auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Peach Inu

DATED : 9 APRIL 23'

AUDIT SUMMARY

Project name – Peach Inu

Date: 9 April, 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	2	0	0	0

USED TOOLS

Tools:

1- Manual Review:
a line by line code review has been performed by audit ace team.

2- BSC Testnet 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 Link: Contract has been tested on binance smart chain testnet which can be found in below link:
https://testnet.bscscan.com/token/0xeA9b9184e28192d6bFB0443cFdB6E9b18257f58D

Token Information

Token Name : Peach Inu

Token Symbol: PEACH

Decimals: 9

Token Supply: 5,000,000,000,000,000

Token Address:
0x2A374d02e244aAa175b38bA1Ba9ee443d20E7E41

Checksum:
9814facf1fbfd4e831bad29b666e83d7a9f466e1

Owner:
0xdAAF39E49e294c04727CA1C89c7A16b203Cf6Cd1
(at time of audit)

TOKEN OVERVIEW

Fees:

Buy Fees: 10%

Sell Fees: 10%

Transfer Fees: 0%

Fees Privilege: None

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: including and excluding form fee - changing distribution settings (min tokens to be eligible, cool down between claims etc)

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

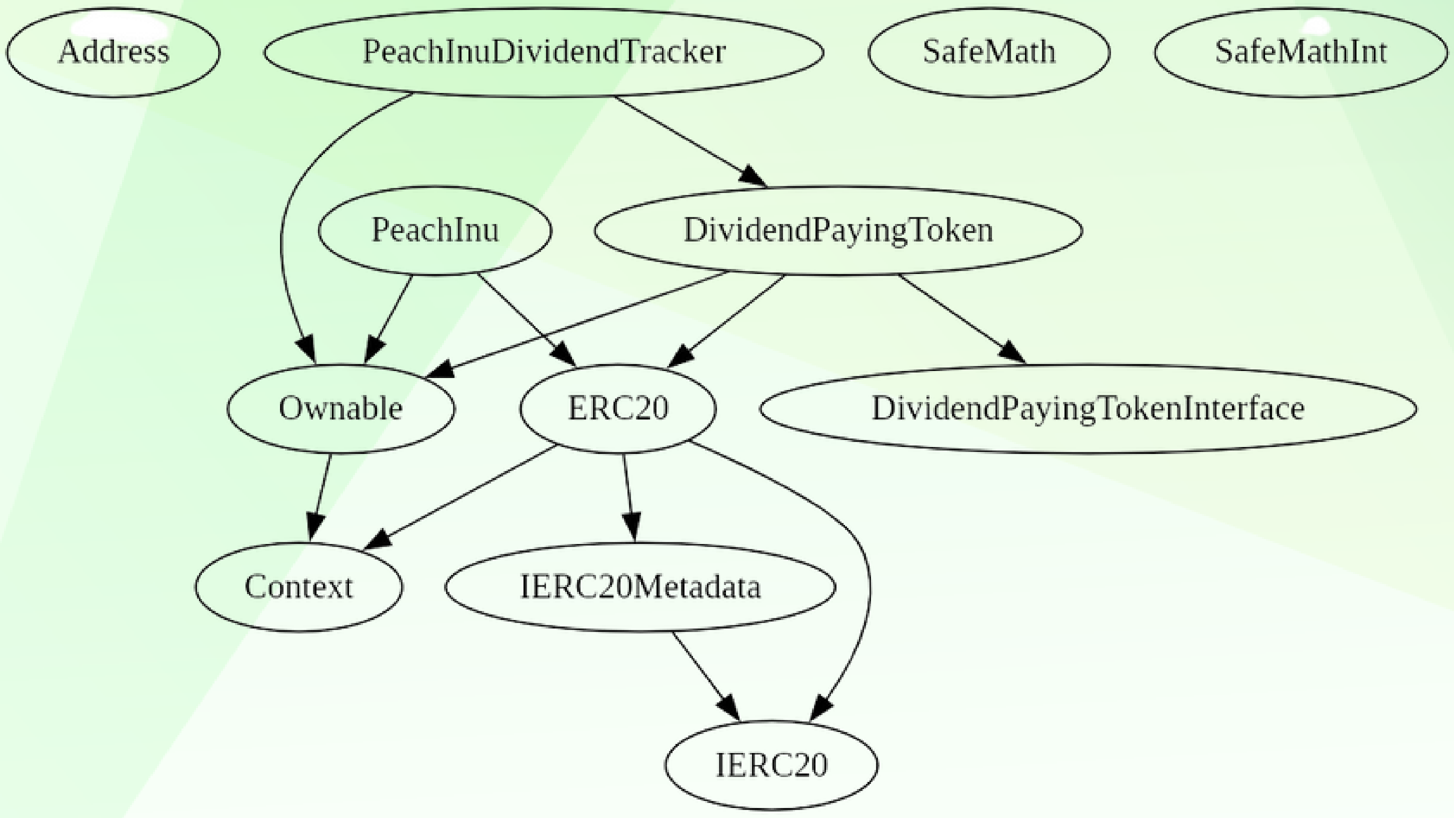
CLASSIFICATION OF RISK

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	2
🟧 Medium-Risk	0
🟨 Low-Risk	0
🔷 Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to modify buy/sell fees
- Owner is not able to set transfer fees (0% always)
- Owner is not able to set max buy/sell/transfer/hold 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 trading for investors**

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Address	Library			
└	sendValue	Internal 🔒	🛑	
PeachInu	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO ❗
└	updateDividendTracker	Public ❗	🛑	onlyOwner
└	processDividendTracker	External ❗	🛑	NO ❗
└	claim	External ❗	🛑	NO ❗
└	rescueBEP20Tokens	External ❗	🛑	onlyOwner
└	forceSend	External ❗	🛑	NO ❗
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setOpsWallet	External ❗	🛑	onlyOwner
└	setDevWallet	External ❗	🛑	onlyOwner
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	setSwapEnabled	External ❗	🛑	onlyOwner
└	enableTradingEnabled	External ❗	🛑	onlyOwner
└	setAntiBotBlocks	External ❗	🛑	onlyOwner
└	setMinBalanceForDividends	External ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	setGasForProcessing	External ❗	🛑	onlyOwner
└	setClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗		NO ❗
