MythX

REPORT 60908DEEDF176A0019DFAD89

Created	Mon May 03 2021 23:57:34 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60908aaf67db365926bad1be

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
66284081-1202-42fb-b8da-4cbc3a5bc672	contracts/Zephyrus.sol	30

Analysis 66284081-1202-42fb-b8da-4cbc3a5bc672

MythX

Started	Mon May 03 2021 23:57:43 GMT+0000 (Coordinated Universal Time)
Finished	Tue May 04 2021 00:13:11 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Remythx
Main Source File	Contracts/Zephyrus.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	27	3

ISSUES

MEDIUM Function could be marked as external.

SWC-000

The function definition of "renounceOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```
* thereby removing any functionality that is only available to the owner.
*/
function renounceOwnership() public virtual onlyOwner {
emit OwnershipTransferred(_owner, address(0));
_owner = address(0);
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```
* Can only be called by the current owner.
*/
function transferOwnership(address newOwner) public virtual onlyOwner {
require(newOwner != address(0), "Ownable: new owner is the zero address");
emit OwnershipTransferred(_owner, newOwner);
_owner = newOwner;
}

function geUnlockTime() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "geUnlockTime" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```
}

function geUnlockTime() public view returns (uint256) {
return _lockTime;
}

//Locks the contract for owner for the amount of time provided
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "lock" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```

//Locks the contract for owner for the amount of time provided
function lock(uint256 time) public virtual onlyOwner {
_previousOwner = _owner;
_owner = address(0);
_lockTime = now + time;
emit OwnershipTransferred(_owner, address(0));
}

//Unlocks the contract for owner when _lockTime is exceeds
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "unlock" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```

//Unlocks the contract for owner when _lockTime is exceeds
function unlock() public virtual {
require(_previousOwner == msg.sender, "You don't have permission to unlock");
require(now > _lockTime , "Contract is locked until 7 days");
emit OwnershipTransferred(_owner, _previousOwner);
_owner = _previousOwner;
}
}

```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "name" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function name() public view returns (string memory) {
return _name;
}

function symbol() public view returns (string memory) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function symbol() public view returns (string memory) {
return _symbol;
}

function decimals() public view returns (uint8) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function decimals() public view returns (uint8) {
return _decimals;
}

function totalSupply() public view override returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function totalSupply() public view override returns (uint256) {
return _tTotal;
}

function balanceOf(address account) public view override returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function transfer(address recipient, uint256 amount) public override returns (bool) {
_transfer(_msgSender(), recipient, amount);
return true;
}

function allowance(address owner, address spender) public view override returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "allowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function allowance(address owner, address spender) public view override returns (uint256) {
return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) public override returns (bool) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function approve(address spender, uint256 amount) public override returns (bool) {
_approve(_msgSender(), spender, amount);
return true;
}

function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
_transfer(sender, recipient, amount);
_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
return true;
}

function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
806 }
807
808 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
809 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
810 return true;
811 }
812
813 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
811 }
812
813 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
814 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
815 return true;
816 }
817
818 function isExcludedFromReward(address account) public view returns (bool) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isExcludedFromReward" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
816 }
817
818 function isExcludedFromReward(address account) public view returns (bool) {
819 return _isExcluded[account];
820 }
821
822 function totalFees() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalFees" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```
}

function totalFees() public view returns (uint256) {
return _tFeeTotal;
}

function changeExludedRetail(address account, bool status) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "changeExludedRetail" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```
}

function changeExludedRetail(address account, bool status) public onlyOwner {
_isExcludedFromFee[account] = status;
}

function deliver(uint256 tAmount) public {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "deliver" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/Zephyrus.sol

Locations

```
}

function deliver(uint256 tAmount) public {
address sender = _msgSender();
require(!_isExcluded[sender], "Excluded addresses cannot call this function");
(uint256 rAmount,,,,,,) = _getValues(tAmount);
_rOwned[sender] = _rOwned[sender].sub(rAmount);
_rTotal = _rTotal.sub(rAmount);
_tFeeTotal = _tFeeTotal.add(tAmount);
}

function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "reflectionFromToken" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {
require(tAmount <= _tTotal, "Amount must be less than supply");
if (!deductTransferFee) {
(uint256 rAmount,,,,,,) = _getValues(tAmount);
return rAmount;
} else {
(,uint256 rTransferAmount,,,,,) = _getValues(tAmount);
return rTransferAmount;
}
}

function tokenFromReflection(uint256 rAmount) public view returns(uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "excludeFromReward" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function excludeFromReward(address account) public onlyOwner() {
// require(account != 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D, 'We can not exclude Uniswap router.');
require(!_isExcluded[account], "Account is already excluded");
if(_rOwned[account] > 0) {
_tOwned[account] = tokenFromReflection(_rOwned[account]);
}
_isExcluded[account] = true;
_excluded.push(account);
}

function includeInReward(address account) external onlyOwner() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "excludeFromFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
888 }
889
890 function excludeFromFee(address account) public onlyOwner {
891 _isExcludedFromFee[account] = true;
892 }
893
894 function includeInFee(address account) public onlyOwner {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "includeInFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
892 }
893
894 function includeInFee(address account) public onlyOwner {
895 _isExcludedFromFee[account] = false;
896 }
897
898 function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setSwapAndLiquifyEnabled" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
923 }
924
925 function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
926 swapAndLiquifyEnabled = _enabled;
927 emit SwapAndLiquifyEnabledUpdated(_enabled);
928 }
929
930 //to recieve ETH from uniswapV2Router when swaping
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isExcludedFromFee" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`contracts/Zephyrus.sol`

Locations

```
}

function isExcludedFromFee(address account) public view returns(bool) {
return _isExcludedFromFee[account];
}

function _approve(address owner, address spender, uint256 amount) private {
```

MEDIUM

SWC-107

Read of persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/Zephyrus.sol`

Locations

```

//exclude owner and this contract from fee
_isExcludedFromFee[owner()] = true;
_isExcludedFromFee[address(this)] = true;

```

MEDIUM

SWC-107

Write to persistent state following external call

The contract account state is accessed after an external call to a user defined address. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`contracts/Zephyrus.sol`

Locations

```
//exclude owner and this contract from fee
_isExcludedFromFee[owner()] = true;
_isExcludedFromFee[address(this)] = true;

emit Transfer(address(0), _msgSender(), _tTotal);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.12"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/Zephyrus.sol`

Locations

```
*/

pragma solidity ^0.6.12;
// SPDX-License-Identifier: Unlicensed
interface IERC20 {
```

LOW

SWC-107

A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

`contracts/Zephyrus.sol`

Locations

```
// 0x10ED43C718714eb63d5aA57B78B54704E256024E bsc pancake router v2
// Create a uniswap pair for this new token
uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
.createPair(address(this), _uniswapV2Router.WETH());

// set the rest of the contract variables
```

LOW

SWC-113

Multiple calls are executed in the same transaction.

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

Source file

`contracts/Zephyrus.sol`

Locations

```
// Create a uniswap pair for this new token
uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
.createPair(address(this), _uniswapV2Router.WETH());

// set the rest of the contract variables
```