auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PepeShiba

DATED : 6 May 23'

AUDIT SUMMARY

Project name – PepeShiba

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	1	2
Acknowledged	0	0	0	0	0
Resolved	0	1	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/0xa41a57a8518E59b4F3E1805A3916461E11f577BC

Token Information

Token Name : PepeShiba

Token Symbol: PepeShiba

Decimals: 9

Token Supply: 1,000,000,000

Token Address: --

Checksum:
80b4b14b6f2ec91de8765d5ba8fe52cf73411863

Owner: --

Deployer: --

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: None

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: 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

- 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

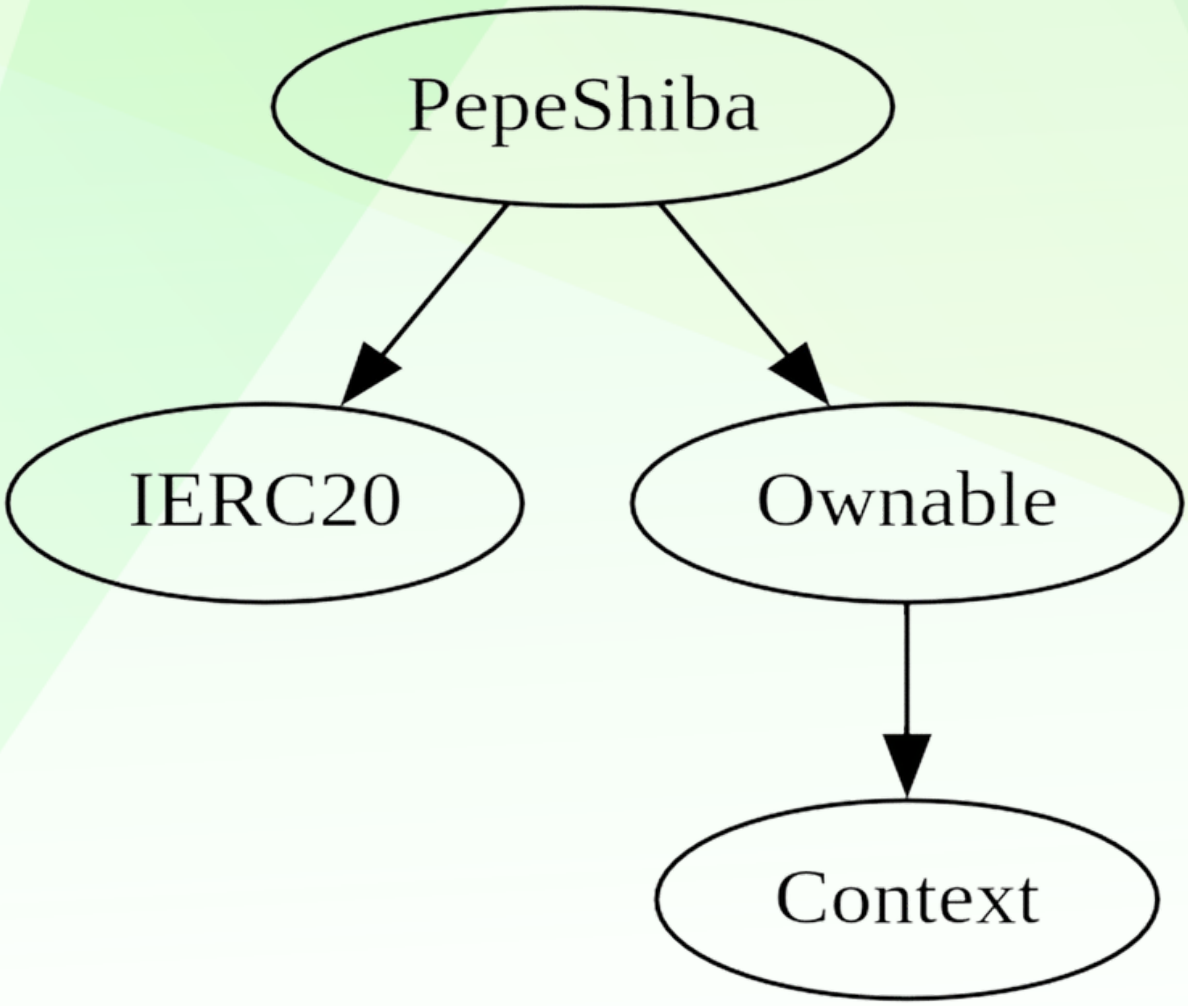
CLASSIFICATION OF RISK

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	1
Medium-Risk	0
Low-Risk	1
Gas Optimization / Suggestions	2

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 not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens
- Owner must enable trades for holders to be able to trade

