auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PeiPei Thinking About Orange

DATED : 26 July 24'

MANUAL TESTING

Centralization – Enabling Trades

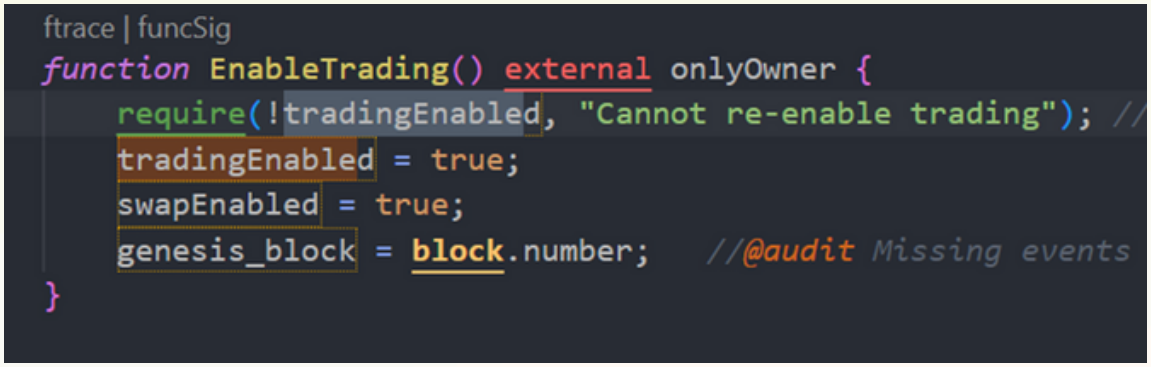
Severity: High

Function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.



Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

AUDIT SUMMARY

Project name – PeiPei Thinking About Orange

Date: 26 July, 2024

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **Passed with High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	1	2	1
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/0x4c44cbddf35d85023c9e33aa435e32ba069527c6

Token Information

Token Address:
0x612837768EF6516338DA9F41909Ac1cBA623C912

Name: PeiPei Thinking About Orange

Symbol: PETHAO

Decimals: 9

Network: Bsc Scan

Token Type: BEP-20

Owner: 0x15cC222BB37A2A4A9af86033fc90f8bdF7019A86

Deployer: 0x15cC222BB37A2A4A9af86033fc90f8bdF7019A86

Token Supply: 1,000,000,000

Checksum: 509f98d5565a2ff955f24ddff42fd7b251bf30fc

Testnet:
https://testnet.bscscan.com/address/0x4c44cbddf35d85023c9e33aa435e32ba069527c6

TOKEN OVERVIEW

Buy Fee: 0-0%

Sell Fee: 0-0%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

AUDIT METHODOLOGY

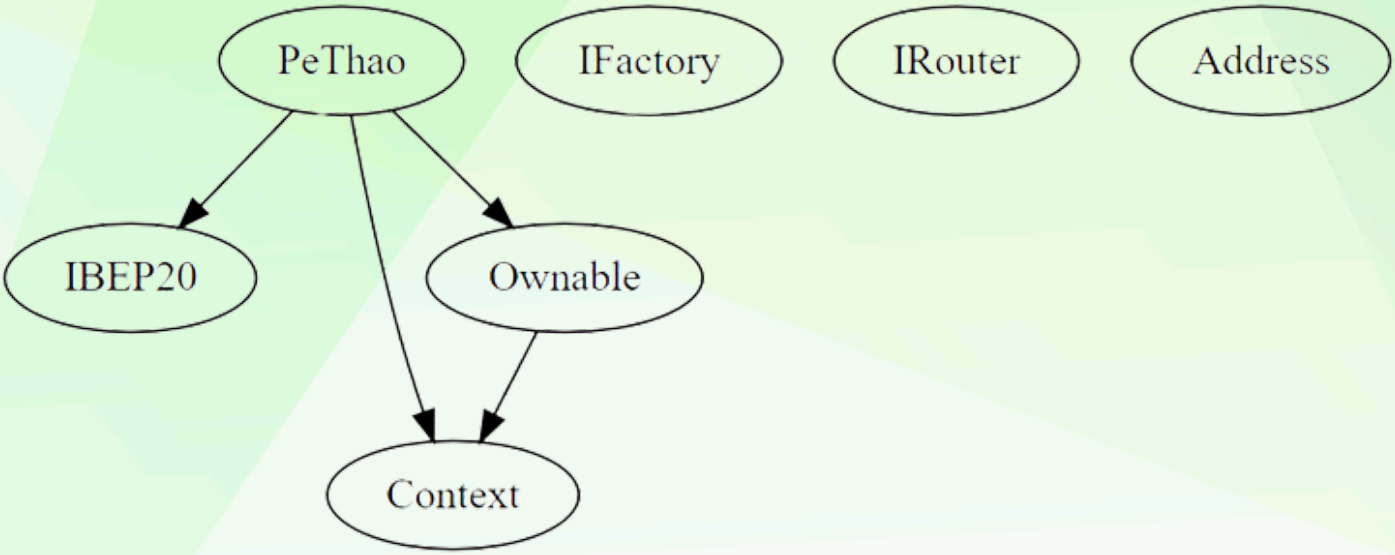
The auditing process will follow a routine as special considerations by Auditace:

