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 SWC-101 Arithmetic operation "**" discovered

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

Source file

Bribe.sol

Locations

```

uint public constant DURATION = 7 days; // rewards are released over 7 days
uint public constant PRECISION = 10 ** 18;

// default snx staking contract implementation
```

UNKNOWN SWC-101 Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```
lower = center;
} else {
upper = center - 1;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```
lower = center;
} else {
upper = center - 1;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```
lower = center;
} else {
upper = center - 1;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```
uint _nCheckPoints = numCheckpoints[tokenId];

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```
uint _nCheckPoints = rewardPerTokenNumCheckpoints[token];

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```
uint _timestamp = block.timestamp;

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`Bribe.sol`

Locations

```
} else {
supplyCheckpoints[_nCheckPoints] = SupplyCheckpoint(_timestamp, totalSupply);
supplyNumCheckpoints = _nCheckPoints + 1;
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```

uint _startIndex = getPriorSupplyIndex(_startTimestamp);
uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);

for (uint i = _startIndex; i < _endIndex; i++) {
```

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Bribe.sol

Locations

```
uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);

for (uint i = _startIndex; i < _endIndex; i++) {
SupplyCheckpoint memory sp0 = supplyCheckpoints[i];
if (sp0.supply > 0) {
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```

uint _startIndex = getPriorSupplyIndex(_startTimestamp);
uint _endIndex = supplyNumCheckpoints-1;

if (_endIndex - _startIndex > 1) {
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```
uint _endIndex = supplyNumCheckpoints-1;

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```

uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
uint _endIndex = numCheckpoints[tokenId]-1;

uint reward = 0;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

```
uint reward = 0;

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "++" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

`Bribe.sol`

Locations

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

return reward;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`Bribe.sol`

Locations

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

return reward;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

Bribe.sol

Locations

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

return reward;
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

return reward;
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

Bribe.sol

Locations

```
require(msg.sender == factory);
totalSupply += amount;
balanceOf[tokenId] += amount;

_writeCheckpoint(tokenId, balanceOf[tokenId]);
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

Bribe.sol

Locations

```
require(msg.sender == factory);
totalSupply -= amount;
balanceOf[tokenId] -= amount;

_writeCheckpoint(tokenId, balanceOf[tokenId]);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

Bribe.sol

Locations

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```
lower = center;
} else {
upper = center - 1;
}
}
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```
lower = center;
} else {
upper = center - 1;
}
}
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```
lower = center;
} else {
upper = center - 1;
}
}
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```
uint _nCheckPoints = numCheckpoints[tokenId];

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN
SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```
uint _nCheckPoints = rewardPerTokenNumCheckpoints[token];

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

UNKNOWN
SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN
SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```
uint _timestamp = block.timestamp;

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```

uint _startIndex = getPriorSupplyIndex(_startTimestamp);
uint _endIndex = Math.min(supplyNumCheckpoints-1, maxRuns);

for (uint i = _startIndex; i < _endIndex; i++) {
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```

uint _startIndex = getPriorSupplyIndex(_startTimestamp);
uint _endIndex = supplyNumCheckpoints-1;

if (_endIndex - _startIndex > 1) {
```

UNKNOWN SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

```
uint _startIndex = getPriorBalanceIndex(tokenId, _startTimestamp);
uint _endIndex = numCheckpoints[tokenId]-1;

uint reward = 0;
```

UNKNOWN SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

`Bribe.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`libraries/Math.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`libraries/Math.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

`libraries/Math.sol`

Locations

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

UNKNOWN
SWC-101

Arithmetic operation "+" discovered

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

Source file
`libraries/Math.sol`

Locations

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

UNKNOWN
SWC-101

Arithmetic operation "/" discovered

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

Source file
`libraries/Math.sol`

Locations

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

UNKNOWN
SWC-101

Arithmetic operation "+" discovered

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

Source file
`libraries/Math.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`libraries/Math.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

`libraries/Math.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

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

Source file

`libraries/Math.sol`

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

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

Source file

libraries/Math.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

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

Source file

libraries/Math.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

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

Source file

libraries/Math.sol

Locations

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

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

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

Source file

libraries/Math.sol

Locations

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

UNKNOWN

SWC-110

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

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

Source file

Bribe.sol

Locations

```
mapping(address => mapping(uint => uint)) public userRewardPerTokenStored;

address[] public rewards;
mapping(address => bool) public isReward;

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

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

uint _reward = earned(tokens[i], tokenId);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

uint _reward = earned(tokens[i], tokenId);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

uint _reward = earned(tokens[i], tokenId);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

```
(rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);

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

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

```

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

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

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

emit ClaimRewards(msg.sender, tokens[i], _reward);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

```
if (_reward > 0) _safeTransfer(tokens[i], msg.sender, _reward);

emit ClaimRewards(msg.sender, tokens[i], _reward);
}
}
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

Bribe.sol

Locations

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

uint _reward = earned(tokens[i], tokenId);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

uint _reward = earned(tokens[i], tokenId);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

uint _reward = earned(tokens[i], tokenId);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

```
(rewardPerTokenStored[tokens[i]], lastUpdateTime[tokens[i]]) = _updateRewardPerToken(tokens[i]);

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

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

```

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

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

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

emit ClaimRewards(_owner, tokens[i], _reward);
```

UNKNOWN

SWC-110

Out of bounds array access

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

Source file

`Bribe.sol`

Locations

```
if (_reward > 0) _safeTransfer(tokens[i], _owner, _reward);

emit ClaimRewards(_owner, tokens[i], _reward);
}
}
```