MythX

REPORT 628D679DCEFD55001961E4FD

Created	Tue May 24 2022 23:17:49 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	6197960e3494e9c8c076e89b

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
3a0fc266-5f84-48f2-af44-f07996293b58	Bribe.sol	0

Analysis 3a0fc266-5f84-48f2-af44-f07996293b58 MythX

Started	Tue May 24 2022 23:17:59 GMT+0000 (Coordinated Universal Time)
Finished	Tue May 24 2022 23:18:04 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Bribe.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

Bribe.sol

Locations

```
14
15  uint public constant DURATION = 7 days; // rewards are released over 7 days
16  uint public constant PRECISION = 10 ** 18;
17
18  // default snx staking contract implementation
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
95
96  // First check most recent balance
97  if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
98  return (nCheckpoints - 1);
99  }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
96  // First check most recent balance
97  if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
98  return (nCheckpoints - 1);
99  }
100
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
105
106 uint lower = 0;
107 uint upper = nCheckpoints - 1;
108 while (upper > lower) {
109 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

Bribe.sol

Locations

```
107 uint upper = nCheckpoints - 1;
108 while (upper > lower) {
109 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
110 Checkpoint memory cp = checkpoints[tokenId][center];
111 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
107 uint upper = nCheckpoints - 1;
108 while (upper > lower) {
109 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
110 Checkpoint memory cp = checkpoints[tokenId][center];
111 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
107 uint upper = nCheckpoints - 1;
108 while (upper > lower) {
109 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
110 Checkpoint memory cp = checkpoints[tokenId][center];
111 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
114 lower = center;
115 } else {
116 upper = center - 1;
117 }
118 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
127
128 // First check most recent balance
129 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
130 return (nCheckpoints - 1);
131 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
128 // First check most recent balance
129 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
130 return (nCheckpoints - 1);
131 }
132
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
137
138 uint lower = 0;
139 uint upper = nCheckpoints - 1;
140 while (upper > lower) {
141 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

Bribe.sol

Locations

```
139 uint upper = nCheckpoints - 1;
140 while (upper > lower) {
141 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
142 SupplyCheckpoint memory cp = supplyCheckpoints[center];
143 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
139 uint upper = nCheckpoints - 1;
140 while (upper > lower) {
141 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
142 SupplyCheckpoint memory cp = supplyCheckpoints[center];
143 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
139 uint upper = nCheckpoints - 1;
140 while (upper > lower) {
141 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
142 SupplyCheckpoint memory cp = supplyCheckpoints[center];
143 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
146 lower = center;
147 } else {
148 upper = center - 1;
149 }
150 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
159
160 // First check most recent balance
161 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
162 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
163 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
160 // First check most recent balance
161 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
162 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
163 }
164
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
160 // First check most recent balance
161 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
162 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
163 }
164
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
169
170 uint lower = 0;
171 uint upper = nCheckpoints - 1;
172 while (upper > lower) {
173 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

Bribe.sol

Locations

```
171 uint upper = nCheckpoints - 1;
172 while (upper > lower) {
173 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
174 RewardPerTokenCheckpoint memory cp = rewardPerTokenCheckpoints[token][center];
175 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
171 uint upper = nCheckpoints - 1;
172 while (upper > lower) {
173 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
174 RewardPerTokenCheckpoint memory cp = rewardPerTokenCheckpoints[token][center];
175 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
171 uint upper = nCheckpoints - 1;
172 while (upper > lower) {
173 uint center = upper - (upper - lower) / 2; // ceil, avoiding overflow
174 RewardPerTokenCheckpoint memory cp = rewardPerTokenCheckpoints[token][center];
175 if (cp.timestamp == timestamp) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
178 lower = center;
179 } else {
180 upper = center - 1;
181 }
182 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
188 uint _nCheckPoints = numCheckpoints[tokenId];
189
190 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
191 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
192 } else {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
189
190 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
191 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
192 } else {
193 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

Bribe.sol

Locations

```
192 } else {
193 checkpoints[tokenId][_nCheckPoints] = Checkpoint(_timestamp, balance);
194 numCheckpoints[tokenId] = _nCheckPoints + 1;
195 }
196 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
199 uint _nCheckPoints = rewardPerTokenNumCheckpoints[token];
200
201 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
202 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
203 } else {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
200
201 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
202 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
203 } else {
204 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

Bribe.sol

Locations

