auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

JOKER PEPE

DATED : 27 May 23'

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/0xbc6cde05b595b27a7827897cfb67c0143a4de79f

Token Information

Token Name : JOKER PEPE

Token Symbol: JOP

Decimals: 9

Token Supply: 10,000,000,000,000

Token Address:
0x8c550F10A2f26313c76A22C5c90aBd298054aAD9

Checksum:
f803552931fc9b646677b37fdd16ec74396fdc29

Owner:
0x3f1006d2ee3691546EA4451a650B09E4565ec469

Deployer:
0x3f1006d2ee3691546EA4451a650B09E4565ec469

TOKEN OVERVIEW

Fees:

Buy Fees: 0-18%

Sell Fees: 0-18%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: 0x3f1006d2ee3691546EA4451a650B09E4565ec469

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: Fee modification

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

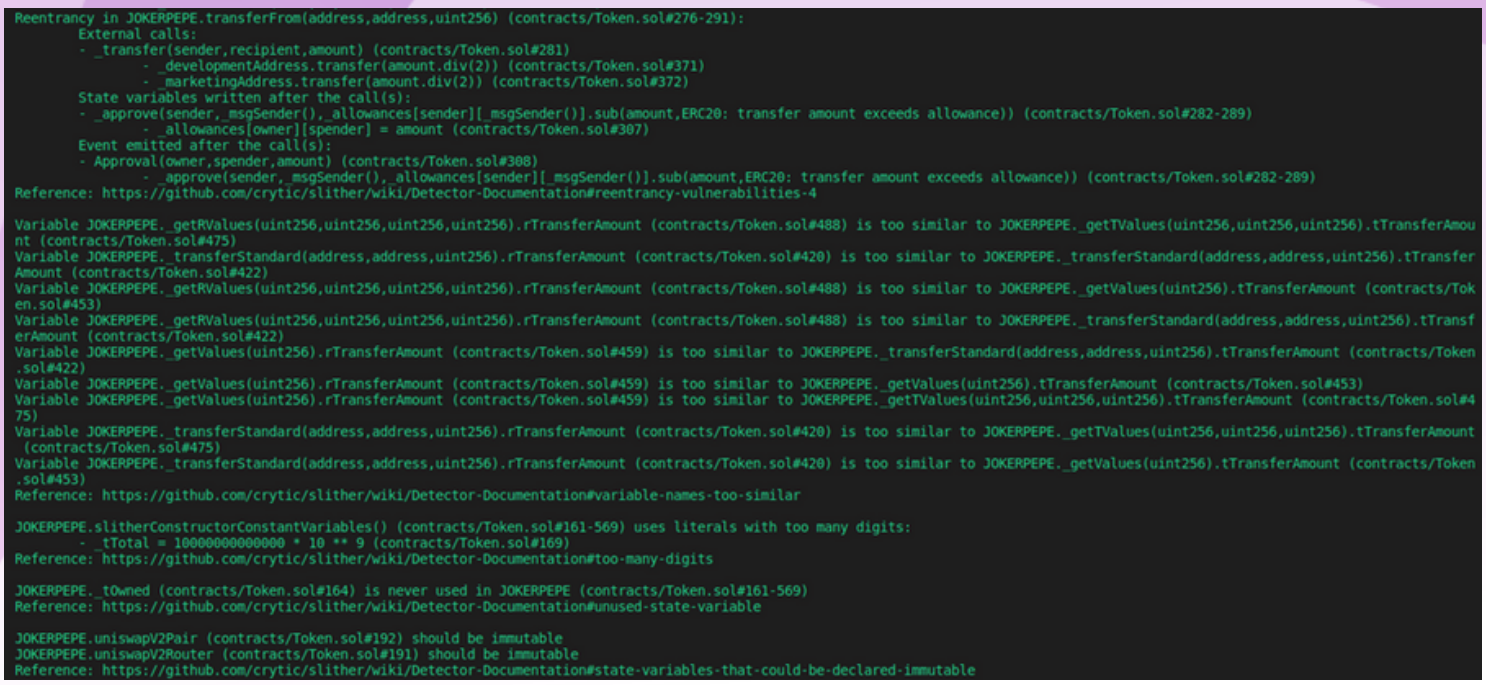
Severity	Found
Critical	0
High-Risk	0
Medium-Risk	1
Low-Risk	0
Gas Optimization / Suggestions	2

INHERITANCE TREE

POINTS TO NOTE

- Owner is able to change fees in range 0-18% for buy and sells (0% transfer tax)
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to limit buy/sell/transfer/wallet amounts
- Owner is not able to mint new token

