AuditAce

FROM INCEPTION TO SUCCESS

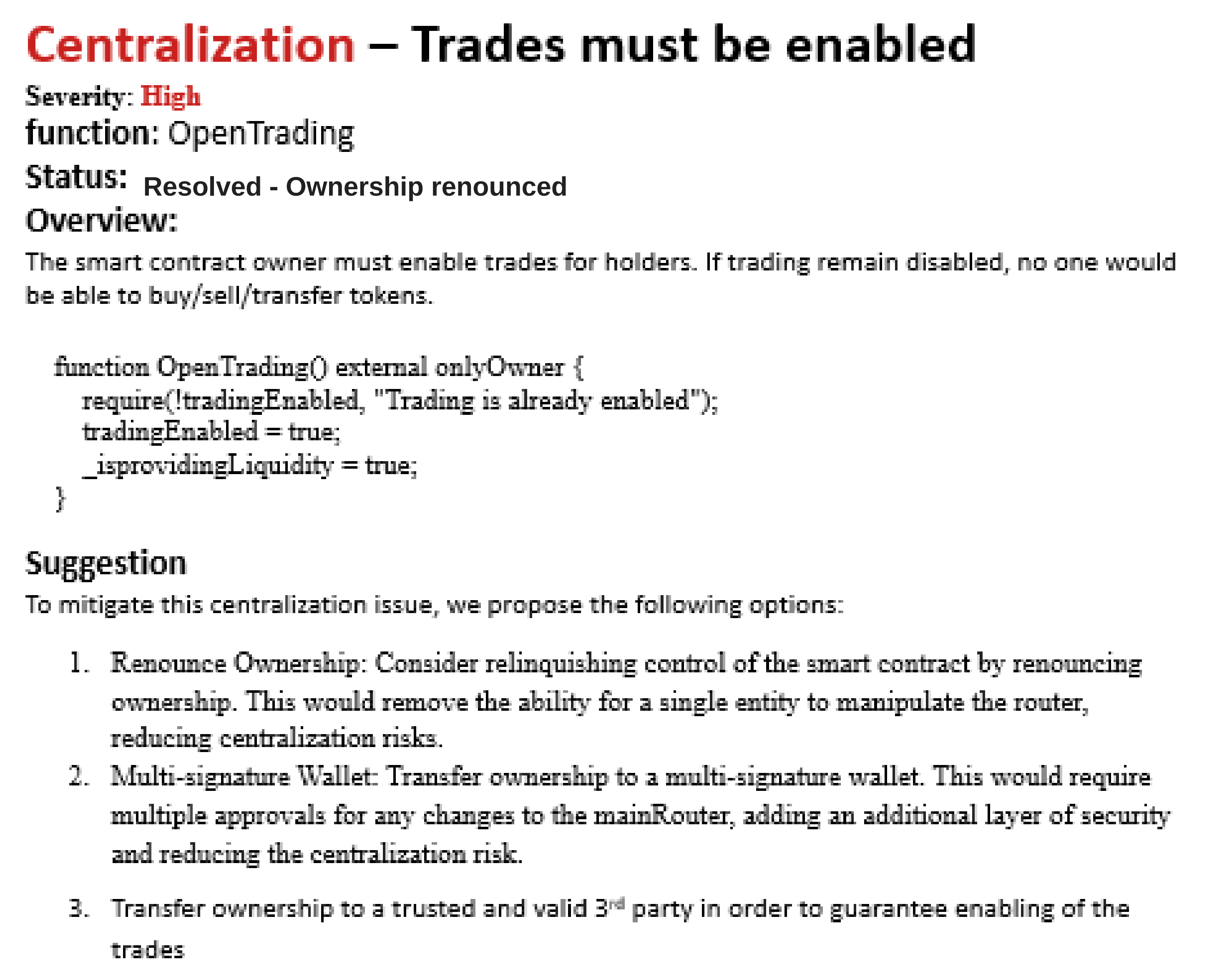
Smart Contract Audit

FOR

Maze

DATED : 27 June 23'

FUNCTIONAL TESTING



Centralization – Trades must be enabled

Severity: High

function: OpenTrading

Status: Resolved - Ownership renounced

Overview:

