sollidify

Project: NodeSynapse

0x30672ae2680c319ec1028b69670a4a786baa0f35

11/03/2024

AUDIT REPORT

SAFETY SCORE: 80

1 - Arbitrary Jump/Storage Write

Result: Pass

2 - Centralization of Control

Result: High

Details: The contract contains functions that allow the owner to exert a high level of control over the contract, which can be a risk for

decentralization. The `removeLimits` function allows the owner to remove the maximum transaction amount and wallet size limits, and the

`removeUnclogLimits` function allows the owner to disable transfer delays and sell limits. Additionally, the owner can manually swap tokens

and send ETH to a tax wallet using the `manualSwap` function.

Code:

```
function removeLimits() external onlyOwner {

_maxTxAmount = _tTotal;

_maxWalletSize=_tTotal;

emit MaxTxAmountUpdated(_tTotal);

}

function removeUnclogLimits() external {

require(_msgSender()==_taxWallet);

transferDelayEnabled = false;

caSellLimit = false;
```

```
}

function manualSwap() external {

require(_msgSender()==_taxWallet);

uint256 tokenBalance=balanceOf(address(this));

if(tokenBalance>0){

swapTokensForEth(tokenBalance);

}

uint256 ethBalance=address(this).balance;

if(ethBalance>0){

sendETHToFee(ethBalance);

}

}
```

Correction:

```
// To mitigate centralization risks, these functions should be removed or their capabilities should be

limited.
```

3 - Compiler Issues

Result: Pass

4 - Delegate Call to Untrusted Contract

Result: Pass

5 - Dependence on Predictable Variables

Result: Medium

Details: The contract uses `block.number` as a mechanism to enforce transfer delays, which can be predicted by miners and manipulated to

some extent.

Code:

```
if (transferDelayEnabled) {

if (to != address(uniswapV2Router) && to !=
address(uniswapV2Pair)) {

require(

_holderLastTransferTimestamp[tx.origin] <

block.number,

"_transfer:: Transfer Delay enabled. Only one purchase per
block allowed."

);

_holderLastTransferTimestamp[tx.origin] = block.number;

}

}
```

Correction:

```
// Replace block.number with a more unpredictable variable,
such as block.timestamp, or remove the

dependency.
```

6 - Ether/Token Theft

Result: Pass

7 - Flash Loans

Result: Pass

8 - Front Running

Result: Medium

Details: The contract is susceptible to front-running attacks because it does not implement any mechanism to prevent them, such as using a

commit-reveal scheme or similar.

Code:

```
// No specific code snippet provided as the entire logic of handling transactions would need to be

adjusted to implement front-running protection.
```

Correction:

```
// Implement a commit-reveal scheme or other anti-front-running measures.
```

9 - Improper Events

Result: Pass

10 - Improper Authorization Scheme

Result: Pass

11 - Integer Over/Underflow

Result: Pass

Details: The contract uses SafeMath library for all arithmetic operations, which protects against overflows and underflows.

12 - Logical Issues

Result: Medium

Details: The contract has a logical issue where the tax amount is calculated and deducted even for transfers between non-excluded accounts

and the owner, which may not be the intended behavior.

Code:

```
if(taxAmount>0){
_balances[address(this)]=_balances[address(this)].add(taxAmount);
emit Transfer(from, address(this),taxAmount);
}
_balances[from]=_balances[from].sub(amount);
_balances[to]=_balances[to].add(amount.sub(taxAmount));
emit Transfer(from, to, amount.sub(taxAmount));
```

Correction:

```
// Ensure that tax is only applied to the intended
transactions and not to owner-involved transfers if
that is the desired logic.
```

13 - Oracle Issues

Result: Pass

14 - Outdated Compiler Version

Result: Pass

15 - Race Conditions

Result: Pass

16 - Reentrancy

Result: Pass

17 - Signature Issues

Result: Pass

18 - Sybil Attack

Result: Pass

19 - Unbounded Loops

Result: Pass

20 - Unused Code

Result: Low

Details: The contract contains unused code, such as the `isBot` function, which checks if an address is marked as a bot but is never used

within the contract.

Code:

```
function isBot(address a) public view returns (bool){

return bots[a];

}
```

Correction:

```
// Remove the isBot function if it is not used, or implement its usage within the contract logic.
```