MythX

REPORT 628D6B2574BE140019ADE14C

Created	Tue May 24 2022 23:32:53 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	6197960e3494e9c8c076e89b

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
4865328d-4dd1-4fdb-a55e-89bd5c1740ab	Voter.sol	0

Analysis 4865328d-4dd1-4fdb-a55e-89bd5c1740ab MythX

Started	Tue May 24 2022 23:33:03 GMT+0000 (Coordinated Universal Time)
Finished	Tue May 24 2022 23:33:08 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Voter.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

UNKNOWN SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```
function initialize(address[] memory _tokens, address _minter) external {
require(msg.sender == minter);
for (uint i = 0; i < _tokens.length; i++) {
_whitelist(_tokens[i]);
}
```

UNKNOWN SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```
uint256 _totalWeight = 0;

for (uint i = 0; i < _poolVoteCnt; i ++) {
address _pool = _poolVote[i];
uint256 _votes = votes[_tokenId][_pool];
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

Voter.sol

Locations

```
if (_votes != 0) {
_updateFor(gauges[_pool]);
weights[_pool] -= _votes;
votes[_tokenId][_pool] -= _votes;
if (_votes > 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

Voter.sol

Locations

```
_updateFor(gauges[_pool]);
weights[_pool] -= _votes;
votes[_tokenId][_pool] -= _votes;
if (_votes > 0) {
IBribe(bribes[gauges[_pool]])._withdraw(uint256(_votes), _tokenId);
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```
if (_votes > 0) {
IBribe(bribes[gauges[_pool]])._withdraw(uint256(_votes), _tokenId);
_totalWeight += _votes;
} else {
_totalWeight -= _votes;
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

Voter.sol

Locations

```
_totalWeight += _votes;
} else {
_totalWeight -= _votes;
}
emit Abstained(_tokenId, _votes);
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

Voter.sol

Locations

```
}
}
totalWeight -= uint256(_totalWeight);
usedWeights[_tokenId] = 0;
delete poolVote[_tokenId];
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```
uint256[] memory _weights = new uint256[](_poolCnt);

for (uint i = 0; i < _poolCnt; i ++) {
_weights[i] = votes[_tokenId][_poolVote[i]];
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Voter.sol`

Locations

```
uint256 _usedWeight = 0;

for (uint i = 0; i < _poolCnt; i++) {
_totalVoteWeight += _weights[i];
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`Voter.sol`

Locations

```

for (uint i = 0; i < _poolCnt; i++) {
_totalVoteWeight += _weights[i];
}

```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Voter.sol`

Locations

```
}

for (uint i = 0; i < _poolCnt; i++) {
address _pool = _poolVote[i];
address _gauge = gauges[_pool];
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`Voter.sol`

Locations

```

if (isGauge[_gauge]) {
uint256 _poolWeight = _weights[i] * _weight / _totalVoteWeight;
require(votes[_tokenId][_pool] == 0);
require(_poolWeight != 0);
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`Voter.sol`

Locations

```

if (isGauge[_gauge]) {
uint256 _poolWeight = _weights[i] * _weight / _totalVoteWeight;
require(votes[_tokenId][_pool] == 0);
require(_poolWeight != 0);
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`Voter.sol`

Locations

```
poolVote[_tokenId].push(_pool);

weights[_pool] += _poolWeight;
votes[_tokenId][_pool] += _poolWeight;
IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```

weights[_pool] += _poolWeight;
votes[_tokenId][_pool] += _poolWeight;
IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
_usedWeight += _poolWeight;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```
votes[_tokenId][_pool] += _poolWeight;
IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
_usedWeight += _poolWeight;
_totalWeight += _poolWeight;
emit Voted(msg.sender, _tokenId, _poolWeight);
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```
IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
_usedWeight += _poolWeight;
_totalWeight += _poolWeight;
emit Voted(msg.sender, _tokenId, _poolWeight);
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```
}
if (_usedWeight > 0) IVotingEscrow(_ve).voting(_tokenId);
totalWeight += uint256(_totalWeight);
usedWeights[_tokenId] = uint256(_usedWeight);
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Voter.sol