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

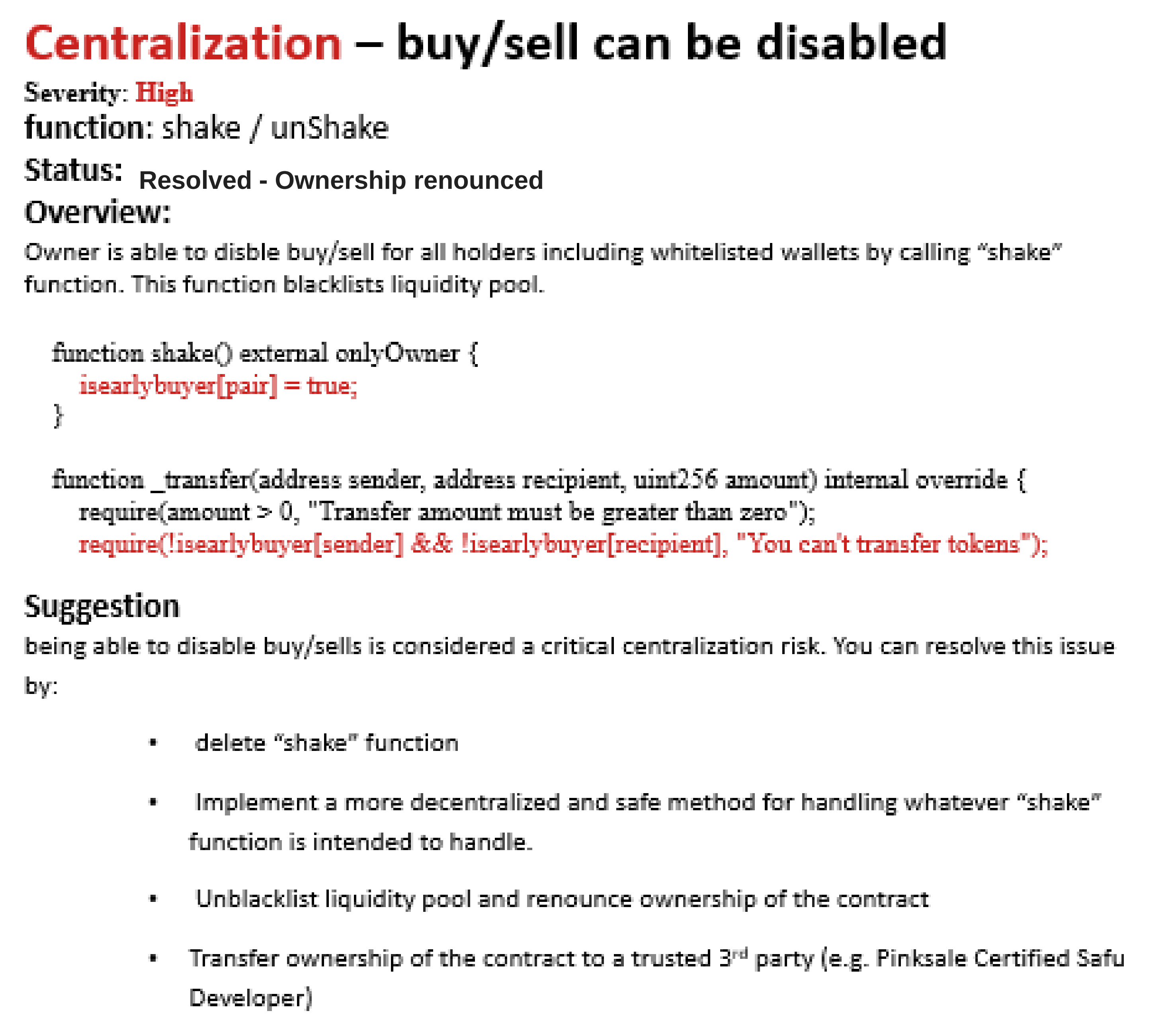
```
function OpenTrading() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    _isprovidingLiquidity = true;
}
```

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

FUNCTIONAL TESTING



Centralization – buy/sell can be disabled

Severity: High

function: shake / unShake

Status: Resolved - Ownership renounced

Overview:

Owner is able to disble buy/sell for all holders including whitelisted wallets by calling "shake" function. This function blacklists liquidity pool.

```
function shake() external onlyOwner {
    isearlybuyer[pair] = true;
}

function _transfer(address sender, address recipient, uint256 amount) internal override {
    require(amount > 0, "Transfer amount must be greater than zero");
    require(!isearlybuyer[sender] && !isearlybuyer[recipient], "You can't transfer tokens");
```

Suggestion

being able to disable buy/sells is considered a critical centralization risk. You can resolve this issue by:

- delete "shake" function
- Implement a more decentralized and safe method for handling whatever "shake" function is intended to handle.
- Unblacklist liquidity pool and renounce ownership of the contract
- Transfer ownership of the contract to a trusted 3rd party (e.g. Pinksale Certified Safu Developer)

AUDIT SUMMARY

Project name – Maze

Date: 27 June, 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	0	2	1	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/0x47bb34cb522647d05057ac0d48dd9965819daeea

Token Information

Token Name : Maze

Token Symbol: $MAZE

Decimals: 9

Token Supply: 100,000,000,000,000

Token Address:
0x78ea2A1D72309c74F2Bb3E4B658FA7309B2A6C0D

Checksum:
87336c8f273d3e3842b2e7ef827ded41952ed023

Owner:
0x0FB1C8F0b7644C86E0aD9f1CBB6cD7a97D02c509
(at time of writing the audit)

Deployer:
0x0FB1C8F0b7644C86E0aD9f1CBB6cD7a97D02c509

TOKEN OVERVIEW

Fees:

Buy Fees: 0-30%

Sell Fees: 0-40%

Transfer Fees: 0-30%

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
- disabling 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