```
203 } else {
204 rewardPerTokenCheckpoints[token][_nCheckPoints] = RewardPerTokenCheckpoint(timestamp, reward);
205 rewardPerTokenNumCheckpoints[token] = _nCheckPoints + 1;
206 }
207 }
```

UNKNOWN

Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
211 uint _timestamp = block.timestamp;
212
213 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
214 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
215 } else {
```

UNKNOWN

Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
212
213 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
214 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
215 } else {
216 supplyCheckpoints[_nCheckPoints] = SupplyCheckpoint(_timestamp, totalSupply);
```

UNKNOWN

Arithmetic operation "+" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
215 } else {
216 supplyCheckpoints[_nCheckPoints] = SupplyCheckpoint(_timestamp, totalSupply);
217 supplyNumCheckpoints = _nCheckPoints + 1;
218 }
219 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
231 function getReward(uint tokenId, address[] memory tokens) external lock {
232 require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, tokenId));
233 for (uint i = 0; i < tokens.length; i++) {
234 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
235
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
247 require(msg.sender == factory);
248 address _owner = IVotingEscrow(_ve).ownerOf(tokenId);
249 for (uint i = 0; i < tokens.length; i++) {
250 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
251
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
263 return rewardPerTokenStored[token];
264 }
265 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
266 }
267
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
263 return rewardPerTokenStored[token];
264 }
265 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
266 }
267
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
263 return rewardPerTokenStored[token];
264 }
265 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
266 }
267
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
263 return rewardPerTokenStored[token];
264 }
265 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
266 }
267
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
263 return rewardPerTokenStored[token];
264 }
265 return rewardPerTokenStored[token] + ((lastTimeRewardApplicable(token) - Math.min(lastUpdateTime[token], periodFinish[token])) * rewardRate[token] * PRECISION / totalSupply);
266 }
267
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
283
284 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
285 uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);
286
287 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

Bribe.sol

Locations

```
285 uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);
286
287 for (uint i = _startIndex; i < _endIndex; i++) {
288 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
289 if (sp0.supply > 0) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
288 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
289 if (sp0.supply > 0) {
290 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
291 (uint _reward, uint endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
292 reward += _reward;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
290 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
291 (uint _reward, uint endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
292 reward += _reward;
293 _writeRewardPerTokenCheckpoint(token, reward, endTime);
294 _startTimestamp = endTime;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
301 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
302 uint endTime = Math.max(timestamp1, startTimestamp);
303 return (((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
304 }
305
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
301 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
302 uint endTime = Math.max(timestamp1, startTimestamp);
303 return (((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
304 }
305
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
301 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
302 uint endTime = Math.max(timestamp1, startTimestamp);
303 return (((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
304 }
305
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
301 function _calcRewardPerToken(address token, uint timestamp1, uint timestamp0, uint supply, uint startTimestamp) internal view returns (uint, uint) {
302 uint endTime = Math.max(timestamp1, startTimestamp);
303 return (((Math.min(endTime, periodFinish[token]) - Math.min(Math.max(timestamp0, startTimestamp), periodFinish[token])) * rewardRate[token] * PRECISION / supply), endTime);
304 }
305
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
317
318 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
319 uint _endIndex = supplyNumCheckpoints-1;
320
321 if (_endIndex - _startIndex > 1) {
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
319 uint _endIndex = supplyNumCheckpoints-1;
320
321 if (_endIndex - _startIndex > 1) {
322 for (uint i = _startIndex; i < _endIndex-1; i++) {
323 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
320
321 if (_endIndex - _startIndex > 1) {
322 for (uint i = _startIndex; i < _endIndex-1; i++) {
323 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
324 if (sp0.supply > 0) {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
320
321 if (_endIndex - _startIndex > 1) {
322 for (uint i = _startIndex; i < _endIndex-1; i++) {
323 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
324 if (sp0.supply > 0) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
323 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
324 if (sp0.supply > 0) {
325 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
326 (uint _reward, uint _endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
327 reward += _reward;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
325 SupplyCheckpoint memory sp1 = supplyCheckpoints[i+1];
326 (uint _reward, uint _endTime) = _calcRewardPerToken(token, sp1.timestamp, sp0.timestamp, sp0.supply, _startTimestamp);
327 reward += _reward;
328 _writeRewardPerTokenCheckpoint(token, reward, _endTime);
329 _startTimestamp = _endTime;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
335 if (sp.supply > 0) {
336 (uint _reward,) = _calcRewardPerToken(token, lastTimeRewardApplicable(token), Math.max(sp.timestamp, _startTimestamp), sp.supply, _startTimestamp);
337 reward += _reward;
338 _writeRewardPerTokenCheckpoint(token, reward, block.timestamp);
339 _startTimestamp = block.timestamp;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
350
351 uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
352 uint _endIndex = numCheckpoints[tokenId]-1;
353
354 uint reward = 0;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
354 uint reward = 0;
355
356 if (_endIndex - _startIndex > 1) {
357 for (uint i = _startIndex; i < _endIndex-1; i++) {
358 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

