auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

POTTERPEPE

DATED : 27 MAY 23'

AUDIT SUMMARY

Project name – POTTERPEPE

Date: 27 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	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.ETH Test Network: All tests were conducted on the ETH 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/0x9c98932c84A5F73855267720d0d4FD96c126A58B

Token Information

Name : Potter Pepe

Symbol : POTTERPEPE

Decimals: 18

Network: BSC

Token Type:BEP20

Token Address:
0x2AE1b523b6E9Bf8D807B370BBc2A8368BbbA353a

Owner:
0x43fA3d8C26bd82BE2DD3743176263B9D7c2e4f1D
(at time of writing the audit)

Deployer:0x854a73a5434aC0A9899751E4Fd102ba87940005f

Token Information

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0-5%

Fees Privilige: owner

Ownership :

0x43fA3d8C26bd82BE2DD3743176263B9D7c2e4f1D

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges:- including in fees

- excluding from fees

- initial distribution of the tokens

- modifying fees

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

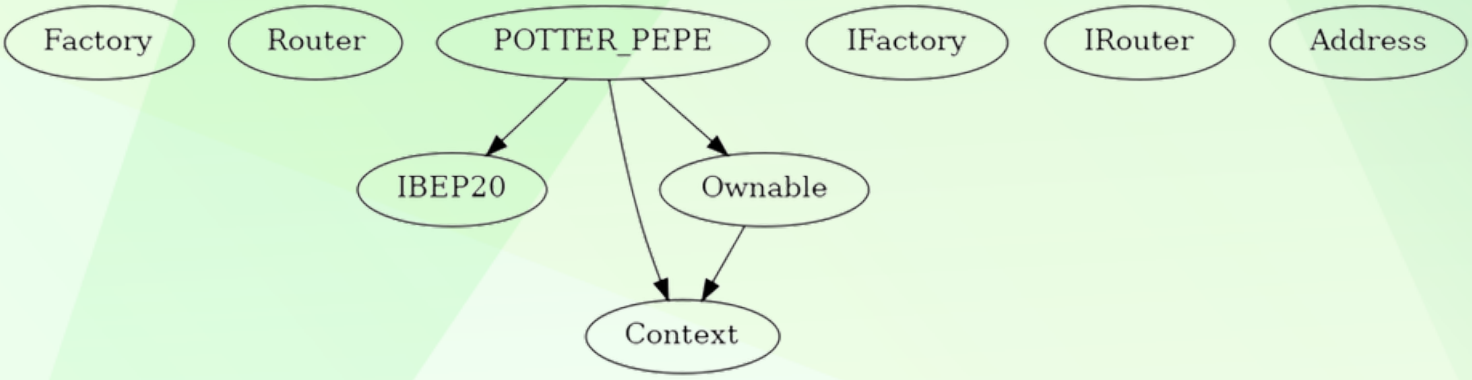
CLASSIFICATION OF RISK

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	1

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to change buy/sell fees over 12.5% and transfer fee over 5%
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens
- Owner must enable trades manually

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Factory	Interface			
└	createPair	External ❗	🛑	NO ❗
Router	Interface			
└	WETH	External ❗		NO ❗
└	factory	External ❗		NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
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 🔒	🛑	
POTTER_PEPE	Implementation	Context, IBEP20, Ownable		

CONTRACT ASSESMENT

└	<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
└	updateBuyTaxes	Public ❗	🛑	onlyOwner
└	updateSellTaxes	Public ❗	🛑	onlyOwner
└	updateTransferTaxes	Public ❗	🛑	onlyOwner
└	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 🔐		
└	_getRValues1	Private 🔐		
└	_getRate	Private 🔐		
└	_getCurrentSupply	Private 🔐		
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	_tokenTransfer	Private 🔐	🛑	
└	InternalSwap	Internal 🔒	🛑	LockSwap
└	bulkExcludeFee	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueAnyBEP20Tokens	Public ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

CONTRACT ASSESMENT

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
POTTER_PEPE.includeInReward(address) (contracts/Token.sol#409-420) has costly operations inside a loop:
        - _excluded.pop() (contracts/Token.sol#416)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop

Address.sendValue(address,uint256) (contracts/Token.sol#143-153) is never used and should be removed
Context._msgData() (contracts/Token.sol#63-66) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

