AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PURR

DATED : 10 august 23'

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.

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

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 provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner.

AUDIT SUMMARY

Project name –PURR

Date: 10 august, 2023

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	0	1	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/0xfD57fE9b373AEA047d1b5D2FB5c0A2fE06a18c35

Token Information

Token Name : PURR

Token Symbol: PURR

Decimals: 18

Token Supply: 9,000,000,000

Token Address:
0xA3F45E557fc3096d29b6c5f993a0D6802CcA809B

Checksum:
ee0aa420516e96c45d2a307eb95f3beb4c8ed192

Owner:
0xB6411A7905B49a8A8471b9f89cdCb791b8465633
(at time of writing the audit)

Deployer:
0xB6411A7905B49a8A8471b9f89cdCb791b8465633

TOKEN OVERVIEW

Fees:

Buy Fees: 0-3%

Sell Fees: 0-3%

Transfer Fees: 0%

Fees Privilege: owner

Ownership: not owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: no

Blacklist: No

Other Privileges: Initial distribution of the tokens

modifying fees

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

- [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

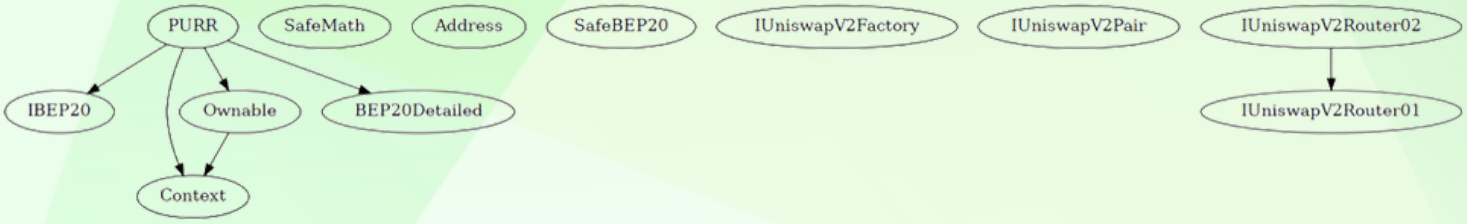
CLASSIFICATION OF RISK

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	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is able to update buy and sell fees (0-3%)**
- **Owner is not able to set fee on transfers**
- Owner is not able to blacklist an address
- Owner is not able to set maximum wallet and maximum buy/sell limits
- Owner is not able to mint new tokens
- Owner is not able to disable trades

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface				
└	totalSupply	External ❗		NO ❗	
└	balanceOf	External ❗		NO ❗	
└	transfer	External ❗	🛑	NO ❗	
└	allowance	External ❗		NO ❗	
└	approve	External ❗	🛑	NO ❗	
└	transferFrom	External ❗	🛑	NO ❗	
SafeMath	Library				
└	tryAdd	Internal 🔒			
└	trySub	Internal 🔒			
└	tryMul	Internal 🔒			
└	tryDiv	Internal 🔒			
└	tryMod	Internal 🔒			
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
└	sub	Internal 🔒			

CONTRACT ASSESMENT

└	div	Internal 🔒			
└	mod	Internal 🔒			
Context	Implementation				
└	<Constructor>	Public ❗	🛑	NO ❗	
└	_msgSender	Internal 🔒			
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🛑	NO ❗	
└	owner	Public ❗		NO ❗	
└	renounceOwnership	Public ❗	🛑	onlyOwner	
└	transferOwnership	Public ❗	🛑	onlyOwner	
BEP20Detailed	Implementation				
└	<Constructor>	Public ❗	🛑	NO ❗	
└	name	Public ❗		NO ❗	
└	symbol	Public ❗		NO ❗	
└	decimals	Public ❗		NO ❗	
Address	Library				
└	isContract	Internal 🔒			
SafeBEP20	Library				
└	safeTransfer	Internal 🔒	🛑		
└	safeTransferFrom	Internal 🔒	🛑		
└	safeApprove	Internal 🔒	🛑		
└	callOptionalReturn	Private 🔐	🛑		
IUniswapV2Factory	Interface				
└	feeTo	External ❗		NO ❗	
└	feeToSetter	External ❗		NO ❗	
└	getPair	External ❗		NO ❗	
└	allPairs	External ❗		NO ❗	
└	allPairsLength	External ❗		NO ❗	

