



Smart Contract Audit

FOR
JolteonInu

DATED : 26 June 23'

HIGH RISK FINDING

Centralization – Trades must be enabled

Severity: **High**

function: EnableTrading

Status: Not Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function startTrading() external onlyOwner {  
    require(!tradingEnabled, "Trading already enabled");  
    tradingEnabled = true;  
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades



AUDIT SUMMARY

Project name – JolteonInu

Date: 26 June, 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	1	0	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/0xaaA5A90881D61211Fab23969b25ad4D49a183b45>



Token Information

Token Name : JolteonInu

Token Symbol: JINU

Decimals: 18

Token Supply: 99,000,000

Token Address:

0xef9c8F844217C862D543CF37407f7f19ADa8D789

Checksum:

6bcc7c30b8c14cbb5151675488fa6710283eee8e

Owner:

0x7dA26D035f6544c6FC4a26C42E591e4fB8C79E83
(at time of writing the audit)

Deployer:

0x9791acd338DbDCE96aF191236e5760B46E606248



TOKEN OVERVIEW

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0%

Fees Privilege: Owner

Ownership: Owned

Minting: none

Max Tx Amount/ Max Wallet Amount: Yes

Blacklist: No

Other Privileges: - initial distribution of tokens

- including or excluding from fees

- changing swap threshold

- enabling trades

- modifying fees



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 is exercised 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 | ✓ Gasless Send |
| ✓ Private modifier | ✓ Using block.timestamp |
| ✓ Multiple Sends | ✓ Re-entrancy |
| ✓ Using Suicide | ✓ Tautology or contradiction |
| ✓ Gas Limitand Loops | ✓ Timestamp Dependence |
| ✓ Address hardcoded | ✓ Revert/require functions |
| ✓ Exception Disorder | ✓ Use of tx.origin |
| ✓ Using inline assembly | ✓ Integer overflow/underflow |
| ✓ Divide before multiply | ✓ Dangerous strict equalities |
| ✓ Missing Zero Address Validation | ✓ Using SHA3 |
| ✓ Compiler version not fixed | ✓ 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

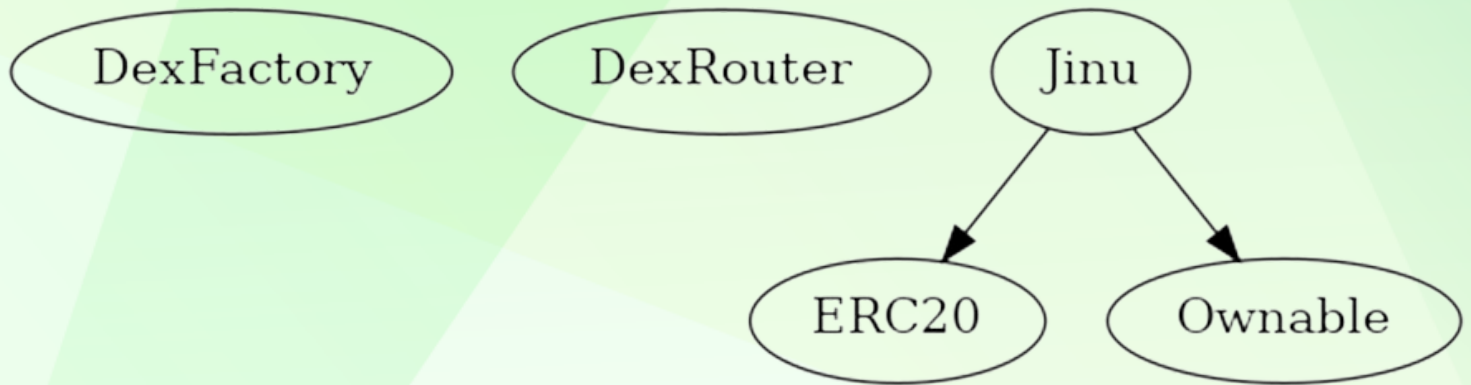
◆ Low-Risk

0

◆ Gas Optimization / Suggestions

0

INHERITANCE TREE





POINTS TO NOTE

- **Owner is not able to change buy/sell/transfer fees more than 10% each**
 - Owner is not able to blacklist an arbitrary address.
 - Owner is not able to disable trades
 - Owner is not able to set max buy/sell/transfer/hold amount to 0
 - Owner is not able to mint new tokens
 - **Owner must enable trades manually**
-



CONTRACT ASSESMENT

Contract	Type	Bases			
:-----: :-----: :-----: :-----: :-----:					
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**	
DexFactory Interface					
L	createPair	External	!	●	NO !
DexRouter Interface					
L	factory	External	!		NO !
L	WETH	External	!		NO !
L	addLiquidityETH	External	!	💵	NO !
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External	!	●	NO !
Jinu Implementation ERC20, Ownable					
L	<Constructor>	Public	!	●	ERC20
L	setmMrketingWallet	External	!	●	onlyOwner
L	setBuyBackWallet	External	!	●	onlyOwner
L	setBuyTaxes	External	!	●	onlyOwner
L	setSellTaxes	External	!	●	onlyOwner
L	setTransferFees	External	!	●	onlyOwner
L	setSwapTokensAtAmount	External	!	●	onlyOwner
L	toggleSwapping	External	!	●	onlyOwner
L	setWhitelistStatus	External	!	●	onlyOwner
L	checkWhitelist	External	!		NO !
L	startTrading	External	!	●	onlyOwner
L	_takeTax	Internal	🔒	●	
L	_transfer	Internal	🔒	●	
L	internalSwap	Internal	🔒	●	
L	swapToETH	Internal	🔒	●	
L	withdrawStuckETH	External	!	●	onlyOwner
L	withdrawStuckTokens	External	!	●	onlyOwner
L	<Receive Ether>	External	!	💵	NO !