POTTER_PEPE._rTotal (contracts/Token.sol#173) 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#3) 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#143-153):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#148)
Low level call in POTTER_PEPE.InternalSwap() (contracts/Token.sol#595-615):
        - (success) = address(marketingWallet).call{value: address(this).balance}() (contracts/Token.sol#612-614)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Router.WETH() (contracts/Token.sol#13) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#119) is not in mixedCase
Contract POTTER_PEPE (contracts/Token.sol#156-640) is not in CapWords
Struct POTTER_PEPE.valuesFromGetValues (contracts/Token.sol#196-204) is not in CapWords
Function POTTER_PEPE.EnableTrading() (contracts/Token.sol#354-357) is not in mixedCase
Function POTTER_PEPE.InternalSwap() (contracts/Token.sol#595-615) is not in mixedCase
Parameter POTTER_PEPE.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (contracts/Token.sol#631) is not in mixedCase
Parameter POTTER_PEPE.rescueAnyBEP20Tokens(address,address,uint256)._to (contracts/Token.sol#632) is not in mixedCase
Parameter POTTER_PEPE.rescueAnyBEP20Tokens(address,address,uint256)._amount (contracts/Token.sol#633) is not in mixedCase
Constant POTTER_PEPE._decimals (contracts/Token.sol#169) is not in UPPER_CASE_WITH_UNDERSCORES
Constant POTTER_PEPE._name (contracts/Token.sol#177) is not in UPPER_CASE_WITH_UNDERSCORES
Constant POTTER_PEPE._symbol (contracts/Token.sol#178) is not in UPPER_CASE_WITH_UNDERSCORES
Modifier POTTER_PEPE.LockSwap() (contracts/Token.sol#211-215) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#64)" inContext (contracts/Token.sol#58-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

POTTER_PEPE._getTValues(uint256,bool,address,address) (contracts/Token.sol#470-492) uses literals with too many digits:
        - s.tRfi = (tAmount * temp.rfi) / 100000 (contracts/Token.sol#488)
POTTER_PEPE._getTValues(uint256,bool,address,address) (contracts/Token.sol#470-492) uses literals with too many digits:
        - s.tMarketing = (tAmount * temp.marketing) / 100000 (contracts/Token.sol#489)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

POTTER_PEPE._tTotal (contracts/Token.sol#172) should be constant
POTTER_PEPE.marketingWallet (contracts/Token.sol#175) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

POTTER_PEPE.pair (contracts/Token.sol#217) should be immutable
POTTER_PEPE.swapRouter (contracts/Token.sol#218) 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/0xc8f5c7c53dccd124a17680f93a48918adf6e1ffad741b9d163c9ccf47acb49c4

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xb38185e488e4a9a7df56acd68680558b70674ea1d42b13f2da04f973f17c84f4

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xe2eac8912f430b7a06e3f51caec06e98226bb70e3b4095b5efc4aaf4927f5437

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xcdcd019c82d7d67aca40a99689beedf62d82aa195c3bd2de0779b4a40577282e

5- Buying when not excluded from fees (0-12.5% tax) (passed):
https://testnet.bscscan.com/tx/0x82f88655ee0d2e7c6cdd300b56de3cc796a653eb0977f025a979cf5d211a884d

6- Selling when not excluded from fees (0-12.5% tax) (passed):
https://testnet.bscscan.com/tx/0x23d632bee1643db077a2183f00165ab6d167821e818828591099cef83f3dd99a

7- Transferring when not excluded from fees (0-5% tax) (passed):
https://testnet.bscscan.com/tx/0x5d53ad743d70db5c01745c81b574d25222dda0e369ea7580b97fe64e5fb94a58

FUNCTIONAL TESTING

8- Internal swap (marketing bnb) (passed):
https://testnet.bscscan.com/address/0xdf552c6f432be69d4e6cce2792d6de0ffc0e76eb#internaltx

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: Suggestion

function: Enable Trading

Status: Not Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remains disabled, no one would be able to buy/sell/transfer tokens.

```
function Enable Trading () external only Owner {
    require (! trading Enabled, "Cannot re-enable trading");
    trading Enabled = 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.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the main Router, 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