MythX

REPORT 61F78D757FC5810019157C5B

Created	Mon Jan 31 2022 07:19:17 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	61f52e351fd393a0c51a34fe

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
e8b437c0-ae9b-49d5-b80c-b531fe334a74	voter.sol	2

Analysis e8b437c0-ae9b-49d5-b80c-b531fe334a74 MythX

Started	Mon Jan 31 2022 07:19:21 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jan 31 2022 07:19:27 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Voter.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

UNKNOWN Arithmetic operation "**" discovered

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

SWC-101

Source file
voter.sol
Locations

```
58
59 uint public constant DURATION = 7 days; // rewards are released over 7 days
60 uint public constant PRECISION = 10 ** 18;
61
62 // default snx staking contract implementation
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file
voter.sol
Locations

```
145
146 // First check most recent balance
147 if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
148 return (nCheckpoints - 1);
149 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
146 // First check most recent balance
147 if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
148 return (nCheckpoints - 1);
149 }
150
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
155
156 uint lower = 0;
157 uint upper = nCheckpoints - 1;
158 while (upper > lower) {
159 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
157 uint upper = nCheckpoints - 1;
158 while (upper > lower) {
159 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
160 Checkpoint memory cp = checkpoints[tokenId][center];
161 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
157 uint upper = nCheckpoints - 1;
158 while (upper > lower) {
159 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
160 Checkpoint memory cp = checkpoints[tokenId][center];
161 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
157 uint upper = nCheckpoints - 1;
158 while (upper > lower) {
159 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
160 Checkpoint memory cp = checkpoints[tokenId][center];
161 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
164 lower = center;
165 } else {
166 upper = center - 1;
167 }
168 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
177
178 // First check most recent balance
179 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
180 return (nCheckpoints - 1);
181 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
178 // First check most recent balance
179 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
180 return (nCheckpoints - 1);
181 }
182
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
187
188 uint lower = 0;
189 uint upper = nCheckpoints - 1;
190 while (upper > lower) {
191 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
189 uint upper = nCheckpoints - 1;
190 while (upper > lower) {
191 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
192 SupplyCheckpoint memory cp = supplyCheckpoints[center];
193 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
189 uint upper = nCheckpoints - 1;
190 while (upper > lower) {
191 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
192 SupplyCheckpoint memory cp = supplyCheckpoints[center];
193 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
189 uint upper = nCheckpoints - 1;
190 while (upper > lower) {
191 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
192 SupplyCheckpoint memory cp = supplyCheckpoints[center];
193 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
196 lower = center;
197 } else {
198 upper = center - 1;
199 }
200 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
209
210 // First check most recent balance
211 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
212 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
213 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
210 // First check most recent balance
211 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
212 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
213 }
214
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
210 // First check most recent balance
211 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
212 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
213 }
214
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
219
220 uint lower = 0;
221 uint upper = nCheckpoints - 1;
222 while (upper > lower) {
223 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
221 uint upper = nCheckpoints - 1;
222 while (upper > lower) {
223 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
224 RewardPerTokenCheckpoint memory cp = rewardPerTokenCheckpoints[token][center];
225 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
221 uint upper = nCheckpoints - 1;
222 while (upper > lower) {
223 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
224 RewardPerTokenCheckpoint memory cp = rewardPerTokenCheckpoints[token][center];
225 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
221 uint upper = nCheckpoints - 1;
222 while (upper > lower) {
223 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
224 RewardPerTokenCheckpoint memory cp = rewardPerTokenCheckpoints[token][center];
225 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
228 lower = center;
229 } else {
230 upper = center - 1;
231 }
232 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
238 uint _nCheckPoints = numCheckpoints[tokenId];
239
240 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
241 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
242 } else {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
239
240 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
241 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
242 } else {
243 checkpoints[tokenId][_nCheckPoints] = Checkpoint(_timestamp, balance);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
242 } else {
243 checkpoints[tokenId][_nCheckPoints] = Checkpoint(_timestamp, balance);
244 numCheckpoints[tokenId] = _nCheckPoints + 1;
245 }
246 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
249 uint _nCheckPoints = rewardPerTokenNumCheckpoints[token];
250
251 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
252 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
253 } else {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
250
251 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
252 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
253 } else {
254 rewardPerTokenCheckpoints[token][_nCheckPoints] = RewardPerTokenCheckpoint(timestamp, reward);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
253 } else {
254 rewardPerTokenCheckpoints[token][_nCheckPoints] = RewardPerTokenCheckpoint(timestamp, reward);
255 rewardPerTokenNumCheckpoints[token] = _nCheckPoints + 1;
256 }
257 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
261 uint _timestamp = block.timestamp;
262
263 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
264 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
265 } else {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
262
263 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
264 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
265 } else {
266 supplyCheckpoints[_nCheckPoints] = SupplyCheckpoint(_timestamp, totalSupply);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
265 } else {
266 supplyCheckpoints[_nCheckPoints] = SupplyCheckpoint(_timestamp, totalSupply);
267 supplyNumCheckpoints = _nCheckPoints + 1;
268 }
269 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
286 function getReward(uint tokenId, address[] memory tokens) external lock {
287 require(ve(_ve).isApprovedOrOwner(msg.sender, tokenId));
288 for (uint i = 0; i < tokens.length; i++) {
289 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
290
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
303 require(msg.sender == factory);
304 address _owner = ve(_ve).ownerOf(tokenId);
305 for (uint i = 0; i < tokens.length; i++) {
306 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
307
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
320 return rewardPerTokenStored[token];
321 }
322 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
323 }
324
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
