MythX

REPORT 60F852EE0A209400195332A2

Created	Wed Jul 21 2021 17:01:34 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60b6a744a6e1845c77c6e3dc

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
55ceb8ac-801f-47c4-968d-6c759c6d9ffd	/contracts-v1/theflashtoken.sol	36

Analysis `55ceb8ac-801f-47c4-968d-6c759c6d9ffd`

MythX

Started	Wed Jul 21 2021 17:01:41 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jul 21 2021 17:17:37 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Mythx-Vscode-Extension
Main Source File	`/Contracts-V1/Theflashtoken.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	21	15

ISSUES

MEDIUM

SWC-000

Function could be marked as external.

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-v1/theflashtoken.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-v1/theflashtoken.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;
}
}

```

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-v1/theflashtoken.sol

Locations

```
* @dev Returns the token decimals.
*/
function decimals() public override view returns (uint8) {
return _decimals;
}

/**
```

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-v1/theflashtoken.sol

Locations

```
* @dev Returns the token symbol.
*/
function symbol() public override view returns (string memory) {
return _symbol;
}

/**
```

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-v1/theflashtoken.sol

Locations

```
* - the caller must have a balance of at least `amount`.
*/
function transfer(address recipient, uint256 amount) public override returns (bool) {
_transfer(_msgSender(), recipient, amount);
return true;
}

/**
```

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-v1/theflashtoken.sol

Locations

```
* @dev See {BEP20-allowance}.
*/
function allowance(address owner, address spender) public override view returns (uint256) {
return _allowances[owner][spender];
}

/**
```

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-v1/theflashtoken.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function approve(address spender, uint256 amount) public override returns (bool) {
_approve(_msgSender(), spender, amount);
return true;
}

/**
```

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-v1/theflashtoken.sol

Locations

```
* `amount`.
*/
function transferFrom(
address sender,
address recipient,
uint256 amount
) public override returns (bool) {
_transfer(sender, recipient, amount);
_approve(
sender,
_msgSender(),
_allowances[sender][_msgSender()].sub(amount, "BEP20: transfer amount exceeds allowance")
);
return true;
}

/**
```

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-v1/theflashtoken.sol

Locations

```
* - `spender` cannot be the zero address.
*/
function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
_approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
return true;
}

/**
```

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-v1/theflashtoken.sol`

Locations

```
* `subtractedValue`.
*/
function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
_approve(
_msgSender(),
spender,
_allowances[_msgSender()][spender].sub(subtractedValue, "BEP20: decreased allowance below zero")
);
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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-v1/theflashtoken.sol`

Locations

```
* - `msg.sender` must be the token owner
*/
function mint(uint256 amount) public onlyOwner returns (bool) {
_mint(_msgSender(), amount);
return true;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" 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-v1/theflashtoken.sol`

Locations

```

/// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
function mint(address _to, uint256 _amount) public onlyOwner {
_mint(_to, _amount);
_moveDelegates(address(0), _delegates[_to], _amount);
}

/// @dev overrides transfer function to meet tokenomics of TFLASH
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "isExcludedFromAntiWhale" 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-v1/theflashtoken.sol

Locations

```
* @dev Returns the address is excluded from antiWhale or not.
*/
function isExcludedFromAntiWhale(address _account) public view returns (bool) {
return _excludedFromAntiWhale[_account];
}

