MythX

REPORT 609D7C655B428600183ADB14

Created	Thu May 13 2021 19:22:13 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	609d7b6f8bfa12ed16f28fb0

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
c9b63408-b673-4d75-bf7d-0f1b4ac6f465	/contracts/cougartoken.sol	14

Analysis `c9b63408-b673-4d75-bf7d-0f1b4ac6f465` MythX

Started	Thu May 13 2021 19:22:20 GMT+0000 (Coordinated Universal Time)
Finished	Thu May 13 2021 19:24:44 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Vscode-Extension
Main Source File	`/Contracts/Cougartoken.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	10	4

ISSUES

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/cougartoken.sol`

Locations

```

/// @dev overrides transfer function to meet tokenomics of COUGAR
function _transfer(address sender, address recipient, uint256 amount) internal virtual override antiWhale(sender, recipient, amount) {
// swap and liquify
if (
swapAndLiquifyEnabled == true swapAndLiquifyEnabled == true
&& _inSwapAndLiquify == false
&& address(cougarSwapRouter) != address(0)
```

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/cougartoken.sol

Locations

```

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

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/cougartoken.sol

Locations

```
function updateTransferTaxRate(uint16 _transferTaxRate) public onlyOperator {
require(_transferTaxRate <= MAXIMUM_TRANSFER_TAX_RATE, "COUGAR::updateTransferTaxRate: Transfer tax rate must not exceed the maximum rate.");
emit TransferTaxRateUpdated(msg.sender, transferTaxRate, _transferTaxRate);
transferTaxRate = _transferTaxRate;
}

/**
* @dev Update the burn rate.
* Can only be called by the current operator.
*/
function updateBurnRate(uint16 _burnRate) public onlyOperator {
require(_burnRate <= 100, "COUGAR::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 "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/cougartoken.sol

Locations

```
require(_burnRate <= 100, "COUGAR::updateBurnRate: Burn rate must not exceed the maximum rate.");
emit BurnRateUpdated(msg.sender, burnRate, _burnRate);
burnRate = _burnRate;
}

/**
* @dev Update the max transfer amount rate.
* Can only be called by the current operator.
*/
function updateMaxTransferAmountRate(uint16 _maxTransferAmountRate) public onlyOperator {
require(_maxTransferAmountRate <= 10000, "COUGAR::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 "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/cougartoken.sol

Locations

```
function updateMaxTransferAmountRate(uint16 _maxTransferAmountRate) public onlyOperator {
require(_maxTransferAmountRate <= 10000, "COUGAR::updateMaxTransferAmountRate: Max transfer amount rate must not exceed the maximum rate.");
emit MaxTransferAmountRateUpdated(msg.sender, maxTransferAmountRate, _maxTransferAmountRate);
maxTransferAmountRate = _maxTransferAmountRate;
}

/**
* @dev Update the min amount to liquify.
* 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 "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/cougartoken.sol

Locations

```

/**
* @dev Exclude or include an address from antiWhale.
* 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 "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/cougartoken.sol

Locations

```
/**
* @dev Update the swapAndLiquifyEnabled.
* 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 "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/cougartoken.sol

Locations

```
/**
* @dev Update the swap router.
* Can only be called by the current operator.
*/
function updateCougarSwapRouter(address _router) public onlyOperator {
cougarSwapRouter = IUniswapV2Router02(_router);
cougarSwapPair = IUniswapV2Factory(cougarSwapRouter.factory()).getPair(address(this), cougarSwapRouter.WETH());
require(cougarSwapPair != address(0), "COUGAR::updateCougarSwapRouter: Invalid pair address.");
emit CougarSwapRouterUpdated(msg.sender, address(cougarSwapRouter), cougarSwapPair);
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "updateCougarSwapRouter" 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/cougartoken.sol

Locations

```
cougarSwapRouter = IUniswapV2Router02(_router);
cougarSwapPair = IUniswapV2Factory(cougarSwapRouter.factory()).getPair(address(this), cougarSwapRouter.WETH());
require(cougarSwapPair != address(0), "COUGAR::updateCougarSwapRouter: Invalid pair address.");
emit CougarSwapRouterUpdated(msg.sender, address(cougarSwapRouter), cougarSwapPair);
}

/**
* @dev Returns the address of the current operator.
*/
function operator() public view returns (address) {
return _operator;
}

/**
* @dev Transfers operator of the contract to a new account (`newOperator`).
* Can only be called by the current operator.
*/
```

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/cougartoken.sol

Locations

```
require(newOperator != address(0), "COUGAR::transferOperator: new operator is the zero address");
emit OperatorTransferred(_operator, newOperator);
_operator = newOperator;
}

// Copied and modified from YAM code:
// https://github.com/yam-finance/yam-protocol/blob/master/contracts/token/YAMGovernanceStorage.sol
// https://github.com/yam-finance/yam-protocol/blob/master/contracts/token/YAMGovernance.sol
// Which is copied and modified from COMPOUND:
// https://github.com/compound-finance/compound-protocol/blob/master/contracts/Governance/Comp.sol
```

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/cougartoken.sol`

Locations

```

// First check most recent balance
if (checkpoints[account][nCheckpoints - 1].fromBlock <= blockNumber) {
return checkpoints[account][nCheckpoints - 1].votes;
}
```

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/cougartoken.sol`

Locations

```
if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
checkpoints[delegatee][nCheckpoints - 1].votes = newVotes;
} else {
checkpoints[delegatee][nCheckpoints] = Checkpoint(blockNumber, newVotes);
numCheckpoints[delegatee] = nCheckpoints + 1;
}
```

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/cougartoken.sol`

Locations

```

// First check most recent balance
if (checkpoints[account][nCheckpoints - 1].fromBlock <= blockNumber) {
return checkpoints[account][nCheckpoints - 1].votes;
}
```

LOW

SWC-123

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Source file

/contracts/cougartoken.sol

Locations

```
cougarSwapPair = IUniswapV2Factory(cougarSwapRouter.factory()).getPair(address(this), cougarSwapRouter.WETH());
require(cougarSwapPair != address(0), "COUGAR::updateCougarSwapRouter: Invalid pair address.");
emit CougarSwapRouterUpdated(msg.sender, address(cougarSwapRouter), cougarSwapPair);
}

/**
```