

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

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 Arithmetic operation "++" discovered
This plugin produces issues to support false positive discovery within MythX.
SWC-101

Source file
Voter.sol
Locations

```
74 | function initialize(address[] memory _tokens, address _minter) external {  
75 |     require(msg.sender == minter);  
76 |     for (uint i = 0; i < _tokens.length; i++) {  
77 |         _whitelist(_tokens[i]);  
78 |     }
```

UNKNOWN Arithmetic operation "++" discovered
This plugin produces issues to support false positive discovery within MythX.
SWC-101

Source file
Voter.sol
Locations

```
101 | uint256 _totalWeight = 0;  
102 |  
103 | for (uint i = 0; i < _poolVoteCnt; i++) {  
104 |     address _pool = _poolVote[i];  
105 |     uint256 _votes = votes[_tokenId][_pool];
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
107 | if (_votes != 0) {  
108 |     _updateFor(gauges[_pool]);  
109 |     weights[_pool] -= _votes;  
110 |     votes[_tokenId][_pool] -= _votes;  
111 |     if (_votes > 0) {
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Voter.sol

Locations

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

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
111 |     if (_votes > 0) {  
112 |         IBribe(bribes[gauges[_pool]])._withdraw(uint256(_votes), _tokenId);  
113 |         totalWeight += _votes;  
114 |     } else {  
115 |         _totalWeight -= _votes;
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
113 | _totalWeight += _votes;
114 | } else {
115 |     totalWeight -= _votes;
116 | }
117 | emit Abstained(_tokenId, _votes);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
118 | }
119 | }
120 | totalWeight -= uint256(_totalWeight);
121 | usedWeights[_tokenId] = 0;
122 | delete poolVote[_tokenId];
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
128 | uint256[] memory _weights = new uint256[](_poolCnt);
129 |
130 | for (uint i = 0; i < _poolCnt; i++) {
131 |     _weights[i] = votes[_tokenId][_poolVote[i]];
132 | }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
143 | uint256 _usedWeight = 0;
144 |
145 | for (uint i = 0; i < _poolCnt; i++) {
146 |     _totalVoteWeight += _weights[i];
147 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
144 |
145 | for (uint i = 0; i < _poolCnt; i++) {
146 |     _totalVoteWeight += _weights[i];
147 | }
148 |
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
147 | }
148 |
149 | for (uint i = 0; i < _poolCnt; i++) {
150 |     address _pool = _poolVote[i];
151 |     address _gauge = gauges[_pool];
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
152 |  
153 | if (isGauge[_gauge]) {  
154 |     uint256 _poolWeight = weights[i] * _weight / _totalVoteWeight;  
155 |     require(votes[_tokenId][_pool] == 0);  
156 |     require(_poolWeight != 0);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
152 |  
153 | if (isGauge[_gauge]) {  
154 |     uint256 _poolWeight = weights[i] * _weight / _totalVoteWeight;  
155 |     require(votes[_tokenId][_pool] == 0);  
156 |     require(_poolWeight != 0);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
159 | poolVote[_tokenId].push(_pool);  
160 |  
161 | weights[_pool] += _poolWeight;  
162 | votes[_tokenId][_pool] += _poolWeight;  
163 | IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
160 |
161 | weights[_pool] += _poolWeight;
162 | votes[_tokenId][_pool] += _poolWeight;
163 | IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
164 | _usedWeight += _poolWeight;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

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

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
163 | IBribe(bribes[_gauge])._deposit(uint256(_poolWeight), _tokenId);
164 | _usedWeight += _poolWeight;
165 | _totalWeight += _poolWeight;
166 | emit Voted(msg.sender, _tokenId, _poolWeight);
167 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
168 | }
169 | if (_usedWeight > 0) IVotingEscrow(_ve).voting(_tokenId);
170 | totalWeight += uint256(_totalWeight);
171 | usedWeights[_tokenId] = uint256(_usedWeight);
172 | }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Voter.sol

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Voter.sol

Locations

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


UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
260 | uint256 _ratio = amount * 1e18 / totalWeight; // 1e18 adjustment is removed during claim
261 | if (_ratio > 0) {
262 |     index += _ratio;
263 | }
264 | emit NotifyReward(msg.sender, base, amount);
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
266 |
267 | function updateFor(address[] memory _gauges) external {
268 |     for (uint i = 0; i < _gauges.length; i++) {
269 |         _updateFor(_gauges[i]);
270 |     }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
272 |
273 | function updateForRange(uint start, uint end) public {
274 |     for (uint i = start; i < end; i++) {
275 |         _updateFor(gauges[pools[i]]);
276 |     }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Voter.sol

Locations

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

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
295 | uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
296 | if (_delta > 0) {
297 |     uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
298 |     claimable[_gauge] += _share;
299 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
295 | uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
296 | if (_delta > 0) {
297 |     uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
298 |     claimable[_gauge] += _share;
299 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
296 | if (_delta > 0) {
297 |     uint _share = uint(_supplied) * _delta / 1e18; // add accrued difference for each supplied token
298 |     claimable_gauge += _share;
299 | }
300 | } else {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
304 |
305 | function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
306 |     for (uint i = 0; i < _gauges.length; i++) {
307 |         IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
308 |     }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
311 | function claimBribes(address[] memory _bribes, address[][] memory _tokens, uint _tokenId) external {
312 |     require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
313 |     for (uint i = 0; i < _bribes.length; i++) {
314 |         IBribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
315 |     }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
318 | function claimFees(address[] memory _fees, address[][] memory _tokens, uint _tokenId) external {
319 |     require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
320 |     for (uint i = 0; i < _fees.length; i++) {
321 |         IBribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
322 |     }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
324 |
325 | function distributeFees(address[] memory _gauges) external {
326 |     for (uint i = 0; i < _gauges.length; i++) {
327 |         IGauge(_gauges[i]).claimFees();
328 |     }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Voter.sol

Locations

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

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
350 |
351 | function distribute(uint start, uint finish) public {
352 |   for (uint x = start; x < finish; x++) {
353 |     distribute(gauges[pools[x]]);
354 |   }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Voter.sol

Locations