- 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

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

INHERITANCE TREE



POINTS TO NOTE

- The owner can exclude/includeFrom/InReward.
- The owner can update deadline.
- The owner can Enable trading.
- The owner can exclude/includeFrom/InFee.
- The owner can Set Taxes.
- The owner can set Sell Taxes not more than 10%.
- The owner can rescue BNB.
- The owner can bulkExcludeFee.
- The owner can update marketing wallet.
- The owner can update dev wallet.
- The owner can update marketing wallet.
- The owner can update Swap tokens at amount.
- The owner can update marketing wallet.
- The owner can rescue any BEP20 Tokens.

STATIC ANALYSIS

```
Function IRouter.WETH() (PeThao.sol#81) is not in mixedCase
Struct PeThao.valuesFromGetValues (PeThao.sol#179-193) is not in CapWords
Function PeThao.EnableTrading() (PeThao.sol#313-318) is not in mixedCase
Parameter PeThao.updatedeadline(uint256)._deadline (PeThao.sol#320) is not in mixedCase
Parameter PeThao.setTaxes(uint256,uint256,uint256,uint256,uint256)._rfi (PeThao.sol#717) is not in mixedCase
Parameter PeThao.setTaxes(uint256,uint256,uint256,uint256,uint256)._marketing (PeThao.sol#718) is not in mixedCase
Parameter PeThao.setTaxes(uint256,uint256,uint256,uint256,uint256)._ops (PeThao.sol#719) is not in mixedCase
Parameter PeThao.setTaxes(uint256,uint256,uint256,uint256,uint256)._liquidity (PeThao.sol#720) is not in mixedCase
Parameter PeThao.setTaxes(uint256,uint256,uint256,uint256,uint256)._dev (PeThao.sol#721) is not in mixedCase
Parameter PeThao.setSellTaxes(uint256,uint256,uint256,uint256,uint256)._rfi (PeThao.sol#729) is not in mixedCase
Parameter PeThao.setSellTaxes(uint256,uint256,uint256,uint256,uint256)._marketing (PeThao.sol#730) is not in mixedCase
Parameter PeThao.setSellTaxes(uint256,uint256,uint256,uint256,uint256)._ops (PeThao.sol#731) is not in mixedCase
Parameter PeThao.setSellTaxes(uint256,uint256,uint256,uint256,uint256)._liquidity (PeThao.sol#732) is not in mixedCase
Parameter PeThao.setSellTaxes(uint256,uint256,uint256,uint256,uint256)._dev (PeThao.sol#733) is not in mixedCase
Parameter PeThao.updateSwapEnabled(bool)._enabled (PeThao.sol#751) is not in mixedCase
Parameter PeThao.rescueAnyBEP20Tokens(address,address,uint256)._tokenAddr (PeThao.sol#762) is not in mixedCase
Parameter PeThao.rescueAnyBEP20Tokens(address,address,uint256)._to (PeThao.sol#762) is not in mixedCase
Parameter PeThao.rescueAnyBEP20Tokens(address,address,uint256)._amount (PeThao.sol#762) is not in mixedCase
Constant PeThao._decimals (PeThao.sol#138) is not in UPPER_CASE_WITH_UNDERSCORES
Variable PeThao.genesis_block (PeThao.sol#146) is not in mixedCase
Constant PeThao._name (PeThao.sol#154) is not in UPPER_CASE_WITH_UNDERSCORES
Constant PeThao._symbol (PeThao.sol#155) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
PeThao.updatedeadline(uint256) (PeThao.sol#320-324) should emit an event for:
        - deadline = _deadline (PeThao.sol#323)
PeThao.updateSwapTokensAtAmount(uint256) (PeThao.sol#746-749) should emit an event for:
        - swapTokensAtAmount = amount * 10 ** _decimals (PeThao.sol#748)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
PeThao.constructor(address)._pair (PeThao.sol#206) lacks a zero-check on :
                - pair = _pair (PeThao.sol#209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in PeThao._transfer(address,address,uint256) (PeThao.sol#547-582):
        External calls:
        - swapAndLiquify(swapTokensAtAmount,sellTaxes) (PeThao.sol#573)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (PeThao.sol#672-
679)
                - (success,None) = recipient.call{value: amount}() (PeThao.sol#112)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (PeThao
.sol#691-697)
                - address(marketingWallet).sendValue(marketingAmt) (PeThao.sol#653)
                - address(devWallet).sendValue(devAmt) (PeThao.sol#658)
                - address(opsWallet).sendValue(opsAmt) (PeThao.sol#663)
        - swapAndLiquify(swapTokensAtAmount,taxes) (PeThao.sol#574)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,deadWallet,block.timestamp) (PeThao.sol#672-
679)
                - (success,None) = recipient.call{value: amount}() (PeThao.sol#112)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (PeThao
.sol#691-697)
                - address(marketingWallet).sendValue(marketingAmt) (PeThao.sol#653)
                - address(devWallet).sendValue(devAmt) (PeThao.sol#658)
```

