auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Spare Inu

DATED : 3 May 23'

AUDIT SUMMARY

Project name – Spare Inu

Date: 3 May, 2023

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

Audit Status: Failed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	2	2	2	0	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/0x7Be0e001a20b5D863B95e14F6CBBFdB240FCB06c

Token Information

Token Name : Spare Inu

Token Symbol: Spare Inu

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x1eb0Df45ea658c2d532D6F41eCEE42a45E1CeF2a

Checksum:
36552580df187046a7c9f54d1f604aee2936f4e8

Owner:
0x86339114E164aC1fCf114a4B9A2D17Aea4f14076
(at time of writing the audit)

Deployer:
0x86339114E164aC1fCf114a4B9A2D17Aea4f14076

TOKEN OVERVIEW

Fees:

Buy Fees: up to 100%

Sell Fees: up to 100%

Transfer Fees: up to 100%

Fees Privilege: Owner

Ownership: Owned

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: Yes

Blacklist: No

Other Privileges: updating fee - updating max sell

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

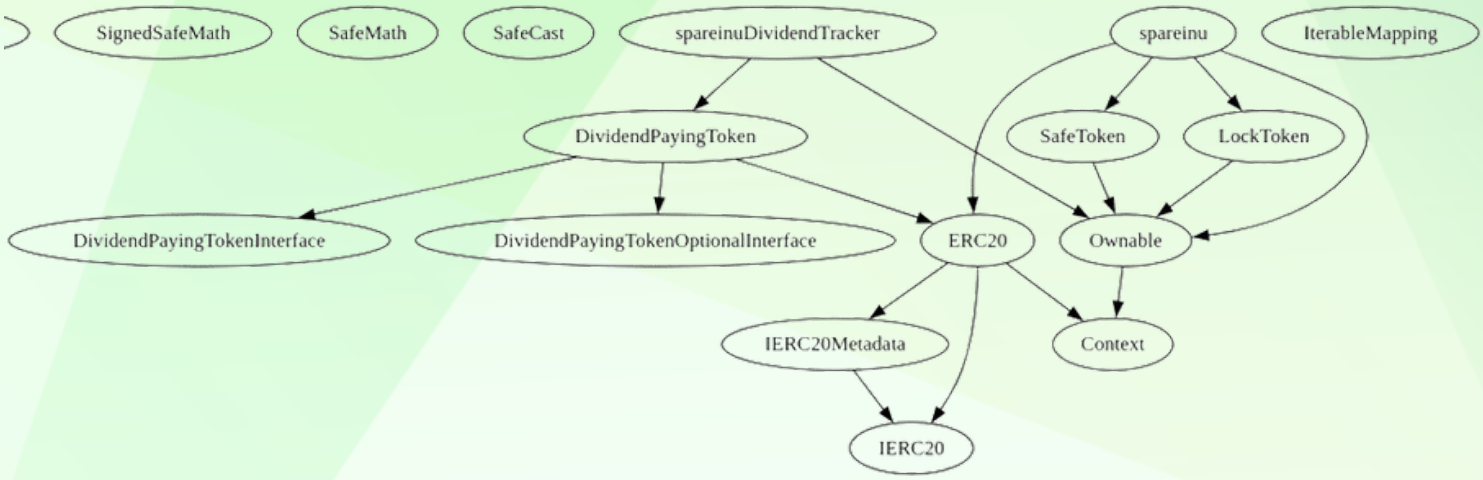
CLASSIFICATION OF RISK

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

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to set buy/sell/transfer fees up to 100%
- Owner is able to set max sell amounts to 0, disabling sells
- Owner is not able to set a max buy/transfer/wallet amount
- Owner is able to blacklist an arbitrary wallet
- Owner is able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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 ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗
└	getPair	External ❗		NO ❗
└	allPairs	External ❗		NO ❗
└	allPairsLength	External ❗		NO ❗
└	createPair	External ❗	🛑	NO ❗
└	setFeeTo	External ❗	🛑	NO ❗
└	setFeeToSetter	External ❗	🛑	NO ❗
SignedSafeMath	Library			