// To receive BNB from flashSwapRouter when swapping
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateTransferTaxRate" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateTransferTaxRate(uint16 _transferTaxRate) public onlyOperator {
require(_transferTaxRate <= MAXIMUM_TRANSFER_TAX_RATE, "TFLASH::updateTransferTaxRate: Transfer tax rate must not exceed the maximum rate.");
emit TransferTaxRateUpdated(msg.sender, transferTaxRate, _transferTaxRate);
transferTaxRate = _transferTaxRate;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateBurnRate" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateBurnRate(uint16 _burnRate) public onlyOperator {
require(_burnRate <= 100, "TFLASH::updateBurnRate: Burn rate must not exceed the maximum rate.");
emit BurnRateUpdated(msg.sender, burnRate, _burnRate);
burnRate = _burnRate;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateMaxTransferAmountRate" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateMaxTransferAmountRate(uint16 _maxTransferAmountRate) public onlyOperator {
require(_maxTransferAmountRate <= 10000, "TFLASH::updateMaxTransferAmountRate: Max transfer amount rate must not exceed the maximum rate.");
emit MaxTransferAmountRateUpdated(msg.sender, maxTransferAmountRate, _maxTransferAmountRate);
maxTransferAmountRate = _maxTransferAmountRate;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateMinAmountToLiquify" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateMinAmountToLiquify(uint256 _minAmount) public onlyOperator {
emit MinAmountToLiquifyUpdated(msg.sender, minAmountToLiquify, _minAmount);
minAmountToLiquify = _minAmount;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setExcludedFromAntiWhale" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function setExcludedFromAntiWhale(address _account, bool _excluded) public onlyOperator {
_excludedFromAntiWhale[_account] = _excluded;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateSwapAndLiquifyEnabled" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateSwapAndLiquifyEnabled(bool _enabled) public onlyOperator {
emit SwapAndLiquifyEnabledUpdated(msg.sender, _enabled);
swapAndLiquifyEnabled = _enabled;
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateflashSwapRouter" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateflashSwapRouter(address _router) public onlyOperator {
flashSwapRouter = IUniswapV2Router02(_router);
flashSwapPair = IUniswapV2Factory(flashSwapRouter.factory()).getPair(address(this), flashSwapRouter.WETH());
require(flashSwapPair != address(0), "TFLASH::updateflashSwapRouter: Invalid pair address.");
emit flashSwapRouterUpdated(msg.sender, address(flashSwapRouter), flashSwapPair);
}

/**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferOperator" 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-v1/theflashtoken.sol

Locations

```
* Can only be called by the current operator.
*/
function transferOperator(address newOperator) public onlyOperator {
require(newOperator != address(0), "TFLASH::transferOperator: new operator is the zero address");
emit OperatorTransferred(_operator, newOperator);
_operator = newOperator;
}

// Copied and modified from YAM code:
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.5.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: @uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol

pragma solidity >=0.5.0;

interface IUniswapV2Factory {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.5.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: @uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol

pragma solidity >=0.5.0;

interface IUniswapV2Pair {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.2"". 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-v1/theflashtoken.sol`

Locations

```
// File: @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol

pragma solidity >=0.6.2;

interface IUniswapV2Router01 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.2"". 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-v1/theflashtoken.sol`

Locations

```
// File: @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol

pragma solidity >=0.6.2;


```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.2<0.8.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: @openzeppelin/contracts/utils/Address.sol

pragma solidity >=0.6.2 <0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<0.8.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: @openzeppelin/contracts/math/SafeMath.sol

pragma solidity >=0.6.0 <0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.4.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: contracts/libs/IBEP20.sol

pragma solidity >=0.4.0;

interface IBEP20 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<0.8.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: @openzeppelin/contracts/utils/Context.sol

pragma solidity >=0.6.0 <0.8.0;

/*
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.6.0<0.8.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: @openzeppelin/contracts/access/Ownable.sol

pragma solidity >=0.6.0 <0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is "">=0.4.0"". 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-v1/theflashtoken.sol`

Locations

```
// File: contracts/libs/BEP20.sol

pragma solidity >=0.4.0;


```

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-v1/theflashtoken.sol`

Locations

```
function updateflashSwapRouter(address _router) public onlyOperator {
flashSwapRouter = IUniswapV2Router02(_router);
flashSwapPair = IUniswapV2Factory(flashSwapRouter.factory()).getPair(address(this), flashSwapRouter.WETH());
require(flashSwapPair != address(0), "TFLASH::updateflashSwapRouter: Invalid pair address.");
emit flashSwapRouterUpdated(msg.sender, address(flashSwapRouter), flashSwapPair);
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/contracts-v1/theflashtoken.sol`

Locations

```
require(signatory != address(0), "TFLASH::delegateBySig: invalid signature");
require(nonce == nonces[signatory]++, "TFLASH::delegateBySig: invalid nonce");
require(now <= expiry, "TFLASH::delegateBySig: signature expired");
return _delegate(signatory, delegatee);
}
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/contracts-v1/theflashtoken.sol`

Locations

```
returns (uint256)
{
require(blockNumber < block.number, "TFLASH::getPriorVotes: not yet determined");

uint32 nCheckpoints = numCheckpoints[account];
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/contracts-v1/theflashtoken.sol`

Locations

```
internal
{
uint32 blockNumber = safe32(block.number, "TFLASH::_writeCheckpoint: block number exceeds 32 bits");

if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
```

LOW

SWC-120

A control flow decision is made based on The block.number environment variable.

The block.number environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/contracts-v1/theflashtoken.sol`

Locations

```
returns (uint256)
{
require(blockNumber < block.number, "TFLASH::getPriorVotes: not yet determined");

uint32 nCheckpoints = numCheckpoints[account];
```