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- EnableTrading (passed):

https://testnet.bscscan.com/tx/0x023b3c4275c8ece9fce34bdf3914a640b7412303df22c9843962a18d9554c04d

2- Update Marketing Wallet (passed):

https://testnet.bscscan.com/tx/0x6f4a3f5275bdac11b5e74621b4fe851c9502aea7a53a435bd3d012bf91f7d0e2

3- Update Ops Wallet (passed):

https://testnet.bscscan.com/tx/0xd459313676bb5eddc7cd40dcf0a5f3cea6ef9d3dbfc77b20255f38c47a1849e3

4- Update Dev Wallet (passed):

https://testnet.bscscan.com/tx/0xb7fb0871bd3f2bb7e05a643e2f9e61e5b213f5737daf2c5d898d84c832495b41

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	1
Low-Risk	2
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Enabling Trades

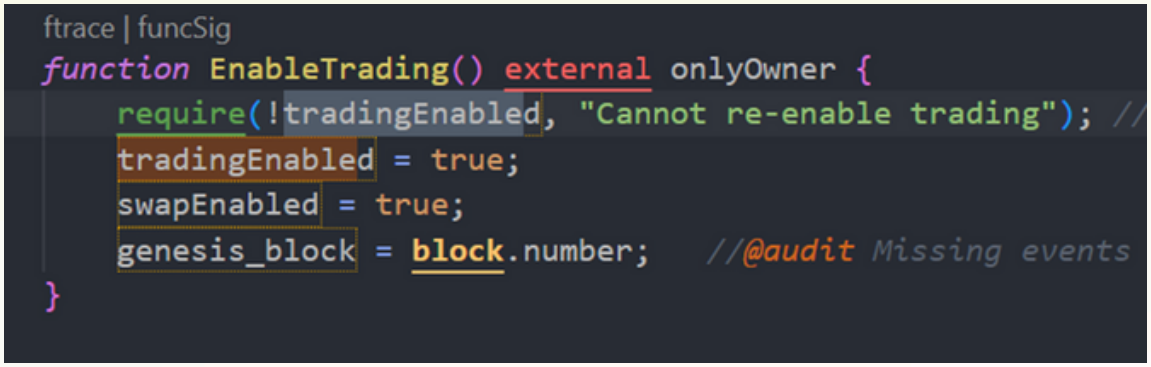
Severity: High

Function: EnableTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.



Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Missing Require Check.

Severity: Medium

Function: Update Marketing Wallet /Update_Dev Wallet

Status: Open

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
ftrace | funcSig
function updateMarketingWallet(address newWallet1) external onlyOwner {
    require(newWallet1 != address(0),"Fee Address cannot be zero address");
    marketingWallet = newWallet1;
}

ftrace | funcSig
function updateDevWallet(address newWallet1) external onlyOwner {
    require(newWallet1 != address(0),"Fee Address cannot be zero address");
    devWallet = newWallet1;
}
```

```
ftrace | funcSig
function updateOpsWallet(address newWallet1) external onlyOwner {
    require(newWallet1 != address(0),"Fee Address cannot be zero address");
    opsWallet = newWallet1;
}
```

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Subject: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function updateOpsWallet(address newWallet↑) external onlyOwner {
    require(newWallet↑ != address(0),"Fee Address cannot be zero address");
    opsWallet = newWallet↑;
}

ftrace | funcSig
function updateSwapTokensAtAmount(uint256 amount↑) external onlyOwner {
    require(amount↑ <= 1e7, "Cannot set swap threshold amount higher than 1% of tokens");
    swapTokensAtAmount = amount↑ * 10**_decimals;
}

ftrace | funcSig
function updateSwapEnabled(bool _enabled↑) external onlyOwner {
    swapEnabled = _enabled↑;
}
```

Suggestion:

Emit an event for critical changes.

MANUAL TESTING

Centralization – Local variable Shadowing

Severity: Low

Subject: Variable Shadowing

Status: Open

Overview:

```
ftrace | funcSig
function _approve(
    address owner↑,
    address spender↑,
    uint256 amount↑
) private {
    require(owner != address(0), "BEP20: approve from the zero address"); //@audit Local variable shadow
    require(spender↑ != address(0), "BEP20: approve to the zero address");
    _allowances[owner][spender↑] = amount↑;
    emit Approval(owner, spender↑, amount↑);
}
```

Suggestion:

Rename the local variables that shadow another component.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma.

Status: Open

Overview:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.26;
```

Suggestion:

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

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