CONTRACT ASSESMENT

└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
SafeCast	Library			
└	toUint224	Internal 🔒		
└	toUint128	Internal 🔒		
└	toUint96	Internal 🔒		
└	toUint64	Internal 🔒		
└	toUint32	Internal 🔒		
└	toUint16	Internal 🔒		
└	toUint8	Internal 🔒		
└	toUint256	Internal 🔒		
└	toInt128	Internal 🔒		
└	toInt64	Internal 🔒		
└	toInt32	Internal 🔒		
└	toInt16	Internal 🔒		
└	toInt8	Internal 🔒		
└	toInt256	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗

CONTRACT ASSESMENT

└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
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 ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_setOwner	Private 🔐	🛑	
IterableMapping	Library			
└	get	Public ❗		NO ❗
└	getIndexOfKey	Public ❗		NO ❗
└	getKeyAtIndex	Public ❗		NO ❗
└	size	Public ❗		NO ❗
└	set	Public ❗	🛑	NO ❗

CONTRACT ASSESMENT

└	remove	Public ❗	🛑	NO ❗
DividendPayingTokenOptionalInterface	Interface			
└	withdrawableDividendOf	External ❗		NO ❗
└	withdrawnDividendOf	External ❗		NO ❗
└	accumulativeDividendOf	External ❗		NO ❗
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗		NO ❗
└	distributeDividends	External ❗	💵	NO ❗
└	withdrawDividend	External ❗	🛑	NO ❗
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO ❗
└	distributeDividends	Public ❗	💵	NO ❗
└	withdrawDividend	Public ❗	🛑	NO ❗
└	_withdrawDividendOfUser	Internal 🔒	🛑	
└	dividendOf	Public ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	withdrawnDividendOf	Public ❗		NO ❗
└	accumulativeDividendOf	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setBalance	Internal 🔒	🛑	
spareinuDividendTracker	Implementation	DividendPayingToken, Ownable		
└	<Constructor>	Public ❗	🛑	DividendPayingToken
└	_transfer	Internal 🔒		
└	withdrawDividend	Public ❗		NO ❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfTokenHolders	External ❗		NO ❗
└	getAccount	Public ❗		NO ❗
└	getAccountAtIndex	Public ❗		NO ❗
└	canAutoClaim	Private 🔐		
└	setBalance	External ❗	🛑	onlyOwner
└	process	Public ❗	🛑	NO ❗
└	processAccount	Public ❗	🛑	onlyOwner

CONTRACT ASSESMENT

SafeToken	Implementation	Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	setSafeManager	Public ❗	🛑	onlyOwner
└	withdraw	External ❗	🛑	NO ❗
└	withdrawBNB	External ❗	🛑	NO ❗
LockToken	Implementation	Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	openTrade	External ❗	🛑	onlyOwner
└	includeToWhiteList	External ❗	🛑	onlyOwner
spareinu	Implementation	ERC20, Ownable, SafeToken, LockToken		
└	setFee	Public ❗	🛑	onlyOwner
└	setExtraFeeOnSell	Public ❗	🛑	onlyOwner
└	setMaxSelltx	Public ❗	🛑	onlyOwner
└	blacklistAddress	External ❗	🛑	onlyOwner
└	setMarketingWallet	Public ❗	🛑	onlyOwner
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO ❗
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	setExcludeFromMaxTx	Public ❗	🛑	onlyOwner
└	setExcludeFromAll	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	setSWapTokensAtAmount	Public ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	updateGasForProcessing	Public ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗		NO ❗
└	getTotalDividendsDistributed	External ❗		NO ❗
└	isExcludedFromFees	Public ❗		NO ❗
└	isExcludedFromMaxTx	Public ❗		NO ❗
└	withdrawableDividendOf	Public ❗		NO ❗
└	dividendTokenBalanceOf	Public ❗		NO ❗
└	getAccountDividendsInfo	External ❗		NO ❗
└	getAccountDividendsInfoAtIndex	External ❗		NO ❗
└	processDividendTracker	External ❗	🛑	NO ❗
└	claim	External ❗	🛑	NO ❗
└	getLastProcessedIndex	External ❗		NO ❗
└	getNumberOfDividendTokenHolders	External ❗		NO ❗
└	excludeFromDividends	External ❗	🛑	onlyOwner