STATIC ANALYSIS



```
Reentrancy in JOKERPEPE.transferFrom(address,address,uint256) (contracts/Token.sol#276-291):
        External calls:
        - _transfer(sender,recipient,amount) (contracts/Token.sol#281)
                - _developmentAddress.transfer(amount.div(2)) (contracts/Token.sol#371)
                - _marketingAddress.transfer(amount.div(2)) (contracts/Token.sol#372)
        State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/Token.sol#282-289)
                - _allowances[owner][spender] = amount (contracts/Token.sol#307)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (contracts/Token.sol#308)
                - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/Token.sol#282-289)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

Variable JOKERPEPE._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#488) is too similar to JOKERPEPE._getTValues(uint256,uint256,uint256).tTransferAmou
nt (contracts/Token.sol#475)
Variable JOKERPEPE._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#420) is too similar to JOKERPEPE._transferStandard(address,address,uint256).tTransfer
Amount (contracts/Token.sol#422)
Variable JOKERPEPE._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#488) is too similar to JOKERPEPE._getValues(uint256).tTransferAmount (contracts/Tok
en.sol#453)
Variable JOKERPEPE._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (contracts/Token.sol#488) is too similar to JOKERPEPE._transferStandard(address,address,uint256).tTransf
erAmount (contracts/Token.sol#422)
Variable JOKERPEPE._getValues(uint256).rTransferAmount (contracts/Token.sol#459) is too similar to JOKERPEPE._transferStandard(address,address,uint256).tTransferAmount (contracts/Token
.sol#422)
Variable JOKERPEPE._getValues(uint256).rTransferAmount (contracts/Token.sol#459) is too similar to JOKERPEPE._getValues(uint256).tTransferAmount (contracts/Token.sol#453)
Variable JOKERPEPE._getValues(uint256).rTransferAmount (contracts/Token.sol#459) is too similar to JOKERPEPE._getTValues(uint256,uint256,uint256).tTransferAmount (contracts/Token.sol#4
75)
Variable JOKERPEPE._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#420) is too similar to JOKERPEPE._getTValues(uint256,uint256,uint256).tTransferAmount
 (contracts/Token.sol#475)
Variable JOKERPEPE._transferStandard(address,address,uint256).rTransferAmount (contracts/Token.sol#420) is too similar to JOKERPEPE._getValues(uint256).tTransferAmount (contracts/Token
.sol#453)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

JOKERPEPE.slitherConstructorConstantVariables() (contracts/Token.sol#161-569) uses literals with too many digits:
        - _tTotal = 10000000000000 * 10 ** 9 (contracts/Token.sol#169)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

JOKERPEPE._tOwned (contracts/Token.sol#164) is never used in JOKERPEPE (contracts/Token.sol#161-569)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

JOKERPEPE.uniswapV2Pair (contracts/Token.sol#192) should be immutable
JOKERPEPE.uniswapV2Router (contracts/Token.sol#191) 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

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
Token	Interface			
└	transferFrom	External ❗	🛑	NO ❗
└	transfer	External ❗	🛑	NO ❗
IUniswapV2Factory	Interface			
└	createPair	External ❗	🛑	NO ❗
IUniswapV2Router02	Interface			
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
Context	Implementation			
└	_msgSender	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
JOKERPEPE	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗

CONTRACT ASSESMENT

┗	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 ❗
┗	tokenFromReflection	Private 🔐		
┗	_approve	Private 🔐	🛑	
┗	_transfer	Private 🔐	🛑	
┗	swapTokensForEth	Private 🔐	🛑	lockTheSwap
┗	sendETHToFee	Private 🔐	🛑	
┗	_tokenTransfer	Private 🔐	🛑	
┗	rescueForeignTokens	Public ❗	🛑	onlyDev
┗	setNewDevAddress	Public ❗	🛑	onlyDev
┗	setNewMarketingAddress	Public ❗	🛑	onlyDev
┗	_transferStandard	Private 🔐	🛑	
┗	_takeTeam	Private 🔐	🛑	
┗	_reflectFee	Private 🔐	🛑	
┗	<Receive Ether>	External ❗	💵 |NO ❗
┗	_getValues	Private 🔐		
┗	_getTValues	Private 🔐		
┗	_getRValues	Private 🔐		
┗	_getRate	Private 🔐		
┗	_getCurrentSupply	Private 🔐		
┗	manualswap	External ❗	🛑 |NO ❗
┗	manualsend	External ❗	🛑 |NO ❗
┗	setFee	Public ❗	🛑	onlyDev
┗	toggleSwap	Public ❗	🛑	onlyDev
┗	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x08648068a8aa60eccee0ad905f23b0c4c1573b111bd0431b01572b7be3c2980e

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

https://testnet.bscscan.com/tx/0xe0d4089a11b3eee9caad5d6335de3b74196931b75de932237eb4ef50da288b7b

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

https://testnet.bscscan.com/tx/0xf0eadd638048fd1f70ba88e1bdf3ce6091a0d03fb061563864567e975f6e1816

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

https://testnet.bscscan.com/tx/0xe2a918249d4a9ef7aedd24a80edc4e32531841e2e905e4f67ea55d719f636535

5- Buying from a regular wallet (0-18% tax) (passed):

https://testnet.bscscan.com/tx/0x959be087d1169e21b867f231871024fefb2477e2a3cdadf85cf5d76ad153d6b7

6- Selling from a regular wallet (0-18% tax) (passed):

https://testnet.bscscan.com/tx/0xe3f07235135d4e600c037150089976c64d2bea594180db7caeeecacd9049cfe3

FUNCTIONAL TESTING

7- Transferring from a regular wallet (0% tax) (passed):

https://testnet.bscscan.com/tx/0x8c710d0371bee6d61c1e6ca500d54c4a16ccfca8801ab47c8bf687a005ca15b7

7- Internal swap (marketing and development wallets received BNB) (passed):

https://testnet.bscscan.com/address/0x3f1006d2ee3691546EA4451a650B09E4565ec469#internaltx

MANUAL TESTING

Issue Category: Centralization – Excessive Fee

Severity: Medium

Function: setFee

Status: Not Resolved

Overview: The function setFee allows the owner to set the transaction fees for buying and selling. This buying and selling fee can be in range 0-18% for each type of tax (buy or sell)

Code:

