auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

DOGECAT

DATED : 9 June 23'

HIGH RISK FINDING

Centralization – Trades must be enabled

Severity: **High**

function: enableTrading

Status: **Resolved (Contract is owned by safu developer)**

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 enableTrading() external onlyOwner {
    require(!isTradeEnabled, "Trading already enabled");
    isTradeEnabled = true;
    listingTime = block.timestamp;
}
```

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 – DOGECAT

Date: 9 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	1
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/0x6179E34f232B27359F10EBC7d699aAc60Bf48f56

Token Information

Token Name : DogeCatChat

Type your text

Token Symbol: DOGECAT

Decimals: 9

Token Supply:420,420,420,420

Token Address:
0x9651BdAc7Cc1BEF2789EC1350fcaC9644F72CC49

Checksum:
b0534ddce337345588aa005e78ff6b65a1887a13

Owner:
0x530D6de7fD1461448568bbfBaa8B1BF785a038aC
(at time of writing the audit)

Deployer:
0x530D6de7fD1461448568bbfBaa8B1BF785a038aC

TOKEN OVERVIEW

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0-10%

Fees Privilege: Owner

Ownership:

0x1Df925D52a7dade6411Cf669fa796F1e62bBc509

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
- changing 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

- ✓ 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

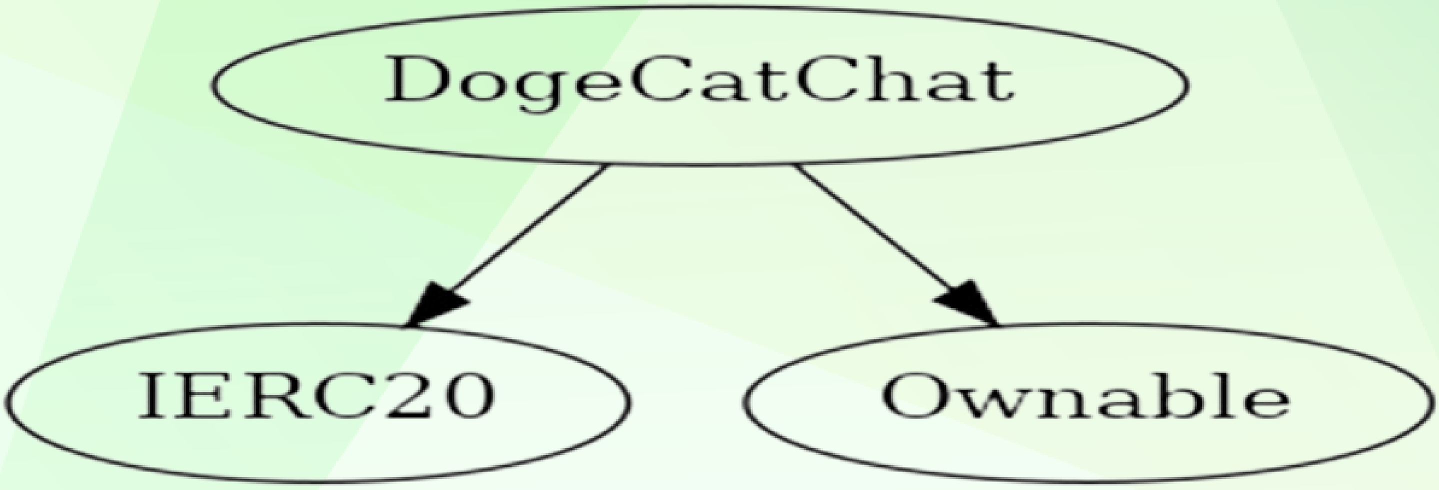
CLASSIFICATION OF RISK

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	1
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	1

INHERITANCE TREE



POINTS TO NOTE

- owner is able to change fees in range of 0-10% for buy/sell/transfer transactions.
- owner is not able to change buy/sell/transfer fees until 7 days after launch
- transfser fees are disabled at time of writing the audit report
- owner is not able to blacklist an arbitrary wallet
- owner is not able to set limit for buy/sell/transfer/holding amounts
- owner is not able to mint new tokens
- owner is not able to disable trades
- **owner must enable trades manually**

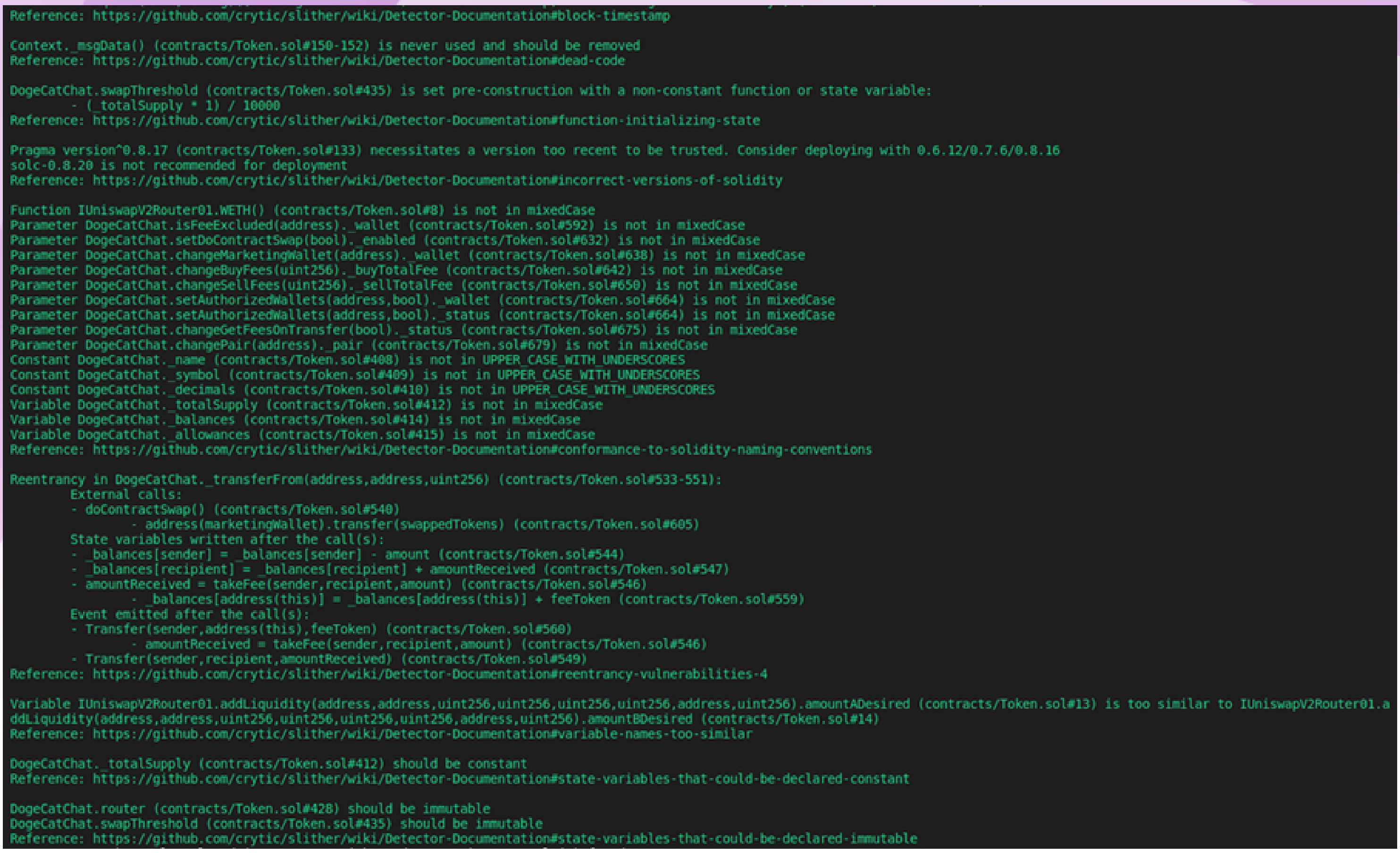
CONTRACT ASSESMENT