```
356 |
357 | function distribute(address[] memory _gauges) external {
358 |   for (uint x = 0; x < _gauges.length; x++) {
359 |     distribute(_gauges[x]);
360 |   }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
11 | if (y > 3) {
12 |   z = y;
13 |   uint x = y / 2 + 1;
14 |   while (x < z) {
15 |     z = x;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
11 | if (y > 3) {
12 |     z = y;
13 |     uint x = y / 2 + 1;
14 |     while (x < z) {
15 |         z = x;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
14 | while (x < z) {
15 |     z = x;
16 |     x = y / x + x / 2;
17 | }
18 | else if (y != 0) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
14 | while (x < z) {
15 |     z = x;
16 |     x = (y / x + x) / 2;
17 | }
18 | else if (y != 0) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
14 | while (x < z) {
15 |     z = x;
16 |     x = (y / x + x) / 2;
17 | }
18 | } else if (y != 0) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
24 | for (uint256 y = 1 << 255; y > 0; y >= 3) {
25 |     x <<= 1;
26 |     uint256 z = 5 + x * (x + 1) + 1;
27 |     if (n / y >= z) {
28 |         n -= y * z;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
24 | for (uint256 y = 1 << 255; y > 0; y >= 3) {
25 |     x <<= 1;
26 |     uint256 z = 5 + x * (x + 1) + 1;
27 |     if (n / y >= z) {
28 |         n -= y * z;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
24 | for (uint256 y = 1 << 255; y > 0; y >= 3) {
25 |     x <= 1;
26 |     uint256 z = 3 * x * (x + 1) + 1;
27 |     if (n / y >= z) {
28 |         n -= y * z;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
24 | for (uint256 y = 1 << 255; y > 0; y >= 3) {
25 |     x <= 1;
26 |     uint256 z = 3 * x * (x + 1) + 1;
27 |     if (n / y >= z) {
28 |         n -= y * z;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
25 | x <= 1;
26 | uint256 z = 3 * x * (x + 1) + 1;
27 | if (n / y >= z) {
28 |     n -= y * z;
29 |     x += 1;
```


UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
26 | uint256 z = 3 * x * (x + 1) + 1;
27 | if (n / y >= z) {
28 |   n -= y * z;
29 |   x += 1;
30 | }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
26 | uint256 z = 3 * x * (x + 1) + 1;
27 | if (n / y >= z) {
28 |   n -= y * z;
29 |   x += 1;
30 | }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

libraries/Math.sol

Locations