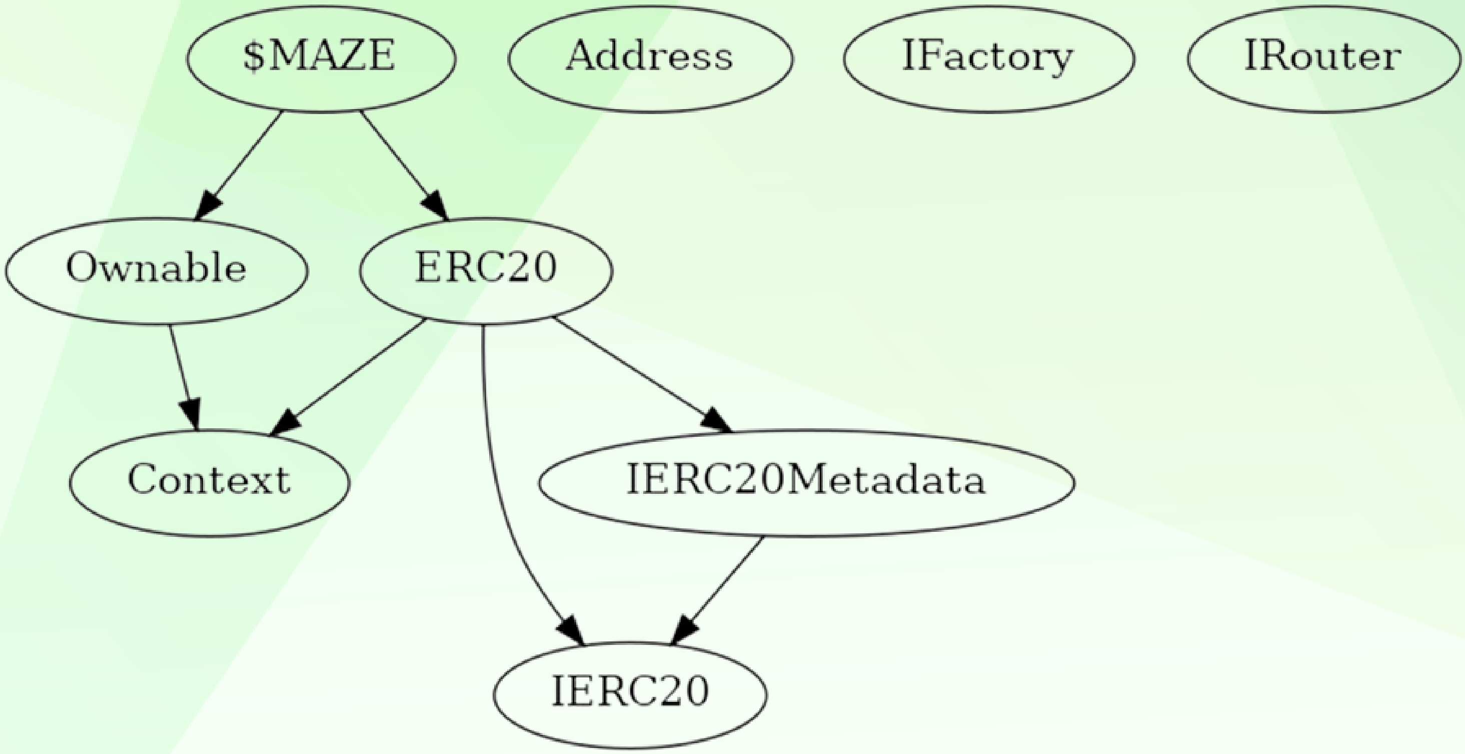
CLASSIFICATION OF RISK

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**	2
◆ **Medium-Risk**	1
◆ **Low-Risk**	1
◆ **Gas Optimization / Suggestions**	0

INHERITANCE TREE



POINTS TO NOTE

- **Owner is able to set buy/transfer tax up to 30% and sell tax up to 40%**
- Owner is not able to blacklist an arbitrary address.
- Owner is able to set max wallet limit. (max wallet limit >= 0.1% of total supply)
- Owner is not able to mint new tokens
- **Owner must enable trades manually**
- **Owner is able to disable trades**

CONTRACT ASSESMENT

```
| Contract |        Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |  **Modifiers**  |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | 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 ❗️ |
||||||
| **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 🔒 | 🛑  | |
| └ | _tokengeneration | Internal 🔒 | 🛑  | |
| └ | _approve | Internal 🔒 | 🛑  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑  | |
||||||
| **Address** | Library |  |||
| └ | sendValue | Internal 🔒 | 🛑  | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO ❗️ |
```