CONTRACT ASSESMENT

└	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
└	_transfer	Internal 🔒	🛑	open
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForBnb	Private 🔐	🛑		
└	swapAndSendBNBToMarketing	Private 🔐	🛑		
└	addLiquidity	Private 🔐	🛑		
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 ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑		
└	_mint	Internal 🔒	🛑		
└	_burn	Internal 🔒	🛑		
└	_approve	Internal 🔒	🛑		
└	_spendAllowance	Internal 🔒	🛑		
└	_beforeTokenTransfer	Internal 🔒	🛑		
└	_afterTokenTransfer	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 🔒			

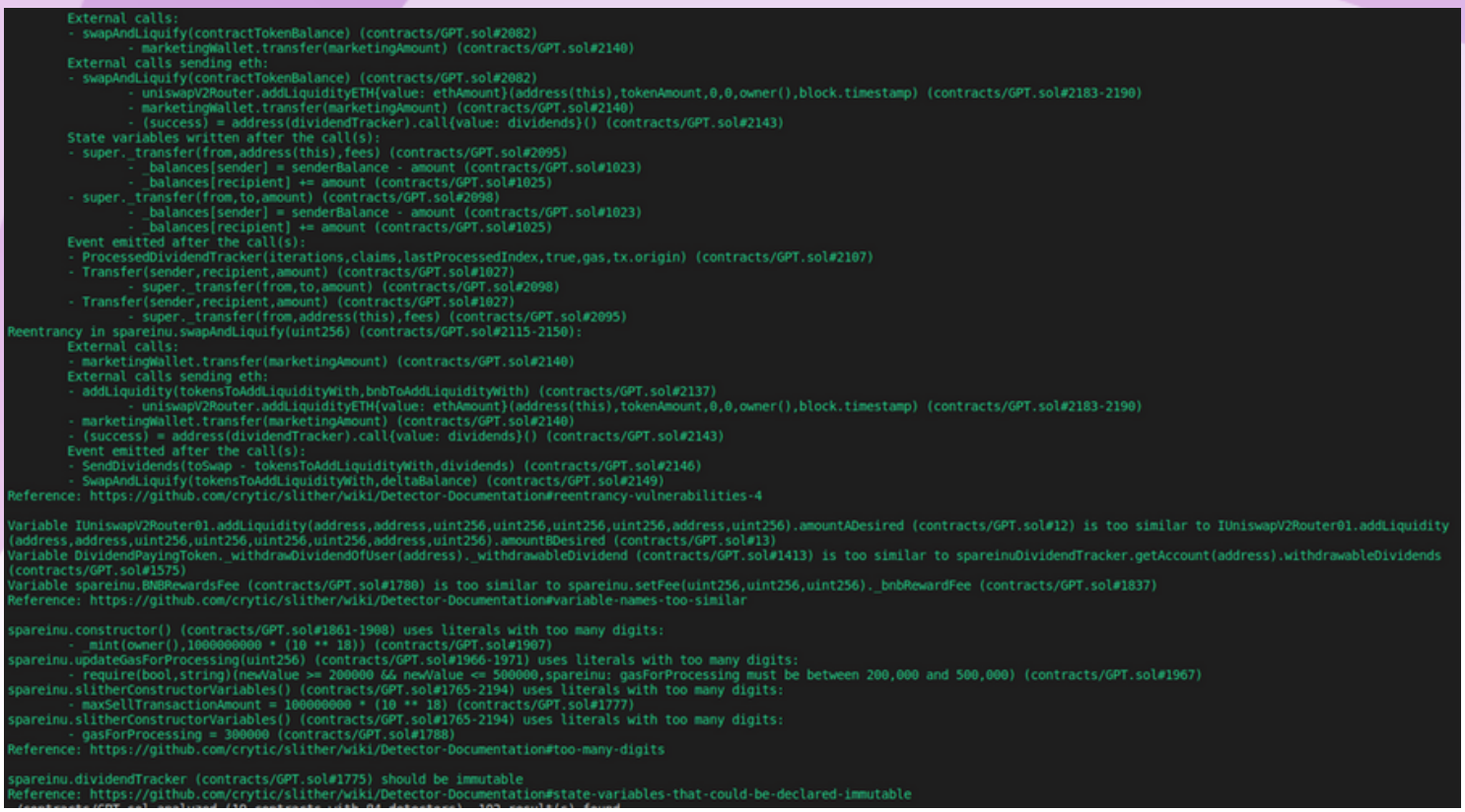
CONTRACT ASSESMENT

| └ | _msgData | Internal 🔒 | | |

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
        External calls:
        - swapAndLiquify(contractTokenBalance) (contracts/GPT.sol#2082)
                - marketingWallet.transfer(marketingAmount) (contracts/GPT.sol#2140)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (contracts/GPT.sol#2082)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/GPT.sol#2183-2190)
                - marketingWallet.transfer(marketingAmount) (contracts/GPT.sol#2140)
                - (success) = address(dividendTracker).call{value: dividends}() (contracts/GPT.sol#2143)
        State variables written after the call(s):
        - super._transfer(from,address(this),fees) (contracts/GPT.sol#2095)
                - _balances[sender] = senderBalance - amount (contracts/GPT.sol#1023)
                - _balances[recipient] += amount (contracts/GPT.sol#1025)
        - super._transfer(from,to,amount) (contracts/GPT.sol#2098)
                - _balances[sender] = senderBalance - amount (contracts/GPT.sol#1023)
                - _balances[recipient] += amount (contracts/GPT.sol#1025)
