MythX

REPORT 60C0BB5B72369300185C706C

Created	Wed Jun 09 2021 13:00:11 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60a253128bfa12714ff2900a

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
15956e6b-599f-4815-acf6-3693bb2fc7d4	contracts/SiberianToken.sol	33

Analysis 15956e6b-599f-4815-acf6-3693bb2fc7d4

MythX

Started	Wed Jun 09 2021 13:00:17 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jun 09 2021 13:15:19 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Remythx
Main Source File	Contracts/SiberianToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	21	12

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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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 SIBERIAN
```

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/SiberianToken.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 siberianSwapRouter 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/SiberianToken.sol`

Locations

```
* Can only be called by the current operator.
*/
function updateTransferTaxRate(uint16 _transferTaxRate) public onlyOperator {
require(_transferTaxRate <= MAXIMUM_TRANSFER_TAX_RATE, "SIBERIAN::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/SiberianToken.sol`

Locations

```
* Can only be called by the current operator.
*/
function updateBurnRate(uint16 _burnRate) public onlyOperator {
require(_burnRate <= 100, "SIBERIAN::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/SiberianToken.sol

Locations

```
* Can only be called by the current operator.
*/
function updateMaxTransferAmountRate(uint16 _maxTransferAmountRate) public onlyOperator {
require(_maxTransferAmountRate <= 10000, "SIBERIAN::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/SiberianToken.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/SiberianToken.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/SiberianToken.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 "updateSiberianSwapRouter" 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/SiberianToken.sol

Locations

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

/**
```

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/SiberianToken.sol

Locations

```
* Can only be called by the current operator.
*/
function transferOperator(address newOperator) public onlyOperator {
require(newOperator != address(0), "SIBERIAN::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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.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/SiberianToken.sol`

Locations

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

pragma solidity >=0.4.0;

```

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/SiberianToken.sol`

Locations

```
returns (uint256)
{
require(blockNumber < block.number, "SIBERIAN::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/SiberianToken.sol`

Locations

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

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