CONTRACT ASSESMENT

└	createPair	External ❗	🛑	NO ❗
└	setFeeTo	External ❗	🛑	NO ❗
└	setFeeToSetter	External ❗	🛑	NO ❗
IUniswapV2Pair	Interface			
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗	NO ❗
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	allowance	External ❗	NO ❗
└	approve	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
└	DOMAIN_SEPARATOR	External ❗	NO ❗
└	PERMIT_TYPEHASH	External ❗		NO ❗
└	nonces	External ❗		NO ❗
└	permit	External ❗	🛑	NO ❗
└	MINIMUM_LIQUIDITY	External ❗		NO ❗
└	factory	External ❗		NO ❗
└	token0	External ❗		NO ❗
└	token1	External ❗		NO ❗
└	getReserves	External ❗	NO ❗
└	price0CumulativeLast	External ❗		NO ❗
└	price1CumulativeLast	External ❗		NO ❗
└	kLast	External ❗	NO ❗
└	mint	External ❗	🛑	NO ❗
└	burn	External ❗	🛑	NO ❗
└	swap	External ❗	🛑	NO ❗
└	skim	External ❗	🛑	NO ❗
└	sync	External ❗	🛑	NO ❗

CONTRACT ASSESMENT

└	initialize	External ❗	🛑	NO ❗
IUniswapV2Router01	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	removeLiquidity	External ❗	🛑	NO ❗
└	removeLiquidityETH	External ❗	🛑	NO ❗
└	removeLiquidityWithPermit	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗
└	swapTokensForExactTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	swapTokensForExactETH	External ❗	🛑	NO ❗
└	swapExactTokensForETH	External ❗	🛑	NO ❗
└	swapETHForExactTokens	External ❗	💵	NO ❗
└	quote	External ❗		NO ❗
└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑
NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗
🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗
🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵
NO ❗ |

