MythX

REPORT 6239F0A070A9630019EF7160

Created	Tue Mar 22 2022 15:52:00 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	61955db4c8c2c714bb27c662

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
3c0d9406-3d12-4572-b53b-223055aec6fd	/contracts/royaltymodule.sol	1

Analysis 3c0d9406-3d12-4572-b53b-223055aec6fd MythX

Started

Finished Tue Mar 22 2022 15:52:00 GMT+0000 (Coordinated Universal Time)

Mode Deep

Client Tool Mythx-Vscode-Extension

Main Source File /Contracts/Royaltymodule.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
28  require(royaltySplitTT < 10000, 'Royalty Split to TT is > 100%'); //new v1.3
29  require(royaltySplitTT + minRoyaltySplit < 10000, 'Royalty Split to TT + Minimal Split is > 100%');
30  require(ttAddress != address(0), 'Zero Address cannot be TT roaylty account');
31  _ttAddress = ttAddress;
32  _royaltySplitTT = royaltySplitTT;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
37  function updateRAccountLimits(uint256 maxSubAccounts, uint256 minRoyaltySplit) public virtual onlyOwner returns (bool) {
38  require(_royaltySplitTT + minRoyaltySplit < 10000, 'Royalty Split to TT + Minimal Split is > 100%');
39  _maxSubAccount = maxSubAccounts;
40  _minRoyaltySplit = minRoyaltySplit;
41  return true;
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
71 uint256 newSum;
72 for (uint256 i = 0; i < affectedSubaccounts.length; i++) {
73 require(affectedSubaccounts[i].royaltySplit >= _minRoyaltySplit, 'Royalty Split is smaller then set limit');
74 newSum += affectedSubaccounts[i].royaltySplit;
75 (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
73 require(affectedSubaccounts[i].royaltySplit >= _minRoyaltySplit, 'Royalty Split is smaller then set limit');
74 newSum += affectedSubaccounts[i].royaltySplit;
75 (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
76 if (found) {
77 RASubAccount storage foundAcc = _royaltysubaccounts[royaltyAccount][indexOld];
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
78 oldSum += foundAcc.royaltySplit;
79 //Check rights to decrease royalty split
80 if (affectedSubaccounts[i].royaltySplit < foundAcc.royaltySplit) {
81 if (foundAcc.isIndividual) {
82 require(affectedSubaccounts[i].accountId == sender, 'Only individual subaccount owner can decrease royaltySplit');
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
95  //Update royalty split for subaccounts and add new subaccounts
96  for (uint256 i = 0; i < affectedSubaccounts.length; i++) {
97  (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
98  if (found) {
99  _royaltysubaccounts[royaltyAccount][indexOld].royaltySplit = affectedSubaccounts[i].royaltySplit;
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
110 for (uint256 i = 0; i < _royaltysubaccounts[royaltyAccount].length; i++) {
111 if (_royaltysubaccounts[royaltyAccount][i].isIndividual) {
112 require(_royaltysubaccounts[royaltyAccount][i].royaltyBalance == 0, "Can't delete non empty royalty account");
113 }
114 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
128
129 require(_royaltySplitTT + royaltySplitForItsChildren <= 10000, 'Royalty Splits sum is > 100%');
130 address raAccountId = address(bytes20(keccak256(abi.encodePacked(tokenId, to, block.number))));
131 if (parentTokenId == 0) {
132 //Create Royalty account without parent
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
135 _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: 10000 - _royaltySplitTT, royaltyBalance: 0, accountId: to}));
136
137 //create the RA subaccount for TreeTrunk ... new v1.3
138 _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: _royaltySplitTT, royaltyBalance: 0, accountId: _ttAddress}));
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
151 //create the RA subaccount for the to address
152 _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: 10000 - parentRA.royaltySplitForItsChildren - _royaltySplitTT, royaltyBalance: 0, accountId:
153 to}));
154
155 //create the RA subaccount for TreeTrunk ... new v1.3
    _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: _royaltySplitTT, royaltyBalance: 0, accountId: _ttAddress}));
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
151 //create the RA subaccount for the to address
152 _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: 10000 - parentRA.royaltySplitForItsChildren - _royaltySplitTT, royaltyBalance: 0, accountId:
153 to}));
154
155 //create the RA subaccount for TreeTrunk ... new v1.3
    _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: _royaltySplitTT, royaltyBalance: 0, accountId: _ttAddress}));
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
184 for (uint256 i = 0; i < _royaltysubaccounts[royaltyAccount].length; i++) {
185 //skip calculate for 0% subaccounts
186 if (_royaltysubaccounts[royaltyAccount][i].royaltySplit == 0) continue;
187 //calculate royalty split sum
188 uint256 paymentSplit = mulDiv(remainsValue, _royaltysubaccounts[royaltyAccount][i].royaltySplit, remainsSplit);
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
188 uint256 paymentSplit = mulDiv(remainsValue, _royaltysubaccounts[royaltyAccount][i].royaltySplit, remainsSplit);
189 remainsValue -= paymentSplit;
190 remainsSplit -= _royaltysubaccounts[royaltyAccount][i].royaltySplit;
191 //distribute if IND subaccount
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
193 _royaltysubaccounts[royaltyAccount][i].royaltyBalance += paymentSplit;
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
189 remainsValue -= paymentSplit;
190 remainsSplit -= _royaltysubaccounts[royaltyAccount][i].royaltySplit;
191 //distribute if IND subaccount
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
193 _royaltysubaccounts[royaltyAccount][i].royaltyBalance += paymentSplit;
194 emit RoyalyDistributed(tokenId, _royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, assetId);
```

UNKNOWN Arithmetic operation "+=" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
193 _royaltysubaccounts[royaltyAccount][i].royaltyBalance += paymentSplit;
194 emit RoyalyDistributed(tokenId, _royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, assetId);
195 }
196 //distribute if RA subaccounts
```