└	getTotalDividendsDistributed	External ❗		NO ❗
└	isExcludedFromFees	Public ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	getCurrentRewardToken	External ❗		NO ❗
└	dividendTokenBalanceOf	Public ❗		NO ❗
└	getAccountDividendsInfo	External ❗		NO ❗
└	getAccountDividendsInfoAtIndex	External ❗		NO ❗
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfDividendTokenHolders	External ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	swapAndLiquify	Private 🔐	🛑	
└	swapTokensForBNB	Private 🔐	🛑	

CONTRACT ASSESMENT

```
| └ | addLiquidity | Private 🔐 | 🛑 | |
||||||
| **PeachInuDividendTracker** | Implementation | Ownable, DividendPayingToken |||
| └ | <Constructor> | Public ❗️ | 🛑 | DividendPayingToken |
| └ | _transfer | Internal 🔒 | | |
| └ | setMinBalanceForDividends | External ❗️ | 🛑 | onlyOwner |
| └ | excludeFromDividends | External ❗️ | 🛑 | onlyOwner |
| └ | updateClaimWait | External ❗️ | 🛑 | onlyOwner |
| └ | getLastProcessedIndex | External ❗️ | |NO ❗️ |
| └ | getNumberOfTokenHolders | External ❗️ | |NO ❗️ |
| └ | getCurrentRewardToken | External ❗️ | |NO ❗️ |
| └ | getAccount | Public ❗️ | |NO ❗️ |
| └ | getAccountAtIndex | Public ❗️ | |NO ❗️ |
| └ | canAutoClaim | Private 🔐 | | |
| └ | setBalance | Public ❗️ | 🛑 | onlyOwner |
| └ | process | Public ❗️ | 🛑 |NO ❗️ |
| └ | processAccount | Public ❗️ | 🛑 | onlyOwner |
||||||
| **DividendPayingToken** | Implementation | ERC20, DividendPayingTokenInterface, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑 | ERC20 |
| └ | <Receive Ether> | External ❗️ | 💵 |NO ❗️ |
| └ | distributeDividends | Public ❗️ | 💵 |NO ❗️ |
| └ | _withdrawDividendOfUser | Internal 🔒 | 🛑 | |
| └ | setRewardToken | External ❗️ | 🛑 | onlyOwner |
| └ | swapBnbForCustomToken | Internal 🔒 | 🛑 | |
| └ | dividendOf | Public ❗️ | |NO ❗️ |
| └ | withdrawableDividendOf | Public ❗️ | |NO ❗️ |
| └ | withdrawnDividendOf | Public ❗️ | |NO ❗️ |
| └ | accumulativeDividendOf | Public ❗️ | |NO ❗️ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | _tokengeneration | Internal 🔒 | 🛑 | |
| └ | _burn | Internal 🔒 | 🛑 | |
| └ | _setBalance | Internal 🔒 | 🛑 | |
||||||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
| └ | <Constructor> | Public ❗️ | 🛑 |NO ❗️ |
| └ | name | Public ❗️ | |NO ❗️ |
| └ | symbol | Public ❗️ | |NO ❗️ |
| └ | decimals | Public ❗️ | |NO ❗️ |
| └ | totalSupply | Public ❗️ | |NO ❗️ |
| └ | balanceOf | Public ❗️ | |NO ❗️ |
| └ | transfer | Public ❗️ | 🛑 |NO ❗️ |
```