```
| Contract | Type | Bases | | |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
||||||
| **DogeCatChat** | Implementation | IERC20, Ownable |||
| └ | <Constructor> | Public ❗ | 🛑 |NO ❗ |
| └ | <Receive Ether> | External ❗ | 💵 |NO ❗ |
| └ | totalSupply | External ❗ | |NO ❗ |
| └ | name | Public ❗ | |NO ❗ |
| └ | symbol | Public ❗ | |NO ❗ |
| └ | decimals | Public ❗ | |NO ❗ |
| └ | balanceOf | Public ❗ | |NO ❗ |
| └ | allowance | External ❗ | |NO ❗ |
| └ | approve | Public ❗ | 🛑 |NO ❗ |
| └ | _approve | Internal 🔒 | 🛑 | |
| └ | approveMax | External ❗ | 🛑 |NO ❗ |
| └ | transfer | External ❗ | 🛑 |NO ❗ |
| └ | transferFrom | External ❗ | 🛑 |NO ❗ |
| └ | _transferFrom | Internal 🔒 | 🛑 | |
| └ | takeFee | Internal 🔒 | 🛑 | |
| └ | _basicTransfer | Internal 🔒 | 🛑 | |
| └ | shouldTakeFee | Internal 🔒 | | |
| └ | shouldDoContractSwap | Internal 🔒 | | |
| └ | isFeeExcluded | Public ❗ | |NO ❗ |
| └ | doContractSwap | Internal 🔒 | 🛑 | swapping |
| └ | swapTokensForEth | Private 🔐 | 🛑 | |
| └ | setIsFeeExempt | External ❗ | 🛑 | onlyOwner |
| └ | setDoContractSwap | External ❗ | 🛑 | onlyOwner |
| └ | changeMarketingWallet | External ❗ | 🛑 | onlyOwner |
| └ | changeBuyFees | External ❗ | 🛑 | onlyOwner |
| └ | changeSellFees | External ❗ | 🛑 | onlyOwner |
| └ | enableTrading | External ❗ | 🛑 | onlyOwner |
| └ | setAuthorizedWallets | External ❗ | 🛑 | onlyOwner |
| └ | rescueBNB | External ❗ | 🛑 | onlyOwner |
| └ | changeGetFeesOnTransfer | External ❗ | 🛑 | onlyOwner |
| └ | changePair | External ❗ | 🛑 | onlyOwner |
```

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Context._msgData() (contracts/Token.sol#150-152) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

DogeCatChat.swapThreshold (contracts/Token.sol#435) is set pre-construction with a non-constant function or state variable:
        - (_totalSupply * 1) / 10000
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state

Pragma version^0.8.17 (contracts/Token.sol#133) 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

Function IUniswapV2Router01.WETH() (contracts/Token.sol#8) is not in mixedCase
Parameter DogeCatChat.isFeeExcluded(address)._wallet (contracts/Token.sol#592) is not in mixedCase
Parameter DogeCatChat.setDoContractSwap(bool)._enabled (contracts/Token.sol#632) is not in mixedCase
Parameter DogeCatChat.changeMarketingWallet(address)._wallet (contracts/Token.sol#638) is not in mixedCase
Parameter DogeCatChat.changeBuyFees(uint256)._buyTotalFee (contracts/Token.sol#642) is not in mixedCase
Parameter DogeCatChat.changeSellFees(uint256)._sellTotalFee (contracts/Token.sol#650) is not in mixedCase
Parameter DogeCatChat.setAuthorizedWallets(address,bool)._wallet (contracts/Token.sol#664) is not in mixedCase
Parameter DogeCatChat.setAuthorizedWallets(address,bool)._status (contracts/Token.sol#664) is not in mixedCase
Parameter DogeCatChat.changeGetFeesOnTransfer(bool)._status (contracts/Token.sol#675) is not in mixedCase
Parameter DogeCatChat.changePair(address)._pair (contracts/Token.sol#679) is not in mixedCase
Constant DogeCatChat._name (contracts/Token.sol#408) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DogeCatChat._symbol (contracts/Token.sol#409) is not in UPPER_CASE_WITH_UNDERSCORES
Constant DogeCatChat._decimals (contracts/Token.sol#410) is not in UPPER_CASE_WITH_UNDERSCORES
Variable DogeCatChat._totalSupply (contracts/Token.sol#412) is not in mixedCase