320 return rewardPerTokenStored[token];
321 }
322 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
323 }
324
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
320 return rewardPerTokenStored[token];
321 }
322 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
323 }
324
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
320 return rewardPerTokenStored[token];
321 }
322 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
323 }
324
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
320 return rewardPerTokenStored[token];
321 }
322 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
323 }
324
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
330
331 uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
332 uint _endIndex = Math.min(numCheckpoints[tokenId]-1, maxRuns);
333
334 uint reward = userRewards[token][tokenId];
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
333
334 uint reward = userRewards[token][tokenId];
335 for (uint i = _startIndex; i < _endIndex; i++) {
336 Checkpoint memory cp0 = checkpoints[tokenId][i];
337 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
335 for (uint i = _startIndex; i < _endIndex; i++) {
336 Checkpoint memory cp0 = checkpoints[tokenId][i];
337 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
338 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
339 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
338 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
339 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
340 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
341 _startTimestamp = cp1.timestamp;
342 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
338 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
339 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
340 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
341 _startTimestamp = cp1.timestamp;
342 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
338 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
339 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
340 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
341 _startTimestamp = cp1.timestamp;
342 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
338 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
339 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
340 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
341 _startTimestamp = cp1.timestamp;
342 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
358
359 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
360 uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);
361
362 for (uint i = _startIndex; i < _endIndex; i++) {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
360 uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);
361
362 for (uint i = _startIndex; i < _endIndex; i++) {
363 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
364 if (sp0.supply > 0) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
363 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
364 if (sp0.supply > 0) {
365 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
366 (uint _reward, uint endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
367 reward += _reward;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
365 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
366 (uint _reward, uint endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
367 reward += _reward;
368 _writeRewardPerTokenCheckpoint(token, reward, endTime);
369 _startTimestamp = endTime;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
376 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
377 uint endTime = Math.max(timestamp1, startTimestamp);
378 return ((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
379 }
380
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
376 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
377 uint endTime = Math.max(timestamp1, startTimestamp);
378 return ((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
379 }
380
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
376 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
377 uint endTime = Math.max(timestamp1, startTimestamp);
378 return ((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
379 }
380
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
376 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
377 uint endTime = Math.max(timestamp1, startTimestamp);
378 return (((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
379 }
380
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
388
389 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
390 uint _endIndex = supplyNumCheckpoints-1;
391
392 if (_endIndex - _startIndex > 1) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
390 uint _endIndex = supplyNumCheckpoints-1;
391
392 if (_endIndex - _startIndex > 1) {
393 for (uint i = _startIndex; i < _endIndex-1; i++) {
394 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
391
392 if (_endIndex - _startIndex > 1) {
393 for (uint i = _startIndex; i < _endIndex-1; i++) {
394 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
395 if (sp0.supply > 0) {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
391
392 if (_endIndex - _startIndex > 1) {
393 for (uint i = _startIndex; i < _endIndex-1; i++) {
394 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
395 if (sp0.supply > 0) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
394 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
395 if (sp0.supply > 0) {
396 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
397 (uint _reward, uint _endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
398 reward += _reward;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
396 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
397 (uint _reward, uint _endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
398 reward += _reward;
399 _writeRewardPerTokenCheckpoint(token, reward, _endTime);
400 _startTimestamp = _endTime;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
406 if (sp.supply > 0) {
407 (uint _reward,) = _calcRewardPerToken(token, lastTimeRewardApplicable(token), Math.max(sp.timestamp, _startTimestamp), sp.supply, _startTimestamp);
408 reward += _reward;
409 _writeRewardPerTokenCheckpoint(token, reward, block.timestamp);
410 _startTimestamp = block.timestamp;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
421
422 uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
423 uint _endIndex = numCheckpoints[tokenId]-1;
424
425 uint reward = userRewards[token][tokenId];
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
425 uint reward = userRewards[token][tokenId];
426
427 if (_endIndex - _startIndex > 1) {
428 for (uint i = _startIndex; i < _endIndex-1; i++) {
429 Checkpoint memory cp0 = checkpoints[tokenId][i];
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
426
427 if (_endIndex - _startIndex > 1) {
428 for (uint i = _startIndex; i < _endIndex-1; i++) {
429 Checkpoint memory cp0 = checkpoints[tokenId][i];
430 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
426
427 if (_endIndex - _startIndex > 1) {
428 for (uint i = _startIndex; i < _endIndex-1; i++) {
429 Checkpoint memory cp0 = checkpoints[tokenId][i];
430 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
428 for (uint i = _startIndex; i < _endIndex-1; i++) {
429 Checkpoint memory cp0 = checkpoints[tokenId][i];
430 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
431 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
432 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
431 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
432 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
433 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
434 }
435 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
431 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
432 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
433 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
434 }
435 }
```

UNKNOWN

Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
431 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
432 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
433 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
434 }
435 }
```