Locations

```
function notifyRewardAmount(uint amount) external {
_safeTransferFrom(base, msg.sender, address(this), amount); // transfer the distro in
uint256 _ratio = amount * 1e18 / totalWeight; // 1e18 adjustment is removed during claim
if (_ratio > 0) {
index += _ratio;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

Voter.sol

Locations

```
function notifyRewardAmount(uint amount) external {
_safeTransferFrom(base, msg.sender, address(this), amount); // transfer the distro in
uint256 _ratio = amount * 1e18 / totalWeight; // 1e18 adjustment is removed during claim
if (_ratio > 0) {
index += _ratio;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```
uint256 _ratio = amount * 1e18 / totalWeight; // 1e18 adjustment is removed during claim
if (_ratio > 0) {
index += _ratio;
}
emit NotifyReward(msg.sender, base, amount);
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```

function updateFor(address[] memory _gauges) external {
for (uint i = 0; i < _gauges.length; i++) {
_updateFor(_gauges[i]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```

function updateForRange(uint start, uint end) public {
for (uint i = start; i < end; i++) {
_updateFor(gauges[pools[i]]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Voter.sol`

Locations

```
uint _index = index; // get global index0 for accumulated distro
supplyIndex[_gauge] = _index; // update _gauge current position to global position
uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
if (_delta > 0) {
uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`Voter.sol`

Locations

```
uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
if (_delta > 0) {
uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
claimable[_gauge] += _share;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`Voter.sol`

Locations

```
uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
if (_delta > 0) {
uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
claimable[_gauge] += _share;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Voter.sol

Locations

```
if (_delta > 0) {
uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
claimable[_gauge] += _share;
}
} else {
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```

function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
for (uint i = 0; i < _gauges.length; i++) {
IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Voter.sol

Locations

```
function claimBribes(address[] memory _bribes, address[][] memory _tokens, uint _tokenId) external {
require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
for (uint i = 0; i < _bribes.length; i++) {
IBribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Voter.sol`

Locations

```
function claimFees(address[] memory _fees, address[][] memory _tokens, uint _tokenId) external {
require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
for (uint i = 0; i < _fees.length; i++) {
IBribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Voter.sol`

Locations

```

function distributeFees(address[] memory _gauges) external {
for (uint i = 0; i < _gauges.length; i++) {
IGauge(_gauges[i]).claimFees();
}
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`Voter.sol`

Locations

```
_updateFor(_gauge);
uint _claimable = claimable[_gauge];
if (_claimable > IGauge(_gauge).left(base) && _claimable / DURATION > 0) {
claimable[_gauge] = 0;
IGauge(_gauge).notifyRewardAmount(base, _claimable);
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Voter.sol`

Locations

```

function distribute(uint start, uint finish) public {
for (uint x = start; x < finish; x++) {
distribute(gauges[pools[x]]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Voter.sol`

Locations

```

function distribute(address[] memory _gauges) external {
for (uint x = 0; x < _gauges.length; x++) {
distribute(_gauges[x]);
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`libraries/Math.sol`

Locations

```
if (y > 3) {
z = y;
uint x = y / 2 + 1;
while (x < z) {
z = x;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`libraries/Math.sol`

Locations

```
if (y > 3) {
z = y;
uint x = y / 2 + 1;
while (x < z) {
z = x;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`libraries/Math.sol`

Locations

```
while (x < z) {
z = x;
x = (y / x + x) / 2;
}
} else if (y != 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`libraries/Math.sol`

Locations

```
while (x < z) {
z = x;
x = (y / x + x) / 2;
}
} else if (y != 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`libraries/Math.sol`

Locations

```
while (x < z) {
z = x;
x = (y / x + x) / 2;
}
} else if (y != 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`libraries/Math.sol`

Locations

```
for (uint256 y = 1 << 255; y > 0; y >>= 3) {
x <<= 1;
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`libraries/Math.sol`

Locations

```
for (uint256 y = 1 << 255; y > 0; y >>= 3) {
x <<= 1;
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`libraries/Math.sol`

Locations

```
for (uint256 y = 1 << 255; y > 0; y >>= 3) {
x <<= 1;
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`libraries/Math.sol`

Locations

```
for (uint256 y = 1 << 255; y > 0; y >>= 3) {
x <<= 1;
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`libraries/Math.sol`

Locations

```
x <<= 1;
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
x += 1;
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