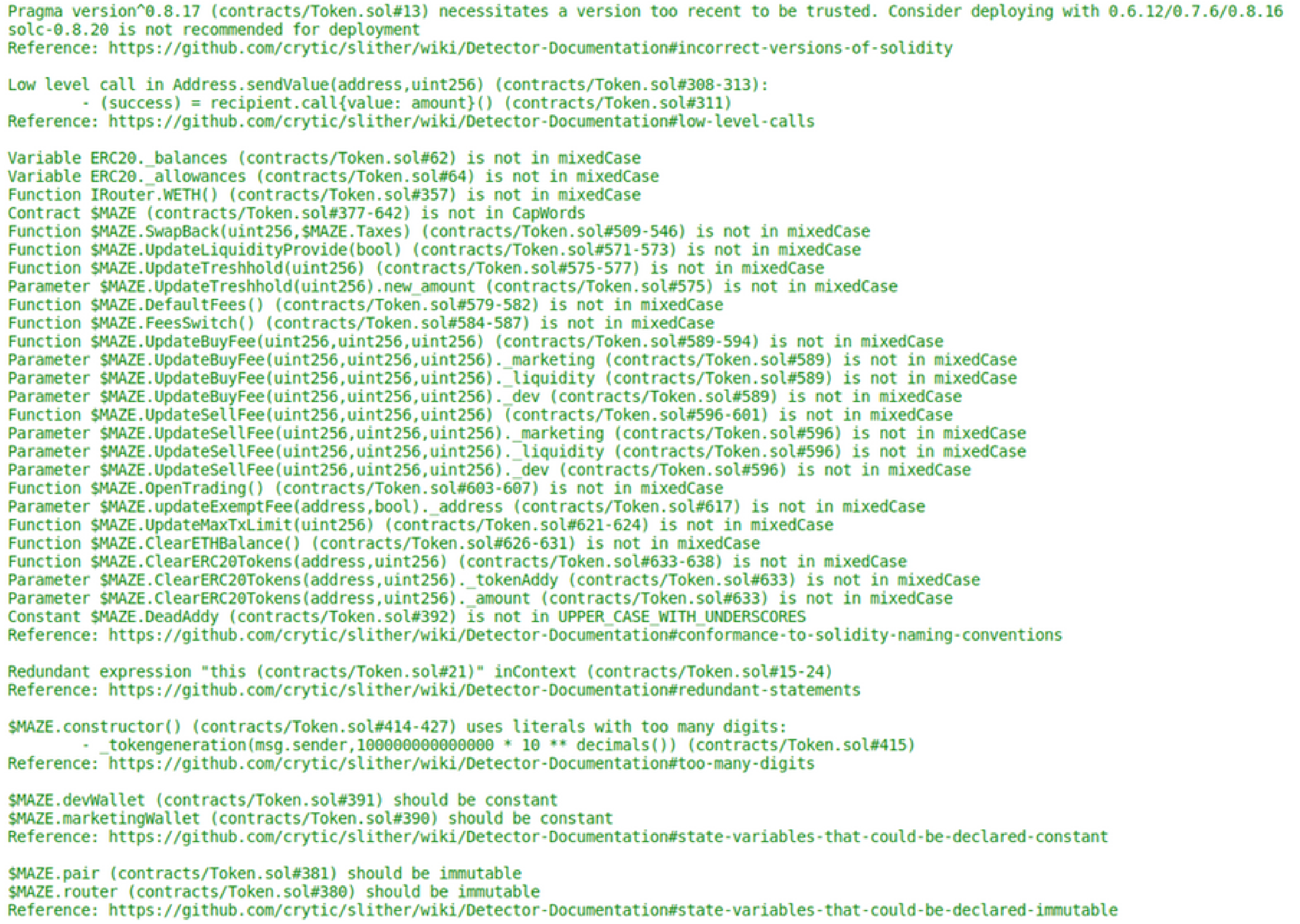
CONTRACT ASSESMENT

```
| └ | owner | Public ❗️ |   |NO ❗️ |
| └ | renounceOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🛑  | onlyOwner |
| └ | _setOwner | Private 🔐 | 🛑  | |
||||||
| **IFactory** | Interface |  |||
| └ | createPair | External ❗️ | 🛑  |NO ❗️ |
||||||
| **IRouter** | Interface |  |||
| └ | factory | External ❗️ |   |NO ❗️ |
| └ | WETH | External ❗️ |   |NO ❗️ |
| └ | addLiquidityETH | External ❗️ | 💵 |NO ❗️ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗️ | 🛑  |NO ❗️ |
||||||
| **$MAZE** | Implementation | ERC20, Ownable |||
| └ | <Constructor> | Public ❗️ | 🛑  | ERC20 |
| └ | approve | Public ❗️ | 🛑  |NO ❗️ |
| └ | transferFrom | Public ❗️ | 🛑  |NO ❗️ |
| └ | increaseAllowance | Public ❗️ | 🛑  |NO ❗️ |
| └ | decreaseAllowance | Public ❗️ | 🛑  |NO ❗️ |
| └ | transfer | Public ❗️ | 🛑  |NO ❗️ |
| └ | _transfer | Internal 🔒 | 🛑  | |
| └ | SwapBack | Private 🔐 | 🛑  | mutexLock |
| └ | swapTokensForETH | Private 🔐 | 🛑  | |
| └ | addLiquidity | Private 🔐 | 🛑  | |
| └ | UpdateLiquidityProvide | External ❗️ | 🛑  | onlyOwner |
| └ | UpdateTreshhold | External ❗️ | 🛑  | onlyOwner |
| └ | DefaultFees | External ❗️ | 🛑  | onlyOwner |
| └ | FeesSwitch | External ❗️ | 🛑  | onlyOwner |
| └ | UpdateBuyFee | External ❗️ | 🛑  | onlyOwner |
| └ | UpdateSellFee | External ❗️ | 🛑  | onlyOwner |
| └ | OpenTrading | External ❗️ | 🛑  | onlyOwner |
| └ | shake | External ❗️ | 🛑  | onlyOwner |
| └ | unShake | External ❗️ | 🛑  | onlyOwner |
| └ | updateExemptFee | External ❗️ | 🛑  | onlyOwner |
| └ | UpdateMaxTxLimit | External ❗️ | 🛑  | onlyOwner |
| └ | ClearETHBalance | External ❗️ | 🛑  |NO ❗️ |
| └ | ClearERC20Tokens | External ❗️ | 🛑  | onlyOwner |
| └ | <Receive Ether> | External ❗️ | 💵 |NO ❗️ |
```