| Symbol | Meaning |

|:-----:|:-----:|

| ● | Function can modify state |

| 💵 | Function is payable |



STATIC ANALYSIS

```
Context._msgData() (contracts/Token.sol#108-110) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#427-443) is never used and should be removed
SafeMath.add(uint256,uint256) (contracts/Token.sol#613-615) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Token.sol#655-657) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#707-712) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#671-673) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#729-734) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Token.sol#641-643) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#627-629) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (contracts/Token.sol#688-693) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#542-548) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#584-589) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#596-601) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#567-577) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#555-560) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
Pragma version^0.8.17 (contracts/Token.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.0 (contracts/Token.sol#91) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#120) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#153) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#524) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#744) allows old versions
Pragma version^0.8.17 (contracts/Token.sol#836) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in Jinu.internalSwap() (contracts/Token.sol#1017-1042):
- (success) = marketingWallet.call{value: (totalMarketingFee * received) / totalShares}() (contracts/Token.sol#1035)
- (success) = buybackWallet.call{value: address(this).balance}() (contracts/Token.sol#1040)
Low level call in Jinu.withdrawStuckETH() (contracts/Token.sol#1055-1058):
- (success) = address(msg.sender).call{value: address(this).balance}() (contracts/Token.sol#1056)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
Function DexRouter.WETH() (contracts/Token.sol#845) is not in mixedCase
Event JinumarketingWalletChanged(address) (contracts/Token.sol#898) is not in CapWords
Parameter Jinu.setMkrketingWallet(address)._newMarketing (contracts/Token.sol#920) is not in mixedCase
Parameter Jinu.setBuyBackWallet(address)._newBuyBack (contracts/Token.sol#925) is not in mixedCase
Parameter Jinu.setBuyTaxes(uint256,uint256)._marketingTax (contracts/Token.sol#930) is not in mixedCase
Parameter Jinu.setBuyTaxes(uint256,uint256)._buybackTax (contracts/Token.sol#930) is not in mixedCase
Parameter Jinu.setSellTaxes(uint256,uint256)._marketingTax (contracts/Token.sol#937) is not in mixedCase
Parameter Jinu.setSellTaxes(uint256,uint256)._buybackTax (contracts/Token.sol#937) is not in mixedCase
Parameter Jinu.setTransferFees(uint256,uint256)._marketingTax (contracts/Token.sol#944) is not in mixedCase
Parameter Jinu.setTransferFees(uint256,uint256)._buybackTax (contracts/Token.sol#944) is not in mixedCase
Parameter Jinu.setSwapTokensAtAmount(uint256)._newAmount (contracts/Token.sol#951) is not in mixedCase
Parameter Jinu.setWhitelistStatus(address,bool)._wallet (contracts/Token.sol#964) is not in mixedCase
Parameter Jinu.setWhitelistStatus(address,bool)._status (contracts/Token.sol#964) is not in mixedCase
Parameter Jinu.checkWhitelist(address)._wallet (contracts/Token.sol#969) is not in mixedCase
Parameter Jinu.swapToETH(uint256)._amount (contracts/Token.sol#1044) is not in mixedCase
Parameter Jinu.withdrawStuckTokens(address).BEP20_token (contracts/Token.sol#1060) is not in mixedCase
Constant Jinu._totalSupply (contracts/Token.sol#871) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
Jinu.slitherConstructorVariables() (contracts/Token.sol#865-1066) uses literals with too many digits:
- swapTokensAtAmount = _totalSupply / 100000 (contracts/Token.sol#889)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

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

1- Adding liquidity (passed):

<https://testnet.bscscan.com/tx/0xf5a0d82f849848dcc70da712694686cd5902c831230355ca739f71d768ee4fe9>

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

<https://testnet.bscscan.com/tx/0xfbb63181be78b9e4b031ab056cd10470195b683d77c22c096dc052789dbde51a>

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

<https://testnet.bscscan.com/tx/0xbc0a62ed4e097699dfb4cff8a9dbde76530817ddb8879cf0d334592e3ebb4525>

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

<https://testnet.bscscan.com/tx/0xfd5a9eb77cc454fd5c39959ed33bcd7e7c6baee5ec91f976a29423cba6174f9>

5- Buying when not excluded from fees (0-10% tax) (passed):

<https://testnet.bscscan.com/tx/0x254b4f59077f7007ad3bbba29a17189db097e6975c770ece3e7af004edd20a96>

6- Selling when not excluded from fees (0-10% tax) (passed):

<https://testnet.bscscan.com/tx/0x783d32312a4f993c9fe7a6f614e3f6e100db7a570d11cc3b400053f8d40f3210>



FUNCTIONAL TESTING

7- Transferring when not excluded from fees (0-10% tax) (passed):

<https://testnet.bscscan.com/tx/0x4c0908a5c7ac5c8acb820bff522ac2976e9f979238a7c4056a0a3158550c4bf8>

8- Internal swap (marketing and buyback wallets received bnb)

(passed):

<https://testnet.bscscan.com/address/0x682b31181c5eeb523c106976343d56a19325ce66#internaltx>

<https://testnet.bscscan.com/address/0x7cdd2bf4e128832a544bb4a3f869131aaca68d54#internaltx>

FUNCTIONAL TESTING

Centralization – Trades must be enabled

Severity: **High**

function: EnableTrading

Status: Not Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function startTrading() external onlyOwner {  
    require(!tradingEnabled, "Trading already enabled");  
    tradingEnabled = true;  
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades



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 specialize 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>