```
solidity

function setFee(
    uint256 redisFeeOnBuy,
    uint256 redisFeeOnSell,
    uint256 taxFeeOnBuy,
    uint256 taxFeeOnSell
) public onlyDev {
    require(redisFeeOnBuy < 9);
    require(redisFeeOnSell < 9);
    require(taxFeeOnBuy < 9);
    require(taxFeeOnSell < 9);
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}
```

Suggestion: Limit the maximum fee that can be set to prevent the owner from setting an excessively high fee. This can be done by modifying the require statements to a reasonable percentage.

Buy Total Fee <= 10

Sell Total Fee <= 10
Tranfser Total Fee <= 10

MANUAL TESTING

Issue Category: Efficiency - High Slippage Risk

Severity: Minor

Function: _transfer

Status: Not Resolved

Overview: In the _transfer function, it appears that the contract will try to swap all tokens in its balance for ETH whenever a transfer occurs that does not involve the owner. This can potentially lead to a situation where, if a large amount of tokens are accumulated in the contract, a huge amount of tokens will be swapped at once, leading to a high slippage. Its expected that during launch of the token, a huge amount of token get accumulated in the contract depending on the buy volume which leads to a high slippage in sell transactions (8-49%).

Code:

solidity

```
function _transfer(address from, address to, uint256 amount) private {
  require(from != address(0), "ERC20: transfer from the zero address");
  require(to != address(0), "ERC20: transfer to the zero address");
  require(amount > 0, "Transfer amount must be greater than zero");

  _redisFee = 0;
  _taxFee = 0;

  if (from != owner() && to != owner()) {
    uint256 contractTokenBalance = balanceOf(address(this));
    if (
      !inSwap &&
      from != uniswapV2Pair &&
      swapEnabled &&
      contractTokenBalance > 0
    ) {
      swapTokensForEth(contractTokenBalance);
      uint256 contractETHBalance = address(this).balance;
      if (contractETHBalance > 0) {
        sendETHToFee(address(this).balance);
      }
    }
  }
}
```

Suggestion: To mitigate the risk of high slippage, you might consider setting a swap threshold, i.e., a maximum amount of tokens that can be swapped at any single transaction. This can help prevent the contract from swapping a massive amount of tokens at once, thus preventing an excessive impact on the token's price.

MANUAL TESTING

Issue Category: Centralization – Unrestricted Withdrawal

Severity: Informational

Function: rescueForeignTokens

Status: Not Applicable

Overview: The owner has unrestricted access to withdraw any tokens from the contract. This poses a risk as it allows the owner to withdraw native tokens from the contract

Code:

```
function rescueForeignTokens(
    address _tokenAddr,
    address _to,
    uint _amount
) public onlyDev {
    emit tokensRescued(_tokenAddr, _to, _amount);
    Token(_tokenAddr).transfer(_to, _amount);
}
```

Suggestion: Implement checks and balances on the owner's ability to withdraw tokens from the contract. This could be achieved by establishing multisig control, time locks, or by setting a withdrawal limit.

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