CONTRACT ASSESMENT

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS



```
Pragma version^0.8.17 (contracts/Token.sol#13) 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 Address.sendValue(address,uint256) (contracts/Token.sol#308-313):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#311)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Variable ERC20._balances (contracts/Token.sol#62) is not in mixedCase
Variable ERC20._allowances (contracts/Token.sol#64) is not in mixedCase
Function IRouter.WETH() (contracts/Token.sol#357) is not in mixedCase
Contract $MAZE (contracts/Token.sol#377-642) is not in CapWords
Function $MAZE.SwapBack(uint256,$MAZE.Taxes) (contracts/Token.sol#509-546) is not in mixedCase
Function $MAZE.UpdateLiquidityProvide(bool) (contracts/Token.sol#571-573) is not in mixedCase
Function $MAZE.UpdateTreshhold(uint256) (contracts/Token.sol#575-577) is not in mixedCase
Parameter $MAZE.UpdateTreshhold(uint256).new_amount (contracts/Token.sol#575) is not in mixedCase
Function $MAZE.DefaultFees() (contracts/Token.sol#579-582) is not in mixedCase
Function $MAZE.FeesSwitch() (contracts/Token.sol#584-587) is not in mixedCase
Function $MAZE.UpdateBuyFee(uint256,uint256,uint256) (contracts/Token.sol#589-594) is not in mixedCase
Parameter $MAZE.UpdateBuyFee(uint256,uint256,uint256)._marketing (contracts/Token.sol#589) is not in mixedCase
Parameter $MAZE.UpdateBuyFee(uint256,uint256,uint256)._liquidity (contracts/Token.sol#589) is not in mixedCase
Parameter $MAZE.UpdateBuyFee(uint256,uint256,uint256)._dev (contracts/Token.sol#589) is not in mixedCase
Function $MAZE.UpdateSellFee(uint256,uint256,uint256) (contracts/Token.sol#596-601) is not in mixedCase
Parameter $MAZE.UpdateSellFee(uint256,uint256,uint256)._marketing (contracts/Token.sol#596) is not in mixedCase
