MythX

REPORT 628267D095CF250019B60E81

Created	Mon May 16 2022 15:03:44 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	628192a6c8c2c7865327e4f7

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
f4494528-2b6f-4fcf-8efa-b57e8fc2d94a	SCTCarbonTreasury.sol	4

Analysis f4494528-2b6f-4fcf-8efa-b57e8fc2d94a MythX

Started	Mon May 16 2022 15:03:48 GMT+0000 (Coordinated Universal Time)
Finished	Mon May 16 2022 15:05:01 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Mythx-Cli-0.6.22
Main Source File	SCTCarbonTreasury.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	4

ISSUES

UNKNOWN Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

SCTCarbonTreasury.sol

Locations

```
SCT.transferFrom(msg.sender, address(this), _offer.totalValue);

offerIdCounter ++;
offers[offerIdCounter] = _offer;
offers[offerIdCounter].statusOffer = StatusOffer.OPEN;
```

UNKNOWN Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

SCTCarbonTreasury.sol

Locations

```
offers[_offerId].statusOffer = StatusOffer.EXECUTED;

carbonProjectBalances[offer.token][offer.tokenId][msg.sender] -= offer.amount;
carbonProjectTons[offer.token][offer.tokenId] -= offer.amount;
totalReserves -= offer.amount;
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

SCTCarbonTreasury.sol

Locations

```

carbonProjectBalances[offer.token][offer.tokenId][msg.sender] -= offer.amount;
carbonProjectTons[offer.token][offer.tokenId] -= offer.amount;
totalReserves -= offer.amount;
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

SCTCarbonTreasury.sol

Locations

```
carbonProjectBalances[offer.token][offer.tokenId][msg.sender] -= offer.amount;
carbonProjectTons[offer.token][offer.tokenId] -= offer.amount;
totalReserves -= offer.amount;

SCT.burn(offer.amount);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

SCTCarbonTreasury.sol

Locations

```
SCT.burn(offer.amount);

if(offer.totalValue - offer.amount > 0) {
SCT.transfer(msg.sender, offer.totalValue - offer.amount);
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

SCTCarbonTreasury.sol

Locations

```

if(offer.totalValue - offer.amount > 0) {
SCT.transfer(msg.sender, offer.totalValue - offer.amount);
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`SCTCarbonTreasury.sol`

Locations

```
SCT.mint(_owner, _amount);

carbonProjectBalances[_token][_tokenId][_owner] += _amount;
carbonProjectTons[_token][_tokenId] += _amount;
totalReserves += _amount;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`SCTCarbonTreasury.sol`

Locations

```

carbonProjectBalances[_token][_tokenId][_owner] += _amount;
carbonProjectTons[_token][_tokenId] += _amount;
totalReserves += _amount;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`SCTCarbonTreasury.sol`

Locations

```
carbonProjectBalances[_token][_tokenId][_owner] += _amount;
carbonProjectTons[_token][_tokenId] += _amount;
totalReserves += _amount;

emit Deposited(_token, _tokenId, _owner, _amount);
```

UNKNOWN Arithmetic operation "++" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

SCTCarbonTreasury.sol

Locations

```
function indexInRegistry(address _address, STATUS _status) public view returns (bool, uint256) {
address[] memory entries = registry[_status];
for (uint256 i = 0; i < entries.length; i++) {
if (_address == entries[i]) {
return (true, i);
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

SCTCarbonTreasury.sol

Locations

```
require(timelockEnabled, "SCT Treasury: timelock is disabled, use enable");

uint256 timelock = block.number + blocksNeededForOrder;
permissionOrder.push(
Order({
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

SCTCarbonTreasury.sol

Locations

```
function permissionToDisableTimelock() external onlyGovernor {
require(timelockEnabled, "SCT Treasury: timelock already disabled");
onChainGovernanceTimelock = block.number + (blocksNeededForOrder * 10);
emit SetOnChainGovernanceTimelock(onChainGovernanceTimelock);
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

SCTCarbonTreasury.sol

Locations

```
function permissionToDisableTimelock() external onlyGovernor {
require(timelockEnabled, "SCT Treasury: timelock already disabled");
onChainGovernanceTimelock = block.number + (blocksNeededForOrder * 10);
emit SetOnChainGovernanceTimelock(onChainGovernanceTimelock);
}
```

UNKNOWN

SWC-110

Public state variable with array type causing reacheable exception by default.

The public state variable "registry" in "SCTCarbonTreasury" contract has type "mapping(enum SCTCarbonTreasury.STATUS => address[])" and can cause an exception in case of use of invalid array index value.

Source file

SCTCarbonTreasury.sol

Locations

```
* @return array of addresses
*/
mapping(STATUS => address[]) public registry;

/**
```

UNKNOWN

SWC-110

Public state variable with array type causing reacheable exception by default.

The public state variable "permissionOrder" in "SCTCarbonTreasury" contract has type "struct SCTCarbonTreasury.Order[]" and can cause an exception in case of use of invalid array index value.

Source file

SCTCarbonTreasury.sol

Locations

```
* @dev return Order[]
*/
Order[] public permissionOrder;

/**
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

SCTCarbonTreasury.sol

Locations

```
address[] memory entries = registry[_status];
for (uint256 i = 0; i < entries.length; i++) {
if (_address == entries[i]) {
return (true, i);
}
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

SCTCarbonTreasury.sol

Locations

```
require(timelockEnabled, "SCT Treasury: timelock is disabled, use enable");

Order memory info = permissionOrder[_index];

require(!info.nullify, "SCT Treasury: order has been nullified");
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

SCTCarbonTreasury.sol

Locations

```
registry[info.managing].push(info.toPermit);
}
permissionOrder[_index].executed = true;

emit Permissioned(info.managing, info.toPermit, true);
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

SCTCarbonTreasury.sol

Locations

```
*/
function nullify(uint256 _index) external onlyGovernor returns(bool) {
permissionOrder[_index].nullify = true;
return true;
}
```

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

SCTCarbonTreasury.sol

Locations

```
require(timelockEnabled, "SCT Treasury: timelock is disabled, use enable");

uint256 timelock = block.number + blocksNeededForOrder;
permissionOrder.push(
Order({
```

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

SCTCarbonTreasury.sol

Locations

```
require(!info.nullify, "SCT Treasury: order has been nullified");
require(!info.executed, "SCT Treasury: order has already been executed");
require(block.number >= info.timelockEnd, "SCT Treasury: timelock not complete");

permissions[info.managing][info.toPermit] = true;
```

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

`SCTCarbonTreasury.sol`

Locations

```
function disableTimelock() external onlyGovernor {
require(timelockEnabled, "SCT Treasury: timelock already disabled");
require(onChainGovernanceTimelock != 0 && onChainGovernanceTimelock <= block.number, "SCT Treasury: governance timelock not expired yet");
timelockEnabled = false;
onChainGovernanceTimelock = 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

`SCTCarbonTreasury.sol`

Locations

```
function permissionToDisableTimelock() external onlyGovernor {
require(timelockEnabled, "SCT Treasury: timelock already disabled");
onChainGovernanceTimelock = block.number + (blocksNeededForOrder * 10);
emit SetOnChainGovernanceTimelock(onChainGovernanceTimelock);
}
```