UNKNOWN

Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
431 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
432 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
433 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
434 }
435 }
```

UNKNOWN

Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
437 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
438 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
439 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
440
441 return reward;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
437 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
438 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
439 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
440
441 return reward;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
437 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
438 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
439 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
440
441 return reward;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
437 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
438 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
439 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
440
441 return reward;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
445 function _deposit(uint amount, uint tokenId) external {
446 require(msg.sender == factory);
447 totalSupply += amount;
448 balanceOf[tokenId] += amount;
449
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
446 require(msg.sender == factory);
447 totalSupply += amount;
448 balanceOf[tokenId] += amount;
449
450 _writeCheckpoint(tokenId, balanceOf[tokenId]);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
456 function _withdraw(uint amount, uint tokenId) external {
457 require(msg.sender == factory);
458 totalSupply -= amount;
459 balanceOf[tokenId] -= amount;
460
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
457 require(msg.sender == factory);
458 totalSupply -= amount;
459 balanceOf[tokenId] -= amount;
460
461 _writeCheckpoint(tokenId, balanceOf[tokenId]);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
471 if (block.timestamp >= periodFinish[token]) {
472 _safeTransferFrom(token, msg.sender, address(this), amount);
473 rewardRate[token] = amount / DURATION;
474 } else {
475 uint _remaining = periodFinish[token] - block.timestamp;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
473 rewardRate[token] = amount / DURATION;
474 } else {
475 uint _remaining = periodFinish[token] - block.timestamp;
476 uint _left = _remaining * rewardRate[token];
477 require(amount > _left);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
474 } else {
475 uint _remaining = periodFinish[token] - block.timestamp;
476 uint _left = _remaining * rewardRate[token];
477 require(amount > _left);
478 _safeTransferFrom(token, msg.sender, address(this), amount);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
477 require(amount > _left);
478 _safeTransferFrom(token, msg.sender, address(this), amount);
479 rewardRate[token] = (amount + _left) / DURATION;
480 }
481 require(rewardRate[token] > 0);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
477 require(amount > _left);
478 _safeTransferFrom(token, msg.sender, address(this), amount);
479 rewardRate[token] = (amount + _left) / DURATION;
480 }
481 require(rewardRate[token] > 0);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