UNKNOWN Arithmetic operation "-=" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
251 }
252
253 //https://hackmd.io/@o70I-dRsSdeopRewqTLbnw/r1NDumcBt#Update-RA-ownership-with-payout-to-approved-address-from
254 //Used in RoyaltyBearingToken._safeTransferFrom(address, address,uint256, bytes)
255 //for transfer royalty account ownership after tranfer token ownership
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
275 if (subAccounts[i].accountId == subaccount) {
276 return (true, i);
277 }
278 }
279 return (false, 0);
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/royaltymodule.sol

Locations

```
297 result[i] = mulDiv(remains, 1, pieces - i);
298 remains -= result[i];
299 }
300 return result;
301 }
```

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

/contracts/royaltymodule.sol

Locations

```
71 uint256 newSum;
72 for (uint256 i = 0; i < affectedSubaccounts.length; i++) {
73 require(affectedSubaccounts[i].royaltySplit >= _minRoyaltySplit, 'Royalty Split is smaller then set limit');
74 newSum += affectedSubaccounts[i].royaltySplit;
75 (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
76 if (found) {
```

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

/contracts/royaltymodule.sol

Locations

```
73 require(affectedSubaccounts[i].royaltySplit >= _minRoyaltySplit, 'Royalty Split is smaller then set limit');
74 newSum += affectedSubaccounts[i].royaltySplit;
75 (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
76 if (found) {
77 RASubAccount storage foundAcc = _royaltysubaccounts[royaltyAccount][indexOld];
```

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

/contracts/royaltymodule.sol

Locations

```
75 (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
76 if (found) {
77 RASubAccount storage foundAcc = _royaltysubaccounts[royaltyAccount][indexOld];
78 oldSum += foundAcc.royaltySplit;
79 //Check rights to decrease royalty split
```

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

/contracts/royaltymodule.sol

Locations

```
77 RASubAccount storage foundAcc = _royaltysubaccounts[royaltyAccount][indexOld];
78 oldSum += foundAcc.royaltySplit;
79 //Check rights to decrease royalty split
80 if (affectedSubaccounts[i].royaltySplit < foundAcc.royaltySplit) {
81 if (foundAcc.isIndividual) {
```

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

/contracts/royaltymodule.sol

Locations

```
79 //Check rights to decrease royalty split
80 if (affectedSubaccounts[i].royaltySplit < foundAcc.royaltySplit) {
81 if (foundAcc.isIndividual) {
82 require(affectedSubaccounts[i].accountId == sender, 'Only individual subaccount owner can decrease royaltySplit');
83 } else {
84 require(isTokenOwner, 'Only parent token owner can decrease royalty subaccount royaltySplit');
```

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

/contracts/royaltymodule.sol

Locations

```
80  if (affectedSubaccounts[i].royaltySplit < foundAcc.royaltySplit) {
81  if (foundAcc.isIndividual) {
82  require(affectedSubaccounts[i].accountId == sender, 'Only individual subaccount owner can decrease royaltySplit');
83  } else {
84  require(isTokenOwner, 'Only parent token owner can decrease royalty subaccount royaltySplit');
85  }
```

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

/contracts/royaltymodule.sol

Locations

```
91  }
92  }
93  require(oldSum == newSum, 'Total royaltySplit must be 10000');
94
95  //Update royalty split for subaccounts and add new subaccounts
```

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

/contracts/royaltymodule.sol

Locations

```
97  (bool found, uint256 indexOld) = _findSubaccountIndex(royaltyAccount, affectedSubaccounts[i].accountId);
98  if (found) {
99  _royaltysubaccounts[royaltyAccount][indexOld].royaltySplit = affectedSubaccounts[i].royaltySplit;
100 } else {
101 require(_royaltysubaccounts[royaltyAccount].length < _maxSubAccount, 'Too many Royalty subaccounts');
```

UNKNOWN

Out of bounds array access

SWC-110

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

Source file

/contracts/royaltymodule.sol

Locations

```
98  if (found) {
99  _royaltysubaccounts[royaltyAccount][indexOld].royaltySplit = affectedSubaccounts[i].royaltySplit;
100 } else {
101 require(_royaltysubaccounts[royaltyAccount].length < _maxSubAccount, 'Too many Royalty subaccounts');
102 _royaltysubaccounts[royaltyAccount].push(RASubAccount(true, affectedSubaccounts[i].royaltySplit, 0, affectedSubaccounts[i].accountId));
103 }
```

