auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Pepe 2.0

DATED : 29 June 23'

AUDIT SUMMARY

Project name – Pepe 2.0

Date: 29 June, 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/token/0x67aef90e4DAa94dE7BCDb37b5C81F4716b19570F

Token Information

Token Name : Pepe 2.0

Token Symbol: PEPE2.0

Decimals: 18

Token Supply: 420,690,000,000,000

Token Address:
0x1F06A61a1ee23e90038c5fBeBB46f74278181498

Checksum:
308e7784927442af6ce8baece4cd27d227f9ccb3

Owner:
0xa563e25789d3F29939a9250e0C560306e96453fA

Deployer:
0xa563e25789d3F29939a9250e0C560306e96453fA

TOKEN OVERVIEW

Fees:

Buy Fees: 1%

Sell Fees: 1%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: initial distribution of tokens

- including or excluding from fees

- changing swap threshold

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

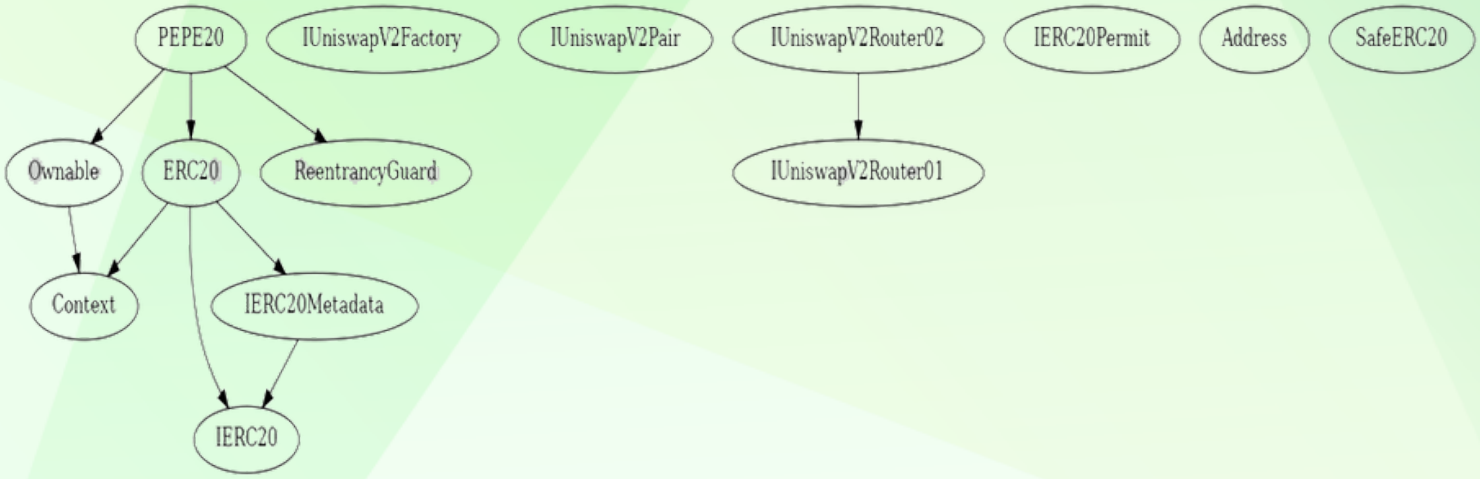
CLASSIFICATION OF RISK

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 change current taxes (1% on buy and sell)
- Owner is not able to set fee on transfers (0%)
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to set max wallet/transfer/buy/sell
- Owner is not able to mint new tokens

STATIC ANALYSIS

```
Address.functionStaticCall(address,bytes) (contracts/Token.sol#339-341) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/Token.sol#343-350) is never used and should be removed
Address.sendValue(address,uint256) (contracts/Token.sol#308-313) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (contracts/Token.sol#380-390) is never used and should be removed
Context._msgData() (contracts/Token.sol#14-16) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#594-609) is never used and should be removed
ReentrancyGuard._reentrancyGuardEntered() (contracts/Token.sol#698-700) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/Token.sol#433-439) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/Token.sol#446-453) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/Token.sol#441-444) is never used and should be removed
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/Token.sol#455-469) is never used and should be removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (contracts/Token.sol#429-431) 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#7) 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#308-313):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#311)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/Token.sol#330-337):
        - (success,returndata) = target.call{value: value}(data) (contracts/Token.sol#335)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/Token.sol#343-350):
        - (success,returndata) = target.staticcall(data) (contracts/Token.sol#348)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/Token.sol#356-362):
        - (success,returndata) = target.delegatecall(data) (contracts/Token.sol#360)
Low level call in PEPE20.sendBNB(address,uint256) (contracts/Token.sol#890-894):
        - (success) = address(_to).call{value: amount}() (contracts/Token.sol#892)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Token.sol#61) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Token.sol#63) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Token.sol#82) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/Token.sol#114) is not in mixedCase
Function IERC20Permit.DOMAIN_SEPARATOR() (contracts/Token.sol#300) is not in mixedCase
Parameter PEPE20.sendBNB(address,uint256)._to (contracts/Token.sol#890) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#119) is too similar to IUniswapV2Router01.addLiquidi
ty(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PEPE20.constructor() (contracts/Token.sol#732-762) uses literals with too many digits:
        - _mint(owner(),420690000000000 * (10 ** 18)) (contracts/Token.sol#733)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PEPE20.marketingWallet (contracts/Token.sol#711) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

PEPE20.buyTax (contracts/Token.sol#707) should be immutable
PEPE20.marketingWalletShares (contracts/Token.sol#713) should be immutable
PEPE20.sellTax (contracts/Token.sol#708) should be immutable
PEPE20.setSwapTokensLimit (contracts/Token.sol#716) should be immutable
PEPE20.taxDenominator (contracts/Token.sol#706) should be immutable
PEPE20.totalTax (contracts/Token.sol#709) should be immutable
PEPE20.uniswapV2Pair (contracts/Token.sol#721) should be immutable
PEPE20.uniswapV2Router (contracts/Token.sol#720) 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/0xcd236c7d331a790abf219cc24176723ce768c316c5b2bb67a868fb4e4c7aabec

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

https://testnet.bscscan.com/tx/0x939d513e94c2c0ccbe120019fabd2817f41dbe186a0ced96ddda25dc6cb53e12

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

https://testnet.bscscan.com/tx/0x52b981e2ae2eb6fcbd50ef21402fe1e4318c80c9f5fff4954ea9df2e0134dc2c

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

https://testnet.bscscan.com/tx/0x086515569bae8aebbd7564814accb2cd4451d8b24f4bb7bef7193b5e7bae552b

5- Buying when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0xb0d8f65c44a04ba988c89a12f7f79c81030ec10b0ad94a2ce5916077a063bebc

6- Selling when not excluded from fees (1% tax) (passed):

https://testnet.bscscan.com/tx/0x7fb8c1dcf633b640b80f1a4f3e8ffea06a77333bfe16c097331bd6bea0c6e67a

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x26a2e328a4f1c04bfcc80f3623f8e80d2ad875f80c6d7bd1675b4ae5ac15b0dc

8- Internal swap (Marketing wallet receiving BNB) (passed):
https://testnet.bscscan.com/address/0x84e126231e0a76a93fc4e887c447a57e29789f44#internaltx

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