        Event emitted after the call(s):
        - ProcessedDividendTracker(iterations,claims,lastProcessedIndex,true,gas,tx.origin) (contracts/GPT.sol#2107)
        - Transfer(sender,recipient,amount) (contracts/GPT.sol#1027)
                - super._transfer(from,to,amount) (contracts/GPT.sol#2098)
        - Transfer(sender,recipient,amount) (contracts/GPT.sol#1027)
                - super._transfer(from,address(this),fees) (contracts/GPT.sol#2095)
Reentrancy in spareinu.swapAndLiquify(uint256) (contracts/GPT.sol#2115-2150):
        External calls:
        - marketingWallet.transfer(marketingAmount) (contracts/GPT.sol#2140)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (contracts/GPT.sol#2137)
                - uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (contracts/GPT.sol#2183-2190)
        - marketingWallet.transfer(marketingAmount) (contracts/GPT.sol#2140)
        - (success) = address(dividendTracker).call{value: dividends}() (contracts/GPT.sol#2143)
        Event emitted after the call(s):
        - SendDividends(toSwap - tokensToAddLiquidityWith,dividends) (contracts/GPT.sol#2146)
        - SwapAndLiquify(tokensToAddLiquidityWith,deltaBalance) (contracts/GPT.sol#2149)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/GPT.sol#12) is too similar to IUniswapV2Router01.addLiquidity
(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/GPT.sol#13)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (contracts/GPT.sol#1413) is too similar to spareinuDividendTracker.getAccount(address).withdrawableDividends
(contracts/GPT.sol#1575)
Variable spareinu.BNBRewardsFee (contracts/GPT.sol#1780) is too similar to spareinu.setFee(uint256,uint256,uint256)._bnbRewardFee (contracts/GPT.sol#1837)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