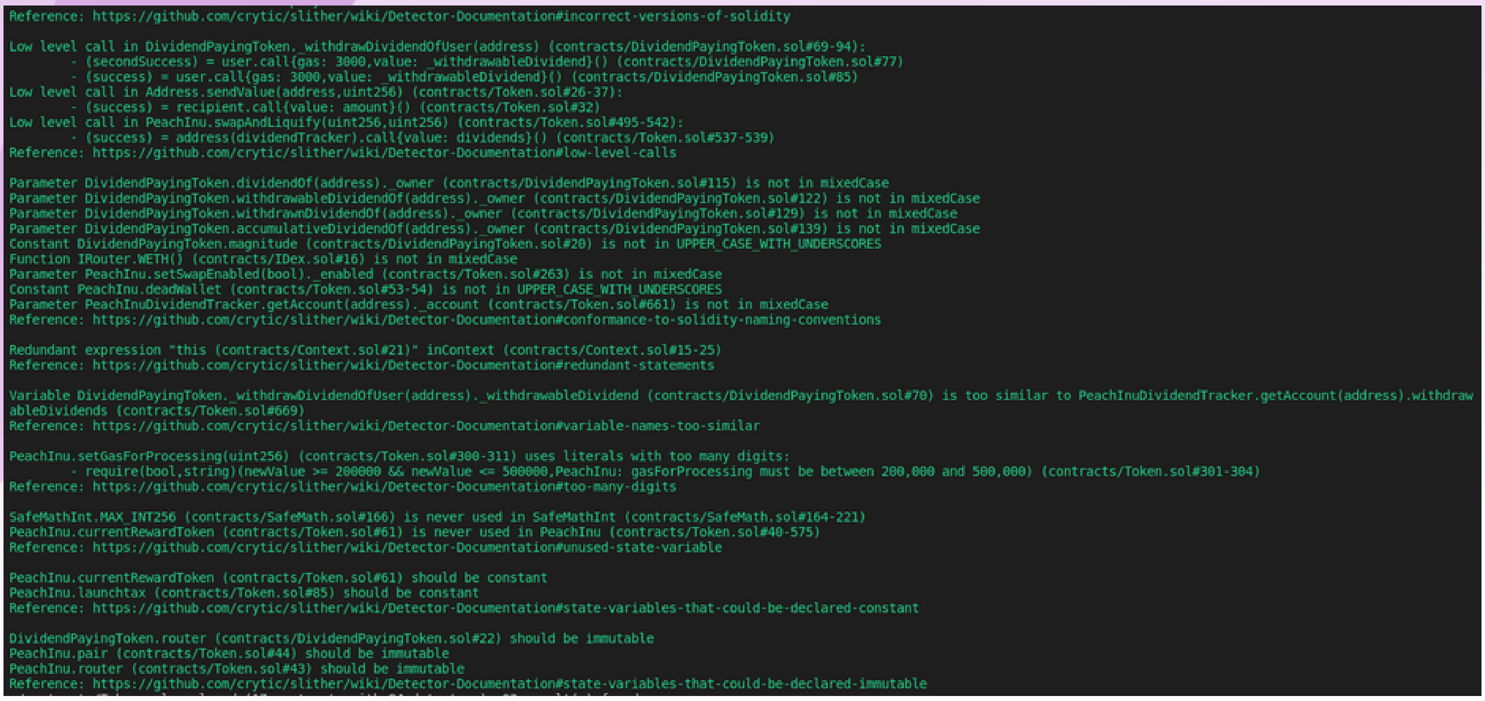
CONTRACT ASSESMENT

```
| └ | allowance | Public ❗ | |NO ❗ |
| └ | approve | Public ❗ | 🛑 |NO ❗ |
| └ | transferFrom | Public ❗ | 🛑 |NO ❗ |
| └ | increaseAllowance | Public ❗ | 🛑 |NO ❗ |
| └ | decreaseAllowance | Public ❗ | 🛑 |NO ❗ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | _tokengeneration | Internal 🔒 | 🛑 | |
| └ | _burn | Internal 🔒 | 🛑 | |
| └ | _approve | Internal 🔒 | 🛑 | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |
||||||
| **IERC20** | Interface | |||
| └ | totalSupply | External ❗ | |NO ❗ |
| └ | balanceOf | External ❗ | |NO ❗ |
| └ | transfer | External ❗ | 🛑 |NO ❗ |
| └ | allowance | External ❗ | |NO ❗ |
| └ | approve | External ❗ | 🛑 |NO ❗ |
| └ | transferFrom | External ❗ | 🛑 |NO ❗ |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ❗ | |NO ❗ |
| └ | symbol | External ❗ | |NO ❗ |
| └ | decimals | External ❗ | |NO ❗ |
||||||
| **Context** | Implementation | |||
| └ | _msgSender | Internal 🔒 | | |
| └ | _msgData | Internal 🔒 | | |
||||||
| **SafeMath** | Library | |||
| └ | add | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | mul | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | mod | Internal 🔒 | | |
| └ | mod | Internal 🔒 | | |
||||||
| **SafeMathInt** | Library | |||
| └ | mul | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | add | Internal 🔒 |   | |
| └ | abs | Internal 🔒 |   | |
| └ | toUint256Safe | Internal 🔒 |   | |
||||||
| **SafeMathUint** | Library |  |||
| └ | toInt256Safe | Internal 🔒 |   | |
||||||
| **DividendPayingTokenInterface** | Interface |  |||
| └ | dividendOf | External ❗️ |   |NO ❗️ |
| └ | distributeDividends | External ❗️ |  💵 |NO ❗️ |
| └ | withdrawableDividendOf | External ❗️ |   |NO ❗️ |
| └ | withdrawnDividendOf | External ❗️ |   |NO ❗️ |
| └ | accumulativeDividendOf | External ❗️ |   |NO ❗️ |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO ❗️ |
| └ | owner | Public ❗️ |   |NO ❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑  | onlyOwner |
||||||
| **IPair** | Interface |  |||
| └ | sync | External ❗️ | 🛑  |NO ❗️ |
||||||
| **IFactory** | Interface |  |||
| └ | createPair | External ❗️ | 🛑  |NO ❗️ |
| └ | getPair | External ❗️ |   |NO ❗️ |
||||||
| **IRouter** | Interface |  |||
| └ | factory | External ❗️ |   |NO ❗️ |
| └ | WETH | External ❗️ |   |NO ❗️ |
| └ | addLiquidityETH | External ❗️ |  💵 |NO ❗️ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗️ | 🛑  |NO ❗️ |
| └ | swapExactETHForTokens | External ❗️ |  💵 |NO ❗️ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗️ | 🛑  |NO ❗️ |
||||||
| **IterableMapping** | Library |  |||
| └ | get | Internal 🔒 |   | |
| └ | getIndexOfKey | Internal 🔒 |   | |
| └ | getKeyAtIndex | Internal 🔒 |   | |
| └ | size | Internal 🔒 |   | |
| └ | set | Internal 🔒 | 🛑  | |
| └ | remove | Internal 🔒 | 🛑  | |
```