Variable DogeCatChat._balances (contracts/Token.sol#414) is not in mixedCase
Variable DogeCatChat._allowances (contracts/Token.sol#415) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in DogeCatChat._transferFrom(address,address,uint256) (contracts/Token.sol#533-551):
        External calls:
        - doContractSwap() (contracts/Token.sol#540)
                - address(marketingWallet).transfer(swappedTokens) (contracts/Token.sol#605)
        State variables written after the call(s):
        - _balances[sender] = _balances[sender] - amount (contracts/Token.sol#544)
        - _balances[recipient] = _balances[recipient] + amountReceived (contracts/Token.sol#547)
        - amountReceived = takeFee(sender,recipient,amount) (contracts/Token.sol#546)
                - _balances[address(this)] = _balances[address(this)] + feeToken (contracts/Token.sol#559)
        Event emitted after the call(s):
        - Transfer(sender,address(this),feeToken) (contracts/Token.sol#560)
                - amountReceived = takeFee(sender,recipient,amount) (contracts/Token.sol#546)
        - Transfer(sender,recipient,amountReceived) (contracts/Token.sol#549)
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/Token.sol#13) is too similar to IUniswapV2Router01.a
ddLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

DogeCatChat._totalSupply (contracts/Token.sol#412) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

DogeCatChat.router (contracts/Token.sol#428) should be immutable
DogeCatChat.swapThreshold (contracts/Token.sol#435) 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/0x539abd7782169bb10309446eee9f2e204d66000bb3d3cf55c25117788bccba34

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

https://testnet.bscscan.com/tx/0xe1e441687455a172571fce8f7da4f4ac5c92ffef454d5eb8bf0fdc4ba2220745

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

https://testnet.bscscan.com/tx/0x3cc8d3fa91ad83e4d28088f150eef8f9d106b6427f7477856700e76248d232ae

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

https://testnet.bscscan.com/tx/0x5935a1138d0817f97b5a5bee41b4c681bce707e1bb6c69d854be9969390269f9

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

https://testnet.bscscan.com/tx/0x00738315363b5b749834704a3d887229d3561b1e3ce354021a01078e811d6e9e

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

https://testnet.bscscan.com/tx/0xade98c1d8bab0f676def5d659bd24fbe7aa0223aae748d3683ad66eee9b560a8

FUNCTIONAL TESTING

7- Transferring when not excluded from fees (get fee on transfer disabled) (0% tax) (passed):

https://testnet.bscscan.com/tx/0xa63f870477a153724e1d9b371f7c6f5a2476a63b4b48edb203931967a8a8f632

8- Transferring when not excluded from fees (get fee on transfer enabled) (0-10% tax) (passed):

https://testnet.bscscan.com/tx/0x0ed9e07b0e286c5fc69a3a99668a964dbc5113f6c0616dc1879f15611e8528f8

9- Internal swap (passed):

- Tax converted to BNB and sent to marketing wallet

https://testnet.bscscan.com/tx/0xade98c1d8bab0f676def5d659bd24fbe7aa0223aae748d3683ad66eee9b560a8

FUNCTIONAL TESTING

Centralization – Trades must be enabled

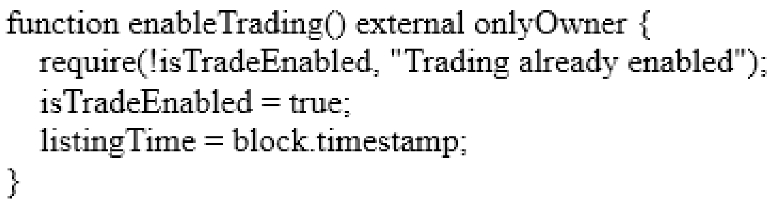
Severity: **High**

function: enableTrading

Status: **Resolved (Contract is owned by safu developer)**

Overview:

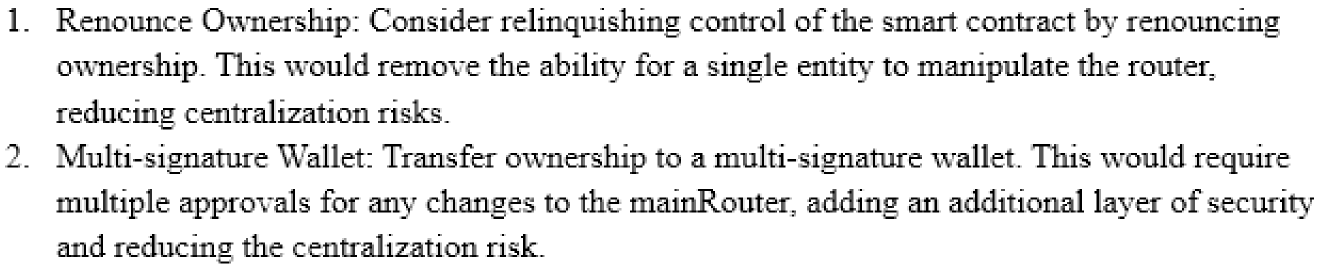
The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.