```
481 require(rewardRate[token] > 0);
482 uint balance = erc20(token).balanceOf(address(this));
483 require(rewardRate[token] <= balance / DURATION, "Provided reward too high");
484 periodFinish[token] = block.timestamp + DURATION;
485 if (!isReward[token]) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

voter.sol

Locations

```
482 uint balance = erc20(token).balanceOf(address(this));
483 require(rewardRate[token] <= balance / DURATION, "Provided reward too high");
484 periodFinish[token] = block.timestamp + DURATION;
485 if (!isReward[token]) {
486 isReward[token] = true;
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
559 uint _totalWeigth = 0;
560
561 for (uint i = 0; i < _poolVoteCnt; i ++) {
562 address _pool = _poolVote[i];
563 uint _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

```
565 if (_votes > 0) {
566 _updateFor(gauges[_pool]);
567 _totalWeigth += _votes;
568 weights[_pool] -= _votes;
569 votes[_tokenId][_pool] -= _votes;
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
566 _updateFor(gauges[_pool]);
567 _totalWeigth += _votes;
568 weights[_pool] -= _votes;
569 votes[_tokenId][_pool] -= _votes;
570 Bribe(bribes[gauges[_pool]])._withdraw(_votes, _tokenId);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
567 _totalWeigth += _votes;
568 weights[_pool] -= _votes;
569 votes[_tokenId][_pool] -= _votes;
570 Bribe(bribes[gauges[_pool]])._withdraw(_votes, _tokenId);
571 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

```
572 }
573 }
574 totalWeight -= _totalWeigth;
575 usedWeights[_tokenId] = 0;
576 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

```
582 uint[] memory _weights = new uint[](_poolCnt);
583
584 for (uint i = 0; i < _poolCnt; i ++) {
585 _weights[i] = votes[_tokenId][_poolVote[i]];
586 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
597 uint _usedWeight = 0;
598
599 for (uint i = 0; i < _poolCnt; i ++) {
600 _totalVoteWeight += _weights[i];
601 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
598
599 for (uint i = 0; i < _poolCnt; i ++) {
600 _totalVoteWeight += _weights[i];
601 }
602
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
601 }
602
603 for (uint i = 0; i < _poolCnt; i ++) {
604 address _pool = _poolVote[i];
605 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

```
604 address _pool = _poolVote[i];
605 address _gauge = gauges[_pool];
606 uint _poolWeight = _weights[i] * _weight / _totalVoteWeight;
607
608 if (isGauge[_gauge]) {
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
604 address _pool = _poolVote[i];
605 address _gauge = gauges[_pool];
606 uint _poolWeight = _weights[i] * _weight / _totalVoteWeight;
607
608 if (isGauge[_gauge]) {
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
608 if (isGauge[_gauge]) {
609 _updateFor(_gauge);
610 _usedWeight += _poolWeight;
611 _totalWeight += _poolWeight;
612 weights[_pool] += _poolWeight;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
609 _updateFor(_gauge);
610 _usedWeight += _poolWeight;
611 _totalWeight += _poolWeight;
612 weights[_pool] += _poolWeight;
613 poolVote[_tokenId].push(_pool);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
610 _usedWeight += _poolWeight;
611 _totalWeight += _poolWeight;
612 weights[_pool] += _poolWeight;
613 poolVote[_tokenId].push(_pool);
614 votes[_tokenId][_pool] += _poolWeight;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
612 weights[_pool] += _poolWeight;
613 poolVote[_tokenId].push(_pool);
614 votes[_tokenId][_pool] += _poolWeight;
615 Bribe(bribes[_gauge])._deposit(_poolWeight, _tokenId);
616 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