Bribe.sol

Locations

```
355
356 if (_endIndex - _startIndex > 1) {
357 for (uint i = _startIndex; i < _endIndex-1; i++) {
358 Checkpoint memory cp0 = checkpoints[tokenId][i];
359 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

Bribe.sol

Locations

```
355
356 if (_endIndex - _startIndex > 1) {
357 for (uint i = _startIndex; i < _endIndex-1; i++) {
358 Checkpoint memory cp0 = checkpoints[tokenId][i];
359 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

Bribe.sol

Locations

```
357 for (uint i = _startIndex; i < _endIndex-1; i++) {
358 Checkpoint memory cp0 = checkpoints[tokenId][i];
359 Checkpoint memory cp1 = checkpoints[tokenId][i+1];
360 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
361 (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

Bribe.sol

Locations

```
360 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
361 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
362 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
363 }
364 }
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
360 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
361 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
362 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
363 }
364 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
360 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
361 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
362 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
363 }
364 }
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
360 (uint _rewardPerTokenStored0,) = getPriorRewardPerToken(token, cp0.timestamp);
361 (uint _rewardPerTokenStored1,) = getPriorRewardPerToken(token, cp1.timestamp);
362 reward += cp0.balanceOf * (_rewardPerTokenStored1 - _rewardPerTokenStored0) / PRECISION;
363 }
364 }
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
366 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
367 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
368 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
369
370 return reward;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
366 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
367 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
368 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
369
370 return reward;
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
366 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
367 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
368 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
369
370 return reward;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
366 Checkpoint memory cp = checkpoints[tokenId][_endIndex];
367 (uint _rewardPerTokenStored,) = getPriorRewardPerToken(token, cp.timestamp);
368 reward += cp.balanceOf * (rewardPerToken(token) - Math.max(_rewardPerTokenStored, userRewardPerTokenStored[token][tokenId])) / PRECISION;
369
370 return reward;
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
374 function _deposit(uint amount, uint tokenId) external {
375 require(msg.sender == factory);
376 totalSupply += amount;
377 balanceOf[tokenId] += amount;
378
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
375 require(msg.sender == factory);
376 totalSupply += amount;
377 balanceOf[tokenId] += amount;
378
379 _writeCheckpoint(tokenId, balanceOf[tokenId]);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
385 function _withdraw(uint amount, uint tokenId) external {
386 require(msg.sender == factory);
387 totalSupply -= amount;
388 balanceOf[tokenId] -= amount;
389
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
386 require(msg.sender == factory);
387 totalSupply -= amount;
388 balanceOf[tokenId] -= amount;
389
390 _writeCheckpoint(tokenId, balanceOf[tokenId]);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
396 function left(address token) external view returns (uint) {
397 if (block.timestamp >= periodFinish[token]) return 0;
398 uint _remaining = periodFinish[token] - block.timestamp;
399 return _remaining * rewardRate[token];
400 }
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
397 if (block.timestamp >= periodFinish[token]) return 0;
398 uint _remaining = periodFinish[token] - block.timestamp;
399 return _remaining * rewardRate[token];
400 }
401
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
408 if (block.timestamp >= periodFinish[token]) {
409 _safeTransferFrom(token, msg.sender, address(this), amount);
410 rewardRate[token] = amount / DURATION;
411 } else {
412 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

Bribe.sol

Locations

```
410 rewardRate[token] = amount / DURATION;
411 } else {
412 uint _remaining = periodFinish[token] - block.timestamp;
413 uint _left = _remaining * rewardRate[token];
414 require(amount > _left);
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
411 } else {
412 uint _remaining = periodFinish[token] - block.timestamp;
413 uint _left = _remaining * rewardRate[token];
414 require(amount > _left);
415 _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

Bribe.sol

Locations