Parameter $MAZE.UpdateSellFee(uint256,uint256,uint256)._liquidity (contracts/Token.sol#596) is not in mixedCase
Parameter $MAZE.UpdateSellFee(uint256,uint256,uint256)._dev (contracts/Token.sol#596) is not in mixedCase
Function $MAZE.OpenTrading() (contracts/Token.sol#603-607) is not in mixedCase
Parameter $MAZE.updateExemptFee(address,bool)._address (contracts/Token.sol#617) is not in mixedCase
Function $MAZE.UpdateMaxTxLimit(uint256) (contracts/Token.sol#621-624) is not in mixedCase
Function $MAZE.ClearETHBalance() (contracts/Token.sol#626-631) is not in mixedCase
Function $MAZE.ClearERC20Tokens(address,uint256) (contracts/Token.sol#633-638) is not in mixedCase
Parameter $MAZE.ClearERC20Tokens(address,uint256)._tokenAddy (contracts/Token.sol#633) is not in mixedCase
Parameter $MAZE.ClearERC20Tokens(address,uint256)._amount (contracts/Token.sol#633) is not in mixedCase
Constant $MAZE.DeadAddy (contracts/Token.sol#392) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#21)" inContext (contracts/Token.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

$MAZE.constructor() (contracts/Token.sol#414-427) uses literals with too many digits:
        - _tokengeneration(msg.sender,100000000000000 * 10 ** decimals()) (contracts/Token.sol#415)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

$MAZE.devWallet (contracts/Token.sol#391) should be constant
$MAZE.marketingWallet (contracts/Token.sol#390) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

$MAZE.pair (contracts/Token.sol#381) should be immutable
$MAZE.router (contracts/Token.sol#380) 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

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x0718f9623e5c7fb7d3b111b8344a52060594db615b7dff3b1856fd091b630a8e

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x24d7038b2e81f679f0f60094505a9b07e015156de1cdfa70b5c1c1835bd90f40