```
618 }
619 if (_usedWeight > 0) ve(_ve).voting(_tokenId);
620 totalWeight += _totalWeight;
621 usedWeights[_tokenId] = _usedWeight;
622 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

voter.sol

Locations

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

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

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

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
668 uint256 _ratio = amount * 1e18 / totalWeight; // 1e18 adjustment is removed during claim
669 if (_ratio > 0) {
670 index += _ratio;
671 }
672 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

```
674
675 function updateFor(address[] memory _gauges) external {
676 for (uint i = 0; i < _gauges.length; i++) {
677 _updateFor(_gauges[i]);
678 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
680
681 function updateFor(uint start, uint end) public {
682 for (uint i = start; i < end; i++) {
683 _updateFor(gauges[pools[i]]);
684 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

voter.sol

Locations

```
700 uint _index = index; // get global index0 for accumulated distro
701 supplyIndex[_gauge] = _index; // update _gauge current position to global position
702 uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
703 if (_delta > 0) {
704 uint _share = _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

```
702 uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
703 if (_delta > 0) {
704 uint _share = _supplied * _delta / 1e18; // add accrued difference for each supplied token
705 claimable[_gauge] += _share;
706 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

voter.sol

Locations

```
702 uint _delta = _index - _supplyIndex; // see if there is any difference that need to be accrued
703 if (_delta > 0) {
704 uint _share = _supplied * _delta / 1e18; // add accrued difference for each supplied token
705 claimable[_gauge] += _share;
706 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

voter.sol

Locations

```
703 if (_delta > 0) {
704 uint _share = _supplied * _delta / 1e18; // add accrued difference for each supplied token
705 claimable[_gauge] += _share;
706 }
707 } else {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
711
712 function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
713 for (uint i = 0; i < _gauges.length; i ++) {
714 IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
715 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
718 function claimBribes(address[] memory _bribes, address[][] memory _tokens, uint _tokenId) external {
719 require(ve(_ve).isApprovedOrOwner(msg.sender, _tokenId));
720 for (uint i = 0; i < _bribes.length; i ++) {
721 Bribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
722 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
725 function claimFees(address[] memory _fees, address[][] memory _tokens, uint _tokenId) external {
726 require(ve(_ve).isApprovedOrOwner(msg.sender, _tokenId));
727 for (uint i = 0; i < _fees.length; i ++) {
728 Bribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
729 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
731
732 function distributeFees(address[] memory _gauges) external {
733 for (uint i = 0; i < _gauges.length; i ++) {
734 IGauge(_gauges[i]).claimFees();
735 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
756
757 function distribute(uint start, uint finish) public {
758 for (uint x = start; x < finish; x++) {
759 distribute(gauges[pools[x]]);
760 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

voter.sol

Locations