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in DividendPayingToken._withdrawDividendOfUser(address) (contracts/DividendPayingToken.sol#69-94):
        - (secondSuccess) = user.call{gas: 3000,value: _withdrawableDividend}() (contracts/DividendPayingToken.sol#77)
        - (success) = user.call{gas: 3000,value: _withdrawableDividend}() (contracts/DividendPayingToken.sol#85)
Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#26-37):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#32)
Low level call in PeachInu.swapAndLiquify(uint256,uint256) (contracts/Token.sol#495-542):
        - (success) = address(dividendTracker).call{value: dividends}() (contracts/Token.sol#537-539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter DividendPayingToken.dividendOf(address)._owner (contracts/DividendPayingToken.sol#115) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (contracts/DividendPayingToken.sol#122) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (contracts/DividendPayingToken.sol#129) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (contracts/DividendPayingToken.sol#139) is not in mixedCase
Constant DividendPayingToken.magnitude (contracts/DividendPayingToken.sol#20) is not in UPPER_CASE_WITH_UNDERSCORES
Function IRouter.WETH() (contracts/IDex.sol#16) is not in mixedCase
Parameter PeachInu.setSwapEnabled(bool)._enabled (contracts/Token.sol#263) is not in mixedCase
Constant PeachInu.deadWallet (contracts/Token.sol#53-54) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter PeachInuDividendTracker.getAccount(address)._account (contracts/Token.sol#661) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Context.sol#21)" inContext (contracts/Context.sol#15-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (contracts/DividendPayingToken.sol#70) is too similar to PeachInuDividendTracker.getAccount(address).withdraw
ableDividends (contracts/Token.sol#669)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