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
PepeShiba	Implementation	IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	<Receive Ether>	External ❗	💵	NO ❗
└	totalSupply	External ❗		NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	External ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	_approve	Internal 🔒	🛑	
└	approveMax	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	_transferFrom	Internal 🔒	🛑	
└	_basicTransfer	Internal 🔒	🛑	
└	enableTrading	External ❗	🛑	onlyOwner
└	setAuthorizedWallets	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
PepeShiba._approve(address,address,uint256).owner (contracts/Token.sol#291) shadows:
        - Ownable.owner() (contracts/Token.sol#173-175) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.17']
        - ^0.8.0 (contracts/Token.sol#8)
        - ^0.8.0 (contracts/Token.sol#37)
        - ^0.8.0 (contracts/Token.sol#131)
        - ^0.8.17 (contracts/Token.sol#224)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (contracts/Token.sol#25-27) is never used and should be removed
PepeShiba._approve(address,address,uint256) (contracts/Token.sol#290-300) is never used and should be removed
PepeShiba._basicTransfer(address,address,uint256) (contracts/Token.sol#347-358) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (contracts/Token.sol#8) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#37) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#131) allows old versions
Pragma version^0.8.17 (contracts/Token.sol#224) 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

Parameter PepeShiba.setAuthorizedWallets(address,bool)._wallet (contracts/Token.sol#366) is not in mixedCase
Parameter PepeShiba.setAuthorizedWallets(address,bool)._status (contracts/Token.sol#367) is not in mixedCase
Constant PepeShiba._name (contracts/Token.sol#230) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepeShiba._symbol (contracts/Token.sol#231) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PepeShiba._decimals (contracts/Token.sol#232) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PepeShiba._totalSupply (contracts/Token.sol#234) is not in mixedCase
Variable PepeShiba._balances (contracts/Token.sol#236) is not in mixedCase
Variable PepeShiba._allowances (contracts/Token.sol#237) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

PepeShiba.slitherConstructorVariables() (contracts/Token.sol#229-378) uses literals with too many digits:
        - _totalSupply = 1000000000 * (10 ** _decimals) (contracts/Token.sol#234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

PepeShiba._totalSupply (contracts/Token.sol#234) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

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/0x7124d26e644ce1ee2dc2469a9e612946eee0dc19e12dddf0ba47b9eb5181105a

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

https://testnet.bscscan.com/tx/0xf50e16bec25f6e27fbc88bb2883d66464d0046515d42ac72cea3c5ad44d7d345

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

https://testnet.bscscan.com/tx/0x7574ac712291503fe380dc2cf097836961 5f73fea66b6b1fa15a311ebfe1b73a

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

https://testnet.bscscan.com/tx/0xca355d5b32ece04495ae942e521faeea25730dc2df9bbc02a67f084fa80c5cc0

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

https://testnet.bscscan.com/tx/0x350f083df5f98e9bb7175a71904e0c2e3012386df2761d0222acd34a6f79fb12

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

https://testnet.bscscan.com/tx/0xa198e355ce095e6489dd9f8afc486c3cc4e9398346fb5fc27fe0552721f09b9a

FUNCTIONAL TESTING

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

https://testnet.bscscan.com/tx/0x1bc001a2e7838499b74ddc83b1df4c7529d7e3db1181540b467df62050ce1bc9

MANUAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTrading

Status: Resolved (Contract is owned by Pinksale safu developer)

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(!isTradeEnabled, "Trading already enabled");
    isTradeEnabled = true;
```

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.
   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.

   Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades **(applied)**

MANUAL TESTING

Logical – Trades must be enabled

Severity: Low

function: _transferFrom

Status: Not Resolved

Overview:

Authorized wallets are not able to buy tokens before enabling of the trades

```
function _transferFrom(
    address sender,
    address recipient,
    uint256 amount
) internal returns (bool) {
    if (!isTradeEnabled) require(isAuthorized[sender], "Trading disabled");

    require(_balances[sender] >= amount, "Insufficient Balance");
    _balances[sender] = _balances[sender] - amount;

    _balances[recipient] = _balances[recipient] + amount;

    emit Transfer(sender, recipient, amount);
    return true;
```

Suggestion

To mitigate this logical issue, check if sender of recipient is Authorized or no
Authorized, allow transfer.

MANUAL TESTING

Informational – No way to withdraw stuck tokens

Status: Not Resolved

Overview:

Owner is not able to withdraw stuck ERC20 tokens from the contract. If some tokens are sent to the contract by mistake, those tokens will be stuck for ever.

Suggestion

Implement a function to be able to withraw ERC20 tokens from the contract:

```
function withrawERC20(
    address _tokenAddress,
    address _to,
    uint256 _amount
) external onlyOwner {
    IERC20(_tokenAddress).transfer(_to, _amount);
}
```

MANUAL TESTING

Informational – Redundant code

Status: Not Resolved

Overview:

Some functoins of the contract are never used, this functions are “internal” hence, not being used in other external/public functions means this functinos are redundant.

Suggestion

Delete below functions:

_basicTransfer

_approve

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