```
762
763 function distribute(address[] memory _gauges) external {
764 for (uint x = 0; x < _gauges.length; x++) {
765 distribute(_gauges[x]);
766 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
145
146 // First check most recent balance
147 if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
148 return (nCheckpoints - 1);
149 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
146 // First check most recent balance
147 if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
148 return (nCheckpoints - 1);
149 }
150
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
155
156 uint lower = 0;
157 uint upper = nCheckpoints - 1;
158 while (upper > lower) {
159 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
164 lower = center;
165 } else {
166 upper = center - 1;
167 }
168 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
177
178 // First check most recent balance
179 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
180 return (nCheckpoints - 1);
181 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
178 // First check most recent balance
179 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
180 return (nCheckpoints - 1);
181 }
182
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
187
188 uint lower = 0;
189 uint upper = nCheckpoints - 1;
190 while (upper > lower) {
191 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
196 lower = center;
197 } else {
198 upper = center - 1;
199 }
200 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
209
210 // First check most recent balance
211 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
212 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
213 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
210 // First check most recent balance
211 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
212 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
213 }
214
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
210 // First check most recent balance
211 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
212 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
213 }
214
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
219
220 uint lower = 0;
221 uint upper = nCheckpoints - 1;
222 while (upper > lower) {
223 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
228 lower = center;
229 } else {
230 upper = center - 1;
231 }
232 }
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
238 uint _nCheckPoints = numCheckpoints[tokenId];
239
240 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
241 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
242 } else {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
239
240 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
241 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
242 } else {
243 checkpoints[tokenId][_nCheckPoints] = Checkpoint(_timestamp, balance);
```

UNKNOWN

Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
249 uint _nCheckPoints = rewardPerTokenNumCheckpoints[token];
250
251 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
252 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
253 } else {
```

UNKNOWN

Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
250
251 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
252 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
253 } else {
254 rewardPerTokenCheckpoints[token][_nCheckPoints] = RewardPerTokenCheckpoint(timestamp, reward);
```

UNKNOWN

Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
261 uint _timestamp = block.timestamp;
262
263 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
264 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
265 } else {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
262
263 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
264 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
265 } else {
266 supplyCheckpoints[_nCheckPoints] = SupplyCheckpoint(_timestamp, totalSupply);
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
330
331 uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
332 uint _endIndex = Math.min(numCheckpoints[tokenId]-1, maxRuns);
333
334 uint reward = userRewards[token][tokenId];
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
358
359 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
360 uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);
361
362 for (uint i = _startIndex; i < _endIndex; i++) {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
388
389 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
390 uint _endIndex = supplyNumCheckpoints-1;
391
392 if (_endIndex - _startIndex > 1) {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
391
392 if (_endIndex - _startIndex > 1) {
393 for (uint i = _startIndex; i < _endIndex-1; i++) {
394 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
395 if (sp0.supply > 0) {
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

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

SWC-101

Source file

voter.sol

Locations

```
421
422 uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
423 uint _endIndex = numCheckpoints[tokenId]-1;
424
425 uint reward = userRewards[token][tokenId];
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

voter.sol

Locations

```
426
427 if (_endIndex - _startIndex > 1) {
428 for (uint i = _startIndex; i < _endIndex-1; i++) {
429 Checkpoint memory cp0 = checkpoints[tokenId][i];
430 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "_unlocked" is internal. Other possible visibility settings are public and private.

Source file

voter.sol

Locations

```
118
119 // simple re-entrancy check
120 uint _unlocked = 1;
121 modifier lock() {
122 require(_unlocked == 1);
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "_unlocked" is internal. Other possible visibility settings are public and private.

Source file

voter.sol

Locations

```
523
524 // simple re-entrancy check
525 uint _unlocked = 1;
526 modifier lock() {
527 require(_unlocked == 1);
```

UNKNOWN

SWC-110

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

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

Source file

voter.sol

Locations

```
70 | mapping(address => mapping(uint => uint)) public userRewards;
71 |
72 | address[] public rewards;
73 | mapping(address => bool) public isReward;
74 |
```

UNKNOWN

SWC-110

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

The public state variable "pools" in "BaseV1Voter" contract has type "address[]" and can cause an exception in case of use of invalid array index value.