spareinu.constructor() (contracts/GPT.sol#1861-1908) uses literals with too many digits:
        - _mint(owner(),1000000000 * (10 ** 18)) (contracts/GPT.sol#1907)
spareinu.updateGasForProcessing(uint256) (contracts/GPT.sol#1966-1971) uses literals with too many digits:
        - require(bool,string)(newValue >= 200000 && newValue <= 500000,spareinu: gasForProcessing must be between 200,000 and 500,000) (contracts/GPT.sol#1967)
spareinu.slitherConstructorVariables() (contracts/GPT.sol#1765-2194) uses literals with too many digits:
        - maxSellTransactionAmount = 100000000 * (10 ** 18) (contracts/GPT.sol#1777)
spareinu.slitherConstructorVariables() (contracts/GPT.sol#1765-2194) uses literals with too many digits:
        - gasForProcessing = 300000 (contracts/GPT.sol#1788)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

spareinu.dividendTracker (contracts/GPT.sol#1775) 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

Router (PCS V2):

0xD99D1c33F9fC3444f8101754aBC46c52416550D1

All the functionalities have been tested, no issues were found

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xdae9ddf32e607b1d9cf558ece9641127aeaac30194ca0e002385b4f497d2b256

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

https://testnet.bscscan.com/tx/0x2137eb8d47c016980ee3f3c5a3602ba06bda86ea45f28c2d0d28ac6ca5bffd9b

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

https://testnet.bscscan.com/tx/0x5f9f9b77ac84cc2fda60c5edc64b5e55ccd379f9cee3a782bae90dbd56e7265e

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

https://testnet.bscscan.com/tx/0x59530856e088f7d51ddb031b33dc285cf39f5173104fa2f577871cb02cad2234

5- Buying when not excluded from fees (up to 100% tax) (passed):

https://testnet.bscscan.com/tx/0x9934e80450224b711e064146b4637ea27ab76dd552127fcce2ab02643672f099

6- Selling when not excluded from fees (up to 100% tax) (passed):

https://testnet.bscscan.com/tx/0x6db0238506fae4c70ecbe1e7931743143271c933aef738c72c7a7ad5ca6c950f

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (up to 100% tax)(passed):

https://testnet.bscscan.com/tx/0x303077fae913c9f8025f5d2428a7b99c1a956cb2a143986bb1512623ee547b31

8- Internal swap (passed):

Auto-liquidty & Reflections & fee wallets receiving BNB

https://testnet.bscscan.com/tx/0xe8e15600504999e4e53031cbf77c877a17f6fcf534d97ab4db37ff4efa99f1b9

MANUAL TESTING

Centralization – Disabling sells

Severity: Critical
Function: setMaxSelltx
Status: Not Resolved
Overview:
The contract owner can set the maximum sell amount to 0, effectively disabling all sales for everyone except whitelisted (allowed) wallets.

```
function setMaxSelltx(uint256 _maxSellTxAmount) public onlyOwner {
    maxSellTransactionAmount = _maxSellTxAmount;
}
```

Recommendation:
To address this issue, consider the following options:

- Ensure that the maximum sell amount always remains above a reasonable minimum amount. According to PinkSale Safu criteria, this minimum safeguard is 0.1% of the total supply.
- Renounce ownership of the contract.