CONTRACT ASSESMENT

| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🛑 | NO ❗ |

||||||

| **PURR** | Implementation | Context, Ownable, IBEP20, BEP20Detailed |||

| └ | <Constructor> | Public ❗ | 🛑 | BEP20Detailed |

| └ | totalSupply | Public ❗ | | NO ❗ |

| └ | balanceOf | Public ❗ | | NO ❗ |

| └ | transfer | Public ❗ | 🛑 | NO ❗ |

| └ | allowance | Public ❗ | | NO ❗ |

| └ | approve | Public ❗ | 🛑 | NO ❗ |

| └ | transferFrom | Public ❗ | 🛑 | NO ❗ |

| └ | increaseAllowance | Public ❗ | 🛑 | NO ❗ |

| └ | decreaseAllowance | Public ❗ | 🛑 | NO ❗ |

| └ | _approve | Internal 🔒 | 🛑 | |

| └ | enableTrading | External ❗ | 🛑 | onlyOwner |

| └ | isContract | Internal 🔒 | | |

| └ | setBuyNFTStakingFeePercent | External ❗ | 🛑 | onlyOwner |

| └ | setSellNFTStakingFeePercent | External ❗ | 🛑 | onlyOwner |

| └ | setNFTStakingAddress | External ❗ | 🛑 | onlyOwner |

| └ | setSwapAndLiquifyEnabled | Public ❗ | 🛑 | onlyOwner |

| └ | changeNumTokensSellToFee | External ❗ | 🛑 | onlyOwner |

| └ | clearETH | External ❗ | 🛑 | onlyOwner |

| └ | clearERC20 | External ❗ | 🛑 | onlyOwner |

| └ | excludeFromFee | Public ❗ | 🛑 | onlyOwner |

| └ | includeInFee | Public ❗ | 🛑 | onlyOwner |

| └ | isExcludedFromFee | Public ❗ | | NO ❗ |

| └ | <Receive Ether> | External ❗ | 💸 | NO ❗ |

| └ | _transfer | Internal 🔒 | 🛑 | |

| └ | swapAndLiquify | Private 🔐 | 🛑 | lockTheSwap |

| └ | swapTokensForEth | Private 🔐 | 🛑 | |

CONTRACT ASSESMENT

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in PURR._transfer(address,address,uint256) (contracts/Token.sol#839-904):
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/Token.sol#862)
                - address(stakingPoolAddress).transfer(address(this).balance) (contracts/Token.sol#910)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (contracts/Token.sol#888-891)
        - _balances[recipient] = _balances[recipient].add(TotalSent) (contracts/Token.sol#892)
        - _balances[address(this)] = _balances[address(this)].add(taxAmount) (contracts/Token.sol#893)
        - _balances[sender] = _balances[sender].sub(amount,BEP20: transfer amount exceeds balance) (contracts/Token.sol#897-900)
        - _balances[recipient] = _balances[recipient].add(amount) (contracts/Token.sol#901)
        - stakingPoolFee = buyNFTStakingFee (contracts/Token.sol#882)
        - stakingPoolFee = sellNFTStakingFee (contracts/Token.sol#884)
        Event emitted after the call(s):
        - Transfer(sender,recipient,TotalSent) (contracts/Token.sol#894)
        - Transfer(sender,address(this),taxAmount) (contracts/Token.sol#895)
        - Transfer(sender,recipient,amount) (contracts/Token.sol#902)
Reentrancy in PURR.swapAndLiquify(uint256) (contracts/Token.sol#906-913):
        External calls:
        - address(stakingPoolAddress).transfer(address(this).balance) (contracts/Token.sol#910)
        Event emitted after the call(s):
        - SwapAndLiquify(contractTokenBalance,address(this).balance) (contracts/Token.sol#912)
Reentrancy in PURR.transferFrom(address,address,uint256) (contracts/Token.sol#702-717):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#707)
                - address(stakingPoolAddress).transfer(address(this).balance) (contracts/Token.sol#910)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (contracts/Token.sol#708-715)
                - _allowances[towner][spender] = amount (contracts/Token.sol#749)
        Event emitted after the call(s):
        - Approval(towner,spender,amount) (contracts/Token.sol#750)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (contracts/Token.sol#708-715)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#420) is too similar to IUniswapV2Router01.ad
dLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#421)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
PURR._owner (contracts/Token.sol#645) should be immutable
PURR._totalSupply (contracts/Token.sol#620) 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/0xb478c0e85888fa612d0a99002555c61d9588b361c5a6daca4a8781144b796cb6

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

https://testnet.bscscan.com/tx/0x55246f9e06a18e530592a9d58d966865978c50bbee811139ee09fb9136fcec08

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

https://testnet.bscscan.com/tx/0x02bb77a44684019bb7f783f64352066286a1b35a80040fea7de0d7b60c0857b5

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

https://testnet.bscscan.com/tx/0x6e5451df2af5d9b57b82621faf2313cdd88f96505c3a655f62d8b98e33b3e1df

5- Buying when not excluded from fees (0-3% tax) (passed):

https://testnet.bscscan.com/tx/0x078fea19b2e505d16d64635097140bade3ce10e350e8dd9e3aec70b13d1311b5

6- Selling when not excluded from fees (0-3% tax) (passed):

https://testnet.bscscan.com/tx/0x46d12fab6114ff344fcf71ed1938c532a414c577bc30267cd7ae6ea275c731f1

FUNCTIONAL TESTING

7- Transferring (0% tax) (passed):

https://testnet.bscscan.com/tx/0xc9ca8bf986623de2e734876b6d9e180c2e7a604835554f74f5c52e6a4a4f731a

8- Internal swap(ETH sent to marketing wallet) (passed):

https://testnet.bscscan.com/tx/0x0b0c7d49ce354322905b9f3e25a7c817078b1f8cd8f5fd29286816aef6256e76

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.

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

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 provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner.

MANUAL TESTING

Logical – Updating Staking Wallet Address

Severity: Low
function: setNFTStakingAddress
Status: Not Resolved
Overview:
The function `setNFTStakingAddress` allows the owner to set a new staking wallet address. However, setting the `NFTStaking` wallet to a contract that rejects receiving Ether can cause the internal swaps (like `swap and liquify`) to revert. Such a revert consequently causes the sell or transfer transaction to also revert. It is essential to note that, despite the checks in place, the `stakingPoolAddress` can be set to a contract address. This is possible since contract addresses can be predictable, and the current logic does not thoroughly ensure that this cannot occur.

```
function setNFTStakingAddress(address payable wallet) external onlyOwner {
    require(
        wallet != stakingPoolAddress,
        "NFTStaking address is already that address"
    );
    require(
        wallet != address(0),
        "NFTStaking address cannot be dead address"
    );
    require(!isContract(wallet), "NFTStaking address cannot be contract");
    stakingPoolAddress = wallet;
}
```

Suggestion

To circumvent the potential revert issue due to transferring Ether to a non-receptive contract, it's recommended to utilize a low-level call for Ether transfers. By using this approach, the return value can be ignored, ensuring the `swapAndLiquify` function doesn't revert due to the Ether transfer.

Proposed code modification:

```
function swapAndLiquify(uint256 contractTokenBalance) private lockTheSwap {
    swapTokensForEth(contractTokenBalance);
    (bool success,) = stakingPoolAddress.call{value : address(this).balance}
("");
    emit SwapAndLiquify(contractTokenBalance, address(this).balance);
}
```

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