Source file

voter.sol

Locations

```
531 | }
532 |
533 | address[] public pools; // all pools viable for incentives
534 | mapping(address => address) public gauges; // pool => gauge
535 | 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 "BaseV1Voter" 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

```
537 | mapping(address => uint) public weights; // pool => weight
538 | mapping(uint => mapping(address => uint)) public votes; // nft => pool => votes
539 | mapping(uint => address[]) public poolVote; // nft => pools
540 | mapping(uint => uint) public usedWeights; // nft => total voting weight of user
541 | 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

```
287 require(ve(_ve).isApprovedOrOwner(msg.sender, tokenId));
288 for (uint i = 0; i < tokens.length; i++) {
289 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
290
291 uint _reward = earned(tokens[i], tokenId);
```

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

```
287 require(ve(_ve).isApprovedOrOwner(msg.sender, tokenId));
288 for (uint i = 0; i < tokens.length; i++) {
289 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
290
291 uint _reward = earned(tokens[i], tokenId);
```

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

```
287 require(ve(_ve).isApprovedOrOwner(msg.sender, tokenId));
288 for (uint i = 0; i < tokens.length; i++) {
289 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
290
291 uint _reward = earned(tokens[i], tokenId);
```

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

```
289 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
290
291 uint _reward = earned(tokens[i], tokenId);
292 userRewards[tokens[i]][tokenId] = 0;
293 lastEarn[tokens[i]][tokenId] = block.timestamp;
```

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

```
290
291 uint _reward = earned(tokens[i], tokenId);
292 userRewards[tokens[i]][tokenId] = 0;
293 lastEarn[tokens[i]][tokenId] = block.timestamp;
294 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
```

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

```
291 uint _reward = earned(tokens[i], tokenId);
292 userRewards[tokens[i]][tokenId] = 0;
293 lastEarn[tokens[i]][tokenId] = block.timestamp;
294 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
295 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
```

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

```
292 userRewards[tokens[i]][tokenId] = 0;
293 lastEarn[tokens[i]][tokenId] = block.timestamp;
294 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
295 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
296
```

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

```
292 userRewards[tokens[i]][tokenId] = 0;
293 lastEarn[tokens[i]][tokenId] = block.timestamp;
294 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
295 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
296
```

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

```
293 lastEarn[tokens[i]][tokenId] = block.timestamp;
294 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
295 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
296
297 emit ClaimRewards(msg.sender, tokens[i], _reward);
```

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

```
295 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
296
297 emit ClaimRewards(msg.sender, tokens[i], _reward);
298 }
299 }
```

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

```
304 address _owner = ve(_ve).ownerOf(tokenId);
305 for (uint i = 0; i < tokens.length; i++) {
306 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
307
308 uint _reward = earned(tokens[i], tokenId);
```

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

```
304 address _owner = ve(_ve).ownerOf(tokenId);
305 for (uint i = 0; i < tokens.length; i++) {
306 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
307
308 uint _reward = earned(tokens[i], tokenId);
```

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

```
304 address _owner = ve(_ve).ownerOf(tokenId);
305 for (uint i = 0; i < tokens.length; i++) {
306 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
307
308 uint _reward = earned(tokens[i], tokenId);
```

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

```
306 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
307
308 uint _reward = earned(tokens[i], tokenId);
309 userRewards[tokens[i]][tokenId] = 0;
310 lastEarn[tokens[i]][tokenId] = block.timestamp;
```

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

```
307
308 uint _reward = earned(tokens[i], tokenId);
309 userRewards[tokens[i]][tokenId] = 0;
310 lastEarn[tokens[i]][tokenId] = block.timestamp;
311 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
```

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

```
308 uint _reward = earned(tokens[i], tokenId);
309 userRewards[tokens[i]][tokenId] = 0;
310 lastEarn[tokens[i]][tokenId] = block.timestamp;
311 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
312 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
```

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

```
309 userRewards[tokens[i]][tokenId] = 0;
310 lastEarn[tokens[i]][tokenId] = block.timestamp;
311 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
312 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
313
```