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xb2b0699769c7126cb685119f5cd0bec240616c3cb6acc21286839f456287c25c

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0x1ff784cf728c17a0df92a53d5984e3a65fb817e54d75f045a8db320709d20d71

5- Buying when not excluded from fees (0-10% tax) (passed):
https://testnet.bscscan.com/tx/0x4ee3e58b64c97969810fc1d4807d63c6dfb58afea211e65541c305f34b716571

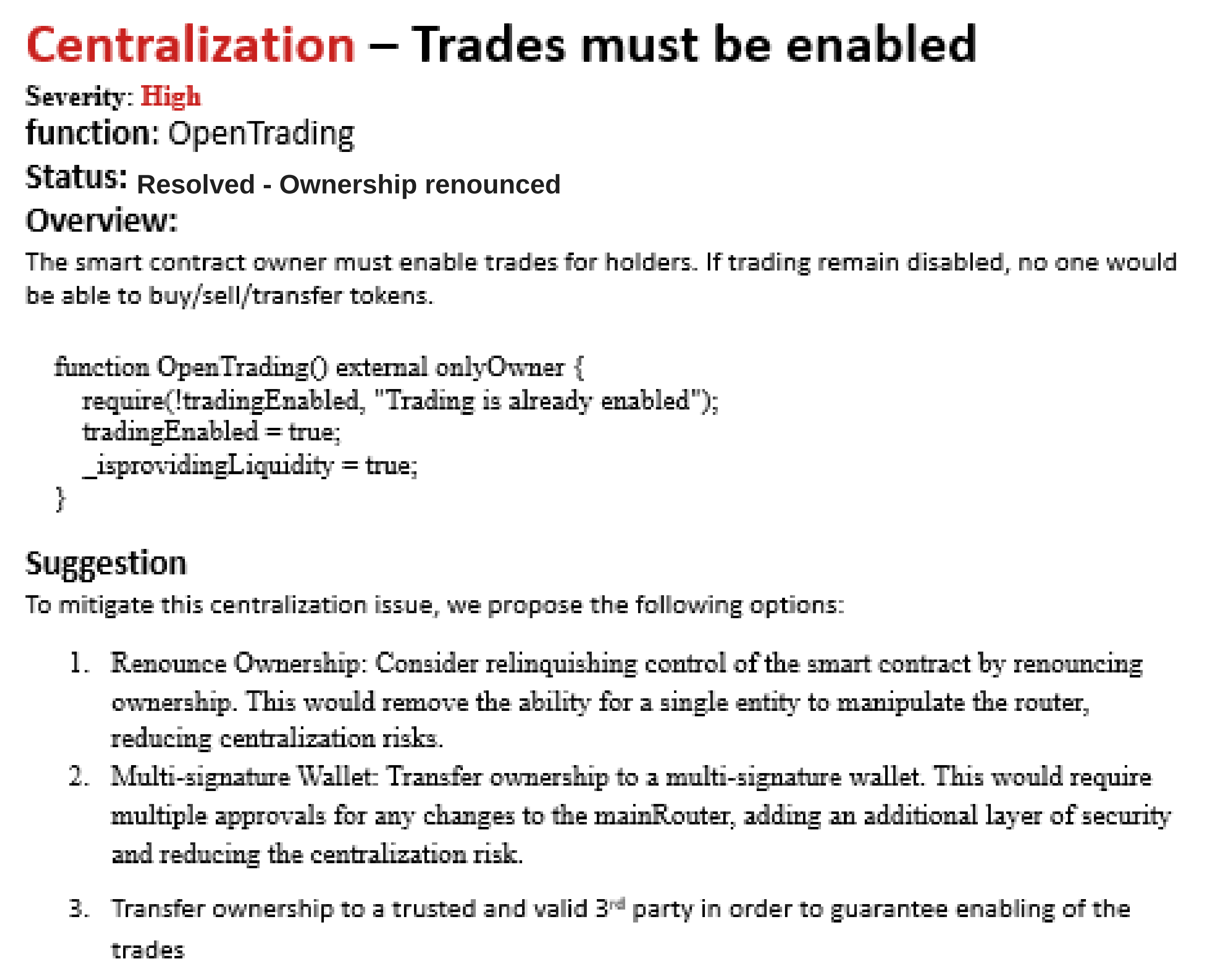
6- Selling when not excluded from fees (0-10% tax) (passed):
https://testnet.bscscan.com/tx/0xf361884612162e4deed9596f9f6f3c3d02c25caa63ebb04032f6380bd6cbf4e0

FUNCTIONAL TESTING

77- Transferring when not excluded from fees (0-10% tax) **(passed):**
https://testnet.bscscan.com/tx/0x64f07cf89e575f201f91e649ef45d76f4e86809ea383a27035e55893d2029173

8- Internal swap (auto-liquidty, marketing and dev wallet received ETH) **(passed):**
https://testnet.bscscan.com/tx/0xd8a3ccb2089808d7eda841cb23a4ba0b240e23fd125bef9e208ad96692b7e07e

FUNCTIONAL TESTING



Centralization – Trades must be enabled

Severity: High

function: OpenTrading

Status: Resolved - Ownership renounced

Overview:

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

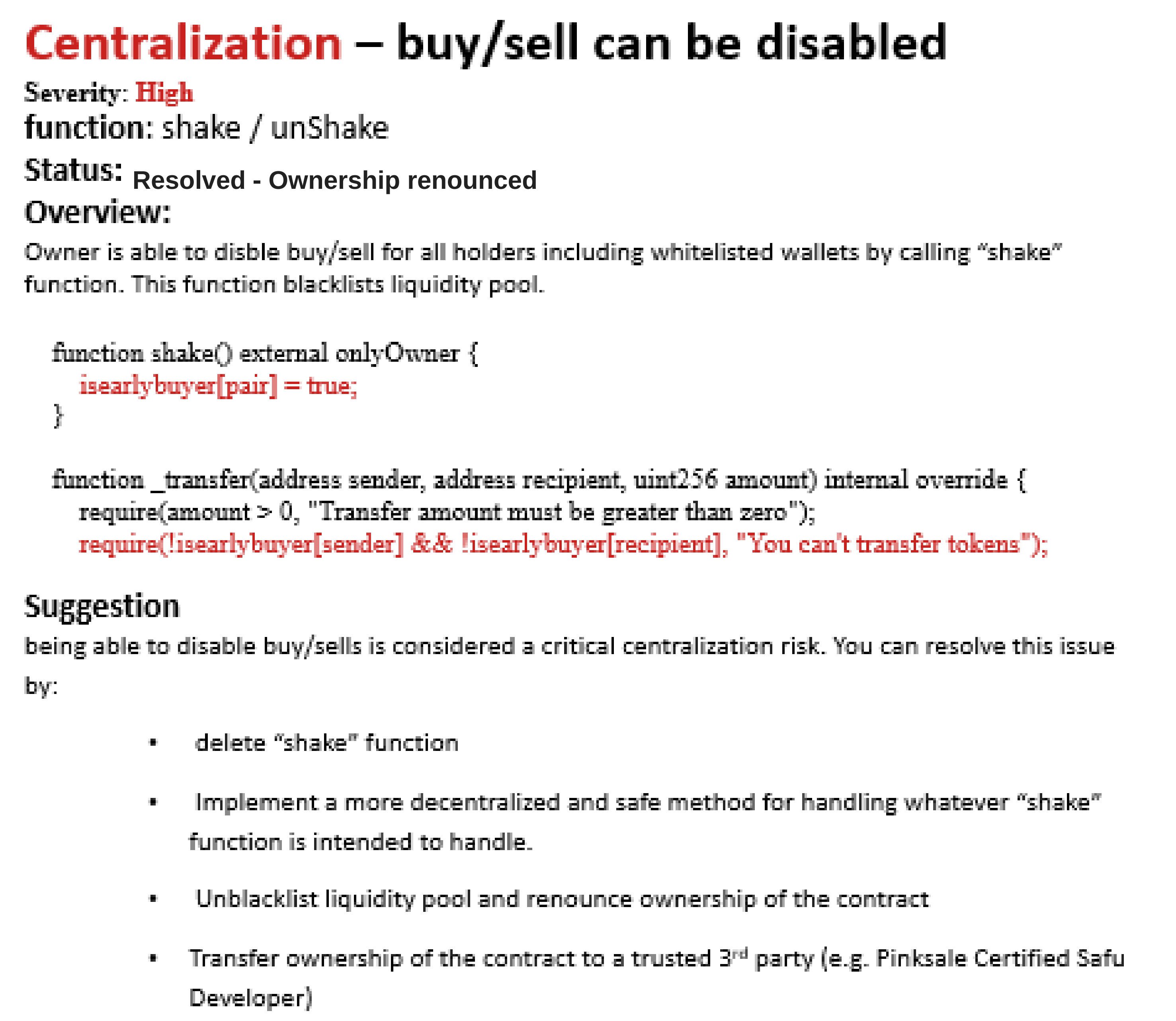
```
function OpenTrading() external onlyOwner {
    require(!tradingEnabled, "Trading is already enabled");
    tradingEnabled = true;
    _isprovidingLiquidity = true;
}
```

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

FUNCTIONAL TESTING



Centralization – buy/sell can be disabled

Severity: **High**

function: shake / unShake

Status: Resolved - Ownership renounced

Overview:

Owner is able to disble buy/sell for all holders including whitelisted wallets by calling "shake" function. This function blacklists liquidity pool.

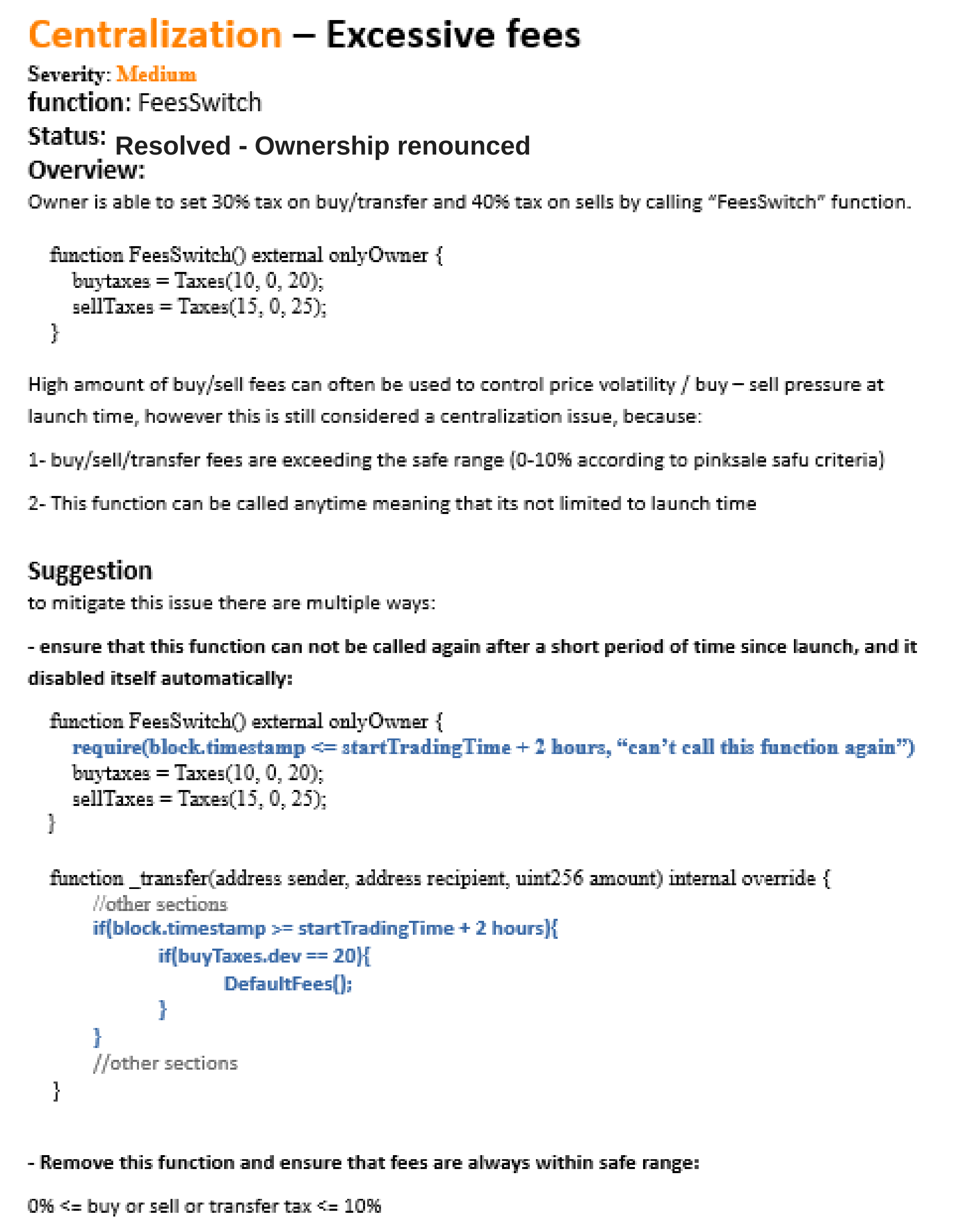
```
function shake() external onlyOwner {
    isearlybuyer[pair] = true;
}