`libraries/Math.sol`

Locations

```
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
x += 1;
}
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`libraries/Math.sol`

Locations

```
uint256 z = 3 * x * (x + 1) + 1;
if (n / y >= z) {
n -= y * z;
x += 1;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`libraries/Math.sol`

Locations

```
if (n / y >= z) {
n -= y * z;
x += 1;
}
}
```

UNKNOWN

SWC-110

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

The public state variable "pools" in "Voter" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source file

`Voter.sol`

Locations

```
uint public totalWeight; // total voting weight

address[] public pools; // all pools viable for incentives
mapping(address => address) public gauges; // pool => gauge
mapping(address => address) public poolForGauge; // gauge => pool
```

UNKNOWN

SWC-110

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

The public state variable "poolVote" in "Voter" contract has type "mapping(uint256 => address[])" and can cause an exception in case of use of invalid array index value.

Source file

`Voter.sol`

Locations

```
mapping(address => uint256) public weights; // pool => weight
mapping(uint => mapping(address => uint256)) public votes; // nft => pool => votes
mapping(uint => address[]) public poolVote; // nft => pools
mapping(uint => uint) public usedWeights; // nft => total voting weight of user
mapping(address => bool) public isGauge;
```

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

`Voter.sol`

Locations

```
require(msg.sender == minter);
for (uint i = 0; i < _tokens.length; i++) {
_whitelist(_tokens[i]);
}
minter = _minter;
```

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

Voter.sol

Locations

```
for (uint i = 0; i < _poolVoteCnt; i ++) {
address _pool = _poolVote[i];
uint256 _votes = votes[_tokenId][_pool];
```

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

Voter.sol

Locations

```
for (uint i = 0; i < _poolCnt; i ++) {
_weights[i] = votes[_tokenId][_poolVote[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

Voter.sol

Locations

```
for (uint i = 0; i < _poolCnt; i ++) {
_weights[i] = votes[_tokenId][_poolVote[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

`Voter.sol`

Locations

```
for (uint i = 0; i < _poolCnt; i++) {
_totalVoteWeight += _weights[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

`Voter.sol`

Locations

```
for (uint i = 0; i < _poolCnt; i++) {
address _pool = _poolVote[i];
address _gauge = gauges[_pool];
```

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

`Voter.sol`

Locations

```
if (isGauge[_gauge]) {
uint256 _poolWeight = _weights[i] * _weight / _totalVoteWeight;
require(votes[_tokenId][_pool] == 0);
require(_poolWeight != 0);
```

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

Voter.sol

Locations

```
function updateFor(address[] memory _gauges) external {
for (uint i = 0; i < _gauges.length; i++) {
_updateFor(_gauges[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

Voter.sol

Locations

```
function updateForRange(uint start, uint end) public {
for (uint i = start; i < end; i++) {
_updateFor(gauges[pools[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

Voter.sol

Locations

```
function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
for (uint i = 0; i < _gauges.length; i++) {
IGauge(_gauges[i]).getReward(msg.sender, _tokens[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

`Voter.sol`

Locations

```
function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
for (uint i = 0; i < _gauges.length; i++) {
IGauge(_gauges[i]).getReward(msg.sender, _tokens[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

`Voter.sol`

Locations

```
require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
for (uint i = 0; i < _bribes.length; i++) {
IBribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[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

`Voter.sol`

Locations

```
require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
for (uint i = 0; i < _bribes.length; i++) {
IBribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[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

Voter.sol

Locations

```
require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
for (uint i = 0; i < _fees.length; i++) {
IBribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[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

Voter.sol

Locations

```
require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
for (uint i = 0; i < _fees.length; i++) {
IBribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[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

Voter.sol

Locations

```
function distributeFees(address[] memory _gauges) external {
for (uint i = 0; i < _gauges.length; i++) {
IGauge(_gauges[i]).claimFees();
}
}
```

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

`Voter.sol`

Locations

```
function distribute(uint start, uint finish) public {
for (uint x = start; x < finish; x++) {
distribute(gauges[pools[x]]);
}
}
```

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

`Voter.sol`

Locations

```
function distribute(address[] memory _gauges) external {
for (uint x = 0; x < _gauges.length; x++) {
distribute(_gauges[x]);
}
}
```