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

```
309 userRewards[tokens[i]][tokenId] = 0;
310 lastEarn[tokens[i]][tokenId] = block.timestamp;
311 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
312 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
313
```

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

```
310 lastEarn[tokens[i]][tokenId] = block.timestamp;
311 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
312 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
313
314 emit ClaimRewards(_owner, tokens[i], _reward);
```

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 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
313
314 emit ClaimRewards(_owner, tokens[i], _reward);
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

```
560
561 for (uint i = 0; i < _poolVoteCnt; i ++) {
562 address _pool = _poolVote[i];
563 uint _votes = votes[_tokenId][_pool];
564
```

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

```
583
584 for (uint i = 0; i < _poolCnt; i ++) {
585 _weights[i] = votes[_tokenId][_poolVote[i]];
586 }
587
```

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

```
583
584 for (uint i = 0; i < _poolCnt; i ++) {
585 _weights[i] = votes[_tokenId][_poolVote[i]];
586 }
587
```

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

```
598
599 for (uint i = 0; i < _poolCnt; i ++) {
600 _totalVoteWeight += _weights[i];
601 }
602
```

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

```
602
603 for (uint i = 0; i < _poolCnt; i ++) {
604 address _pool = _poolVote[i];
605 address _gauge = gauges[_pool];
606 uint _poolWeight = _weights[i] * _weight / _totalVoteWeight;
```

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

```
604 address _pool = _poolVote[i];
605 address _gauge = gauges[_pool];
606 uint _poolWeight = _weights[i] * _weight / _totalVoteWeight;
607
608 if (isGauge[_gauge]) {
```

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

```
675 function updateFor(address[] memory _gauges) external {
676 for (uint i = 0; i < _gauges.length; i++) {
677 _updateFor(_gauges[i]);
678 }
679 }
```

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

```
681 function updateFor(uint start, uint end) public {
682 for (uint i = start; i < end; i++) {
683 _updateFor(gauges[pools[i]]);
684 }
685 }
```

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

```
712 function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
713 for (uint i = 0; i < _gauges.length; i ++) {
714 IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
715 }
716 }
```

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

```
712 function claimRewards(address[] memory _gauges, address[][] memory _tokens) external {
713 for (uint i = 0; i < _gauges.length; i ++) {
714 IGauge(_gauges[i]).getReward(msg.sender, _tokens[i]);
715 }
716 }
```

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

```
719 require(ve(_ve).isApprovedOrOwner(msg.sender, _tokenId));
720 for (uint i = 0; i < _bribes.length; i ++) {
721 Bribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
722 }
723 }
```

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

```
719 require(ve(_ve).isApprovedOrOwner(msg.sender, _tokenId));
720 for (uint i = 0; i < _bribes.length; i ++) {
721 Bribe(_bribes[i]).getRewardForOwner(_tokenId, _tokens[i]);
722 }
723 }
```

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

```
726 require(ve(_ve).isApprovedOrOwner(msg.sender, _tokenId));
727 for (uint i = 0; i < _fees.length; i ++) {
728 Bribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
729 }
730 }
```

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

```
726 require(ve(_ve).isApprovedOrOwner(msg.sender, _tokenId));
727 for (uint i = 0; i < _fees.length; i ++) {
728 Bribe(_fees[i]).getRewardForOwner(_tokenId, _tokens[i]);
729 }
730 }
```

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

```
732 function distributeFees(address[] memory _gauges) external {
733 for (uint i = 0; i < _gauges.length; i ++) {
734 IGauge(_gauges[i]).claimFees();
735 }
736 }
```

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

```
757 function distribute(uint start, uint finish) public {
758 for (uint x = start; x < finish; x++) {
759 distribute(gauges[pools[x]]);
760 }
761 }
```

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

```
763 function distribute(address[] memory _gauges) external {
764 for (uint x = 0; x < _gauges.length; x++) {
765 distribute(_gauges[x]);
766 }
767 }
```