```
414 require(amount > _left);
415 _safeTransferFrom(token, msg.sender, address(this), amount);
416 rewardRate[token] = (amount + _left) / DURATION;
417 }
418 require(rewardRate[token] > 0);
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
414 require(amount > _left);
415 _safeTransferFrom(token, msg.sender, address(this), amount);
416 rewardRate[token] = (amount + _left) / DURATION;
417 }
418 require(rewardRate[token] > 0);
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
418 require(rewardRate[token] > 0);
419 uint balance = IERC20(token).balanceOf(address(this));
420 require(rewardRate[token] <= balance / DURATION, "Provided reward too high");
421 periodFinish[token] = block.timestamp + DURATION;
422 if (!isReward[token]) {
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

Bribe.sol

Locations

```
419 uint balance = IERC20(token).balanceOf(address(this));
420 require(rewardRate[token] <= balance / DURATION, "Provided reward too high");
421 periodFinish[token] = block.timestamp + DURATION;
422 if (!isReward[token]) {
423 isReward[token] = true;
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
95
96  // First check most recent balance
97  if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
98  return (nCheckpoints - 1);
99  }
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
96  // First check most recent balance
97  if (checkpoints[tokenId][nCheckpoints - 1].timestamp <= timestamp) {
98  return (nCheckpoints - 1);
99  }
100
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
105
106 uint lower = 0;
107 uint upper = nCheckpoints - 1;
108 while (upper > lower) {
109 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

Bribe.sol

Locations

```
114 lower = center;
115 } else {
116 upper = center - 1;
117 }
118 }
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
127
128 // First check most recent balance
129 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
130 return (nCheckpoints - 1);
131 }
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
128 // First check most recent balance
129 if (supplyCheckpoints[nCheckpoints - 1].timestamp <= timestamp) {
130 return (nCheckpoints - 1);
131 }
132
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
137
138 uint lower = 0;
139 uint upper = nCheckpoints - 1;
140 while (upper > lower) {
141 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

Bribe.sol

Locations

```
146 lower = center;
147 } else {
148 upper = center - 1;
149 }
150 }
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
159
160 // First check most recent balance
161 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
162 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
163 }
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
160 // First check most recent balance
161 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
162 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
163 }
164
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
160 // First check most recent balance
161 if (rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp <= timestamp) {
162 return (rewardPerTokenCheckpoints[token][nCheckpoints - 1].rewardPerToken, rewardPerTokenCheckpoints[token][nCheckpoints - 1].timestamp);
163 }
164
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
169
170 uint lower = 0;
171 uint upper = nCheckpoints - 1;
172 while (upper > lower) {
173 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

Bribe.sol

Locations

```
178 lower = center;
179 } else {
180 upper = center - 1;
181 }
182 }
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
188 uint _nCheckPoints = numCheckpoints[tokenId];
189
190 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
191 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
192 } else {
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
189
190 if (_nCheckPoints > 0 && checkpoints[tokenId][_nCheckPoints - 1].timestamp == _timestamp) {
191 checkpoints[tokenId][_nCheckPoints - 1].balanceOf = balance;
192 } else {
193 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

Bribe.sol

Locations

```
199 uint _nCheckPoints = rewardPerTokenNumCheckpoints[token];
200
201 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
202 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
203 } else {
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
200
201 if (_nCheckPoints > 0 && rewardPerTokenCheckpoints[token][_nCheckPoints - 1].timestamp == timestamp) {
202 rewardPerTokenCheckpoints[token][_nCheckPoints - 1].rewardPerToken = reward;
203 } else {
204 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

Bribe.sol

Locations

```
211 uint _timestamp = block.timestamp;
212
213 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
214 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
215 } else {
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
212
213 if (_nCheckPoints > 0 && supplyCheckpoints[_nCheckPoints - 1].timestamp == _timestamp) {
214 supplyCheckpoints[_nCheckPoints - 1].supply = totalSupply;
215 } else {
216 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

Bribe.sol

Locations

```
283
284 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
285 uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);
286
287 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

Bribe.sol

Locations

```
317
318 uint _startIndex = getPriorSupplyIndex(_startTimestamp);
319 uint _endIndex = supplyNumCheckpoints-1;
320
321 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