```
function enableTrading() external onlyOwner {
    require(!isTradeEnabled, "Trading already enabled");
    isTradeEnabled = true;
    listingTime = block.timestamp;
}
```

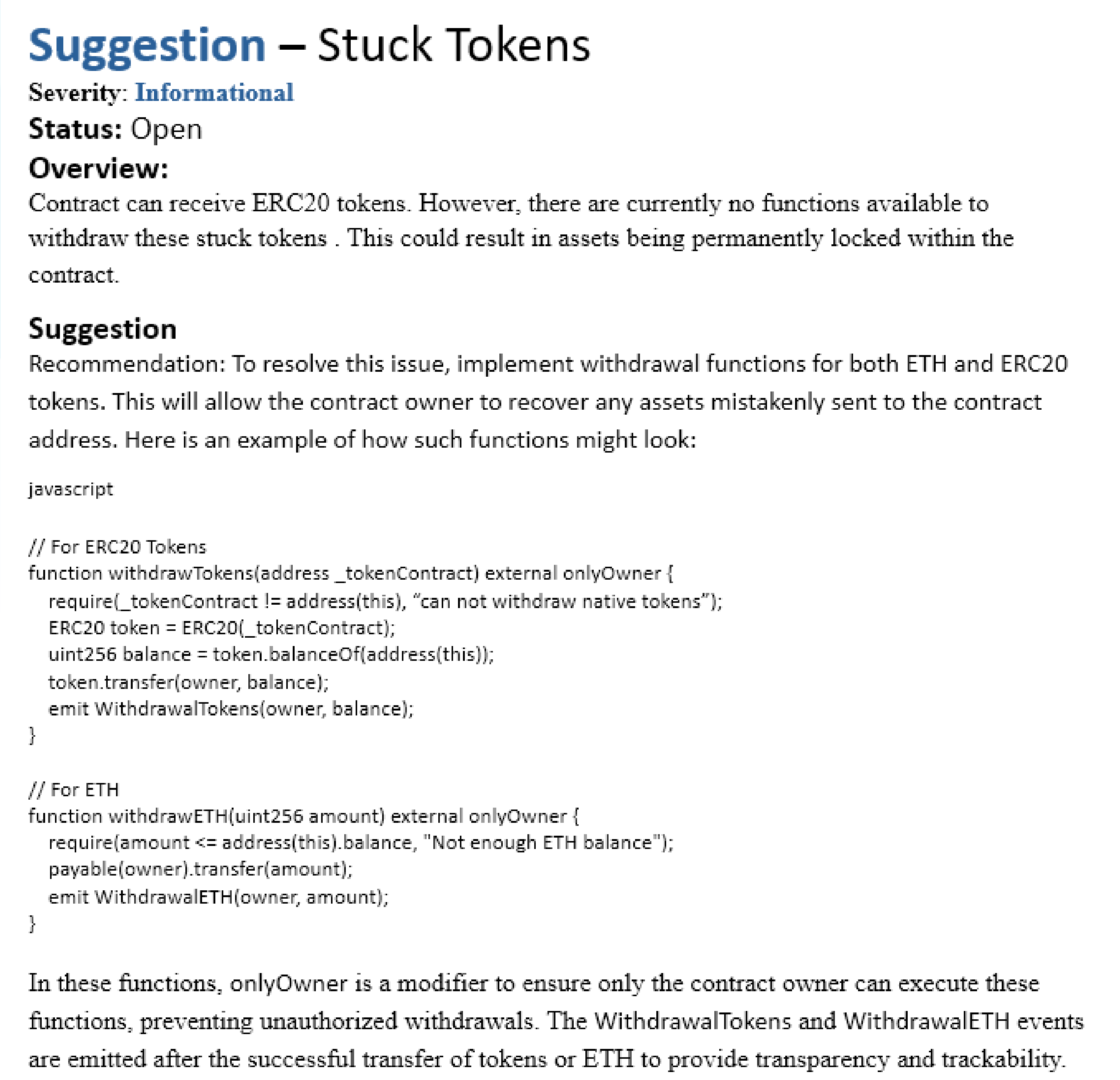
Suggestion

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

FUNCTIONAL TESTING



Suggestion – Stuck Tokens

Severity: **Informational**

Status: Open

Overview:

Contract can receive ERC20 tokens. However, there are currently no functions available to withdraw these stuck tokens . This could result in assets being permanently locked within the contract.

Suggestion

Recommendation: To resolve this issue, implement withdrawal functions for both ETH and ERC20 tokens. This will allow the contract owner to recover any assets mistakenly sent to the contract address. Here is an example of how such functions might look:

javascript

```
// For ERC20 Tokens
function withdrawTokens(address _tokenContract) external onlyOwner {
   require(_tokenContract != address(this), “can not withdraw native tokens”);
   ERC20 token = ERC20(_tokenContract);
   uint256 balance = token.balanceOf(address(this));
   token.transfer(owner, balance);
   emit WithdrawalTokens(owner, balance);
}

// For ETH
function withdrawETH(uint256 amount) external onlyOwner {
   require(amount <= address(this).balance, "Not enough ETH balance");
   payable(owner).transfer(amount);
   emit WithdrawalETH(owner, amount);
}
```

In these functions, onlyOwner is a modifier to ensure only the contract owner can execute these functions, preventing unauthorized withdrawals. The WithdrawalTokens and WithdrawalETH events are emitted after the successful transfer of tokens or ETH to provide transparency and trackability.

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