MANUAL TESTING

Centralization – Excessive buy/sell/trasfer fees

Severity: Critical
Function: setFee - setExtraFeeOnSell
Lines: 1837-1845
Status: Not Resolved
Overview:
The contract owner can set buy, sell, and transfer fees up to 100%, effectively rendering trading unreasonable.

```
function setFee(uint256 _bnbRewardFee, uint256 _liquidityFee, uint256       _marketingFee) public onlyOwner {
    BNBRewardsFee = _bnbRewardFee;
    liquidityFee = _liquidityFee;
    marketingFee = _marketingFee;

    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee); // total              fee transfer and buy
}

function setExtraFeeOnSell(uint256 _extraFeeOnSell) public onlyOwner {
    extraFeeOnSell = _extraFeeOnSell; // extra fee on sell
}
```

Recommendation:
To mitigate this issue, consider the following options:

- Ensure that the total buy/sell/transfer fee cannot exceed a reasonable limit. According to PinkSale Safu criteria, these limits are 25% for (buy + sell in total) and a maximum of 25% for transfers.
- Renounce ownership of the contract.

MANUAL TESTING

Centralization – Owner must enable trades

Severity: High
Function: openTrade
Status: Not Resolved
Overview:
The contract owner is required to enable trades for investors. If trading remains disabled, token holders will be unable to buy, sell, or transfer tokens.

```
function openTrade() external onlyOwner {
    isOpen = true;
}
```

Recommendation:
To address this issue, consider the following options:

- Ensure that trade activation is guaranteed by temporarily transferring contract ownership to a trusted third party, such as a certified PinkSale Safu developer.
- Remove this function and allow investors to trade their tokens immediately after adding liquidity.

MANUAL TESTING

Logical – changing router to an invalid contract can disable sell/transfer

Severity: High
Function: updateUniswapV2Router
Status: Not Resolved
Overview:
Setting the router address (PCS v2 by default) to an invalid address (e.g., a dead wallet) can disable sell/transfer operations, as any attempt to swap the contract's internal balance to BNB will fail.

```
function swapTokensForBnb(uint256 tokenAmount, address _to) private {
    // generate the uniswap pair path of token -> weth
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();

    if(allowance(address(this), address(uniswapV2Router)) < tokenAmount) {
      _approve(address(this), address(uniswapV2Router), ~uint256(0));
    }

    // make the swap
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
      tokenAmount,
      0, // accept any amount of ETH
      path,
      _to,
      block.timestamp
    );

}

function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(newAddress != address(uniswapV2Router), "spareinu: The router                already has that address");
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
}
```

Recommendation:
To address this issue, consider the following options:

- Ensure that the new router is a valid contract that supports the PCS v2 interface.
- Make the router address immutable and unable to be changed later.
- Renounce ownership of the contract.

MANUAL TESTING

Centralization – LP tokens sent to an EOA

Severity: Medium
Function: addLiquidity
Status: Not Resolved
Overview:
The contract owner's wallet (an EOA or Externally Owned Account) receives auto-generated liquidity pool share tokens. Over time, these tokens will accumulate and could potentially be used to remove a significant portion of liquidity, negatively impacting the token's liquidity and price.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );

}
```

Recommendation:
To mitigate this issue, consider the following options:

- Burn new LP tokens.
- Lock new LP tokens.
- Renounce ownership of the contract, which would result in LP tokens being burned automatically.

MANUAL TESTING

Logical – 0 Swapthreshold can disable some sells and transfers or increase slippage if is set to a high value

Severity: Medium
Function: setSWapTokensAtAmount - swapAndLiquify
Lines: 2188
Status: Not Resolved
Overview:
Setting the **swapTokensAtAmount** to 0 can disable some sell/transfer operations if the contract's token balance is 0. This is because swapAndLiquify does not check whether the contract token balance is greater than 0 and attempts to swap 0 tokens for BNB. Additionally, setting the swap threshold to a very high value can increase slippage on sell/transfer operations, potentially exposing traders' tokens to front-runner bots.

```
function setSWapTokensAtAmount(uint256 _newAmount) public onlyOwner {
    swapTokensAtAmount = _newAmount;
}
```

Recommendation:
there are several ways to mitigate this issue

- Ensure that the new swap threshold cannot be set to an unreasonably low or high value.
- Renounce ownership of the contract, which would prevent the swap threshold from being changed arbitrarily.

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