Bribe.sol

Locations

```
320
321 if (_endIndex - _startIndex > 1) {
322 for (uint i = _startIndex; i < _endIndex-1; i++) {
323 SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
324 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

Bribe.sol

Locations

```
350
351 uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
352 uint _endIndex = numCheckpoints[tokenId]-1;
353
354 uint reward = 0;
```

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

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

SWC-101

Source file

Bribe.sol

Locations

```
355
356 if (_endIndex - _startIndex > 1) {
357 for (uint i = _startIndex; i < _endIndex-1; i++) {
358 Checkpoint memory cp0 = checkpoints[tokenId][i];
359 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

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 = 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

```
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 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.

SWC-110

Source file

Bribe.sol

Locations

```
25 mapping(address => mapping(uint => uint)) public userRewardPerTokenStored;
26
27 address[] public rewards;
28 mapping(address => bool) public isReward;
29
```

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

Bribe.sol

Locations

```
232 require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, tokenId));
233 for (uint i = 0; i < tokens.length; i++) {
234 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
235
236 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

Bribe.sol

Locations

```
232 require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, tokenId));
233 for (uint i = 0; i < tokens.length; i++) {
234 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
235
236 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

Bribe.sol

Locations

```
232 require(IVotingEscrow(_ve).isApprovedOrOwner(msg.sender, tokenId));
233 for (uint i = 0; i < tokens.length; i++) {
234 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
235
236 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

Bribe.sol

Locations

```
234 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
235
236 uint _reward = earned(tokens[i], tokenId);
237 lastEarn[tokens[i]][tokenId] = block.timestamp;
238 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

Bribe.sol

Locations

```
235
236 uint _reward = earned(tokens[i], tokenId);
237 lastEarn[tokens[i]][tokenId] = block.timestamp;
238 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
239 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

Bribe.sol

Locations

```
236 uint _reward = earned(tokens[i], tokenId);
237 lastEarn[tokens[i]][tokenId] = block.timestamp;
238 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
239 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
240
```

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

Bribe.sol

Locations

```
236 uint _reward = earned(tokens[i], tokenId);
237 lastEarn[tokens[i]][tokenId] = block.timestamp;
238 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
239 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
240
```

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

Bribe.sol

Locations

```
237 lastEarn[tokens[i]][tokenId] = block.timestamp;
238 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
239 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
240
241 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

Bribe.sol

Locations

```
239 if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);
240
241 emit ClaimRewards(msg.sender, tokens[i], _reward);
242 }
243 }
```

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

Bribe.sol

Locations

```
248 address _owner = IVotingEscrow(_ve).ownerOf(tokenId);
249 for (uint i = 0; i < tokens.length; i++) {
250 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
251
252 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

Bribe.sol

Locations

```
248 address _owner = IVotingEscrow(_ve).ownerOf(tokenId);
249 for (uint i = 0; i < tokens.length; i++) {
250 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
251
252 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

Bribe.sol

Locations

```
248 address _owner = IVotingEscrow(_ve).ownerOf(tokenId);
249 for (uint i = 0; i < tokens.length; i++) {
250 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
251
252 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

Bribe.sol

Locations

```
250 (rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);
251
252 uint _reward = earned(tokens[i], tokenId);
253 lastEarn[tokens[i]][tokenId] = block.timestamp;
254 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

Bribe.sol

Locations

```
251
252 uint _reward = earned(tokens[i], tokenId);
253 lastEarn[tokens[i]][tokenId] = block.timestamp;
254 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
255 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

Bribe.sol

Locations

```
252 uint _reward = earned(tokens[i], tokenId);
253 lastEarn[tokens[i]][tokenId] = block.timestamp;
254 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
255 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
256
```

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

Bribe.sol

Locations

```
252 uint _reward = earned(tokens[i], tokenId);
253 lastEarn[tokens[i]][tokenId] = block.timestamp;
254 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
255 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
256
```

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

Bribe.sol

Locations

```
253 lastEarn[tokens[i]][tokenId] = block.timestamp;
254 userRewardPerTokenStored[tokens[i]][tokenId] = rewardPerTokenStored[tokens[i]];
255 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
256
257 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

Bribe.sol

Locations

```
255 if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);
256
257 emit ClaimRewards(_owner, tokens[i], _reward);
258 }
259 }
```