function _transfer(address sender, address recipient, uint256 amount) internal override {
    require(amount > 0, "Transfer amount must be greater than zero");
    require(!isearlybuyer[sender] && !isearlybuyer[recipient], "You can't transfer tokens");
```

Suggestion

being able to disable buy/sells is considered a critical centralization risk. You can resolve this issue by:

- delete "shake" function
- Implement a more decentralized and safe method for handling whatever "shake" function is intended to handle.
- Unblacklist liquidity pool and renounce ownership of the contract
- Transfer ownership of the contract to a trusted 3rd party (e.g. Pinksale Certified Safu Developer)

FUNCTIONAL TESTING



Centralization – Excessive fees

Severity: Medium

function: FeesSwitch

Status: Resolved - Ownership renounced

Overview:

Owner is able to set 30% tax on buy/transfer and 40% tax on sells by calling “FeesSwitch” function.

```
function FeesSwitch() external onlyOwner {
    buytaxes = Taxes(10, 0, 20);
    sellTaxes = Taxes(15, 0, 25);
}
```

High amount of buy/sell fees can often be used to control price volatility / buy – sell pressure at launch time, however this is still considered a centralization issue, because:

1- buy/sell/transfer fees are exceeding the safe range (0-10% according to pinksale safu criteria)

2- This function can be called anytime meaning that its not limited to launch time

Suggestion

to mitigate this issue there are multiple ways:

- ensure that this function can not be called again after a short period of time since launch, and it disabled itself automatically:

```
function FeesSwitch() external onlyOwner {
    require(block.timestamp <= startTradingTime + 2 hours, “can’t call this function again”)
    buytaxes = Taxes(10, 0, 20);
    sellTaxes = Taxes(15, 0, 25);
}

function _transfer(address sender, address recipient, uint256 amount) internal override {
        //other sections
        if(block.timestamp >= startTradingTime + 2 hours){
                if(buyTaxes.dev == 20){
                        DefaultFees();
                }
        }
        //other sections
}
```

- Remove this function and ensure that fees are always within safe range:

0% <= buy or sell or transfer tax <= 10%

FUNCTIONAL TESTING

Centralization – Max wallet limit

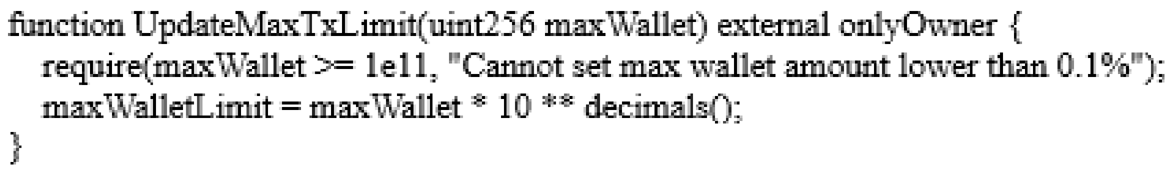
Severity: Low

function: updateMaxTxLimit

Status: Resolved - Ownership renounced

Overview:

Owner is able to set a max wallet limit meaning that wallets (except whitelisted ones) wont be able to hold more than this maximum limit.
Owner is able to adjust this maximum limit in range of 0.1% - 100% of total supply.



```
function UpdateMaxTxLimit(uint256 maxWallet) external onlyOwner {
    require(maxWallet >= 1e11, "Cannot set max wallet amount lower than 0.1%");
    maxWalletLimit = maxWallet * 10 ** decimals();
}
```

Suggestion

Minimum value of maximim wallet is in accordance with pinksale safu criteria, however its still considered a centralization issue.

https://docs.pinksale.finance/important/safu-contract

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