```
27 | if (n / y >= z) {
28 |   n -= y * z;
29 |   x += 1;
30 | }
31 | }
```

UNKNOWN Public state variable with array type causing reachable 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.

SWC-110

Source file

Voter.sol

Locations

```
27 | uint public totalWeight; // total voting weight
28 |
29 | address[] public pools; // all pools viable for incentives
30 | mapping(address => address) public gauges; // pool => gauge
31 | mapping(address => address) public poolForGauge; // gauge => pool
```

UNKNOWN Public state variable with array type causing reachable 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.

SWC-110

Source file

Voter.sol

Locations

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

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
75 | require(msg.sender == minter);
76 | for (uint i = 0; i < _tokens.length; i++) {
77 |     _whitelist(_tokens[i]);
78 | }
79 | minter = _minter;
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
102 |
103 | for (uint i = 0; i < _poolVoteCnt; i++) {
104 |     address _pool = _poolVote[i];
105 |     uint256 _votes = votes[_tokenId][_pool];
106 |
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
129 |
130 | for (uint i = 0; i < _poolCnt; i++) {
131 |     _weights[i] = votes[_tokenId][_poolVote[i]];
132 | }
133 |
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
129 |
130 | for (uint i = 0; i < _poolCnt; i++) {
131 |     _weights[i] = votes[_tokenId][_poolVote[i]];
132 | }
133 |
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
144 |
145 | for (uint i = 0; i < _poolCnt; i++) {
146 |     _totalVoteWeight += _weights[i];
147 | }
148 |
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
148 |
149 | for (uint i = 0; i < _poolCnt; i++) {
150 |     address _pool = _poolVote[i];
151 |     address _gauge = gauges[_pool];
152 |
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
152 |
153 | if (isGauge[_gauge]) {
154 |     uint256 _poolWeight = _weights[i] * _weight / _totalVoteWeight;
155 |     require(votes[_tokenId][_pool] == 0);
156 |     require(_poolWeight != 0);
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
267 | function updateFor(address[] memory _gauges) external {  
268 |     for (uint i = 0; i < _gauges.length; i++) {  
269 |         _updateFor(_gauges[i]);  
270 |     }  
271 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
273 | function updateForRange(uint start, uint end) public {  
274 |     for (uint i = start; i < end; i++) {  
275 |         _updateFor(gauges[pools[i]]);  
276 |     }  
277 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
305 | function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {  
306 |     for (uint i = 0; i < _gauges.length; i++) {  
307 |         IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);  
308 |     }  
309 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
305 | function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
306 |     for (uint i = 0; i < _gauges.length; i++) {
307 |         IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
308 |     }
309 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
312 | require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
313 | for (uint i = 0; i < _bribes.length; i++) {
314 |     IBribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
315 | }
316 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
312 | require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
313 | for (uint i = 0; i < _bribes.length; i++) {
314 |     IBribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
315 | }
316 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
319 | require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
320 | for (uint i = 0; i < _fees.length; i++) {
321 |     IBribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
322 | }
323 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
319 | require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, _tokenId));
320 | for (uint i = 0; i < _fees.length; i++) {
321 |     IBribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
322 | }
323 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
325 | function distributeFees(address[] memory _gauges) external {
326 |     for (uint i = 0; i < _gauges.length; i++) {
327 |         IGauge(_gauges[i]).claimFees();
328 |     }
329 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
351 | function distribute(uint start, uint finish) public {  
352 |   for (uint x = start; x < finish; x++) {  
353 |     distribute(gauges[pools[x]]);  
354 |   }  
355 | }
```

UNKNOWN Out of bounds array access

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

SWC-110

Source file

Voter.sol

Locations

```
357 | function distribute(address[] memory _gauges) external {  
358 |   for (uint x = 0; x < _gauges.length; x++) {  
359 |     distribute(_gauges[x]);  
360 |   }  
361 | }
```