PeachInu.setGasForProcessing(uint256) (contracts/Token.sol#300-311) uses literals with too many digits:
        - require(bool,string)(newValue >= 200000 && newValue <= 500000,PeachInu: gasForProcessing must be between 200,000 and 500,000) (contracts/Token.sol#301-304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

SafeMathInt.MAX_INT256 (contracts/SafeMath.sol#166) is never used in SafeMathInt (contracts/SafeMath.sol#164-221)
PeachInu.currentRewardToken (contracts/Token.sol#61) is never used in PeachInu (contracts/Token.sol#40-575)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

PeachInu.currentRewardToken (contracts/Token.sol#61) should be constant
PeachInu.launchtax (contracts/Token.sol#85) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DividendPayingToken.router (contracts/DividendPayingToken.sol#22) should be immutable
PeachInu.pair (contracts/Token.sol#44) should be immutable
PeachInu.router (contracts/Token.sol#43) 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 issues found

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x8422e8b8eb95814216122abe96f425afc6dc4aaa713173a23a315bf027de4a44

2- Buying when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x35201b1ab2dcff67d045961d20ff5053173997479ab181c945af4894a296432b

3- Selling when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x17a139f374b6112265d9618758003847f3458cfbbcf3ed4161e9a8a8e1f2c5b2

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x56b1a69e379d4efd19a6808d58bb727a22cf1b32655fc2671db0d9be8054c644

5- Buying when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0x7dcb0aa54868330bfc992fc66b1279ee918d45b5fb836e002d9b85c3ec566c76

6- Selling when not excluded from fees (10% tax) (passed):
https://testnet.bscscan.com/tx/0x1c52ce80d71f591e4226d275a345c1df7202d66f858c0d6bcc1d881a463956f3

7- Transferring when not excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x4e0081af2b95427c6f30e0cf33ff03dc02c92e027120e2def64f422263e23411

FUNCTIONAL TESTING

8- Internal swap **(passed):**
fee wallets received BNB
https://testnet.bscscan.com/tx/0x1c52ce80d71f591e4226d275a345c1df7202d66f858c0d6bcc1d881a463956f3

9- Distribution of rewards **(passed):**
BUSD tokens are distributed between holders, this can be seen in this transaction
https://testnet.bscscan.com/tx/0x1c52ce80d71f591e4226d275a345c1df7202d66f858c0d6bcc1d881a463956f3

MANUAL TESTING

Centralization - Owner must enable trading

Severity: High

Function: enableTradingEnabled

Lines: 242

Status: Resolved

Overview:

The owner must activate trading for investors to buy, sell, or transfer tokens. If trading remains disabled, token holders will be unable to trade their tokens.

```
function enableTradingEnabled() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    startTradingBlock = block.number;
}
```

Recommendation:

Incorporate a safety mechanism that allows investors to activate trading if a specified duration has elapsed since the conclusion of the presale or consider alternative ways such as allowing trades ater investors claimed their presale tokens.

Allevation:

Contract is owned by Safu Dev, hence enabling trade is guaranteed

MANUAL TESTING

Logical - Setting internal swap threshold to 0 can disable sells

Severity: High

Function: setSwapThreshold

Lines: 231

Status: Resolved

If the **swaptokensAtAmount** is set to 0, sell transactions will fail at the _transfer function. This occurs because the checks for performing a swapAndLiquify will still pass even if the swapThreshold is set to 0 and the contract has 0 tokens. Consequently, the transaction will fail while attempting to swap 0 tokens (i.e., **swaptokensAtAmount**) to BNB. Additionally, setting the swapThreshold to an excessively large number leads to a high slippage percentage during sell transactions.

```
function setSwapTokensAtAmount(uint256 amount) external onlyOwner {
  require(
     amount < 5e13,
     "Swap Threshold should be less than 1% of total supply");
  swapTokensAtAmount = amount * 10 ** 9;
}
if (
  canSwap &&
  !swapping &&
  swapEnabled &&
  !automatedMarketMakerPairs[from] &&
  !_isExcludedFromFees[from] &&
  !_isExcludedFromFees[to]
) {
  swapping = true;
  if (swapTax > 0) {
    swapAndLiquify(swapTokensAtAmount, swapTax);
  }
  swapping = false;
}
```

MANUAL TESTING

Recommendation:

Ensure that the swapThreshold is set to a value greater than a reasonable minimum and less than a reasonable maximum. This will help prevent issues related to disabled sell transactions or high slippage percentages during trades.

Allevation:

Contract is owned by Safu Dev, swap threshold will remain in a logical range

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