) should be immutable
PEPESHREK.router (contracts/Token.sol#152) 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

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x415b3b18eb456d6eb5ebbc4411a519861b5d31f0dc697e758a074c6c2f3def3f

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x4a6c4566de565384ae0a592bd82cbda06779127534327fa6c0684dfe248a6c17

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x54cacc96390f70f75eb197c5848311ddcbac026781468f67a185135b47a0cf1f

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x7eba01bd8a46d66678368d47d7ad8653a833f11d74151c30f5b60e779e676c19

5- Buying when not excluded from fees (4% tax) (passed):
https://testnet.bscscan.com/tx/0xe8d84fc601ff98688e6b48a516faec8b77095ac3de372374a044c713f4e18ff2

6- Selling when not excluded from fees (4% tax) (passed):
https://testnet.bscscan.com/tx/0xaf0baf835c9add08bd86fa8c8950e0bcdc67c7418727462da7a2d508b3af3f10

7- Transferring when not excluded from fees (4% tax) (passed):
https://testnet.bscscan.com/tx/0x7f9c9f89a15e47ad78043bb262be331726586051c702a376963097bad271eca3

FUNCTIONAL TESTING

8- Internal swap (marketing bnb) (passed):
https://testnet.bscscan.com/tx/0xaf0baf835c9add08bd86fa8c8950e0bcdc67c7418727462da7a2d508b3af3f10

FUNCTIONAL TESTING

Category: Logical
Subject: Incorrect error message in **updateSwapTokensAtAmount** function
Severity: Low
Overview:
The error message in the updateSwapTokensAtAmount function states that the swap threshold amount cannot be higher than 1% of tokens. However, the require statement checks if the amount is less than or equal to 1e15, which is not 1% of the total supply.

Code:
```
require(amount <= 1e15, "Cannot set swap threshold amount higher than 1% of tokens");
```

Suggestion:
Update the condition to accuratly reflect the error message. Change the error message to:
require(amount <= totalSupply()/100/10**_decimals, "Cannot set swap threshold amount higher than 1% of tokens");

FUNCTIONAL TESTING

Category: **Centralization**

Subject: Ownership and control over contract functions

Severity: **Informational**

Status: **not applicable**

Overview:

The contract has an owner who has control over several functions that can affect the token's behavior and centralize control. The owner can exclude or include addresses from fees, change the marketing wallet, update the swap tokens amount, and rescue any BEP20 tokens (except the native token).

Code:

function excludeFromReward(address account) public onlyOwner;

function includeInReward(address account) external onlyOwner;

function excludeFromFee(address account) public onlyOwner;

function includeInFee(address account) public onlyOwner;

function bulkExcludeFee(address[] memory accounts, bool state) external onlyOwner;

function updateMarketingWallet(address newWallet) external onlyOwner;

function updateSwapTokensAtAmount(uint256 amount) external onlyOwner;

function rescueBNB(uint256 weiAmount) external onlyOwner;

function rescueAnyBEP20Tokens(address _tokenAddr,address _to, uint256 _amount) public onlyOwner;

FUNCTIONAL TESTING

Suggestion:

To reduce centralization risks, consider implementing a decentralized governance system that allows token holders to vote on changes to the contract's behavior. This can include voting on fee changes, updating the marketing wallet, and other important decisions. Additionally, consider removing the ability to rescue any BEP20 tokens, as this can lead to potential misuse of funds.

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