UNKNOWN

Out of bounds array access

SWC-110

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

Source file

/contracts/royaltymodule.sol

Locations

```
99  _royaltysubaccounts[royaltyAccount][indexOld].royaltySplit = affectedSubaccounts[i].royaltySplit;
100 } else {
101 require(_royaltysubaccounts[royaltyAccount].length < _maxSubAccount, 'Too many Royalty subaccounts');
102 _royaltysubaccounts[royaltyAccount].push(RASubAccount(true, affectedSubaccounts[i].royaltySplit, 0, affectedSubaccounts[i].accountId));
103 }
```

UNKNOWN

Out of bounds array access

SWC-110

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

Source file

/contracts/royaltymodule.sol

Locations

```
103 }
104 }
105 }
106
107 //Deleting a Royalty Account
108 function deleteRoyaltyAccount(uint256 tokenId) public virtual onlyOwner {
109 address royaltyAccount = _tokenindextoRA[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

/contracts/royaltymodule.sol

Locations

```
106
107 //Deleting a Royalty Account
108 function deleteRoyaltyAccount(uint256 tokenId) public virtual onlyOwner {
109 address royaltyAccount = _tokenindextoRA[tokenId];
110 for (uint256 i = 0; i < _royaltysubaccounts[royaltyAccount].length; 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

/contracts/royaltymodule.sol

Locations

```
110 for (uint256 i = 0; i < _royaltysubaccounts[royaltyAccount].length; i++) {
111 if (_royaltysubaccounts[royaltyAccount][i].isIndividual) {
112 require(_royaltysubaccounts[royaltyAccount][i].royaltyBalance == 0, "Can't delete non empty royalty account");
113 }
114 }
```

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

/contracts/royaltymodule.sol

Locations

```
110 for (uint256 i = 0; i < _royaltysubaccounts[royaltyAccount].length; i++) {
111 if (_royaltysubaccounts[royaltyAccount][i].isIndividual) {
112 require(_royaltysubaccounts[royaltyAccount][i].royaltyBalance == 0, "Can't delete non empty royalty account");
113 }
114 }
115 delete _royaltyaccount[royaltyAccount];
116 delete _royaltysubaccounts[royaltyAccount];
117 delete _tokenindextoRA[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

/contracts/royaltymodule.sol

Locations

```
185 //skip calculate for 0% subaccounts
186 if (_royaltysubaccounts[royaltyAccount][i].royaltySplit == 0) continue;
187 //calculate royalty split sum
188 uint256 paymentSplit = mulDiv(remainsValue, _royaltysubaccounts[royaltyAccount][i].royaltySplit, remainsSplit);
189 remainsValue -= paymentSplit;
190 remainsSplit -= _royaltysubaccounts[royaltyAccount][i].royaltySplit;
```

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

/contracts/royaltymodule.sol

Locations

```
187 //calculate royalty split sum
188 uint256 paymentSplit = mulDiv(remainsValue, _royaltysubaccounts[royaltyAccount][i].royaltySplit, remainsSplit);
189 remainsValue -= paymentSplit;
190 remainsSplit -= _royaltysubaccounts[royaltyAccount][i].royaltySplit;
191 //distribute if IND subaccount
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
```

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

/contracts/royaltymodule.sol

Locations

```
190 remainsSplit -= _royaltysubaccounts[royaltyAccount][i].royaltySplit;
191 //distribute if IND subaccount
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
193 _royaltysubaccounts[royaltyAccount][i].royaltyBalance += paymentSplit;
194 emit RoyalyDistributed(tokenId, _royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, assetId);
```

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

/contracts/royaltymodule.sol

Locations

```
191 //distribute if IND subaccount
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
193 _royaltysubaccounts[royaltyAccount][i].royaltyBalance += paymentSplit;
194 emit RoyalyDistributed(tokenId, _royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, assetId);
195 }
```

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

/contracts/royaltymodule.sol

Locations

```
192 if (_royaltysubaccounts[royaltyAccount][i].isIndividual == true) {
193 _royaltysubaccounts[royaltyAccount][i].royaltyBalance += paymentSplit;
194 emit RoyalyDistributed(tokenId, _royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, assetId);
195 }
196 //distribute if RA subaccounts
```

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

/contracts/royaltymodule.sol

Locations

```
194 emit RoyalyDistributed(tokenId, _royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, assetId);
195 }
196 //distribute if RA subaccounts
197 else {
198 _distributePayment(_royaltysubaccounts[royaltyAccount][i].accountId, paymentSplit, tokenId);
199 }
```

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

/contracts/royaltymodule.sol

Locations

```
199 }
200 }
201 return true;
202 }
203
204 function isSupportedTokenType(uint256 tokenId, string calldata tokenType) public view returns (bool) {
205 return _isSameString(tokenType, _royaltyaccount[_tokenindextoRA[tokenId]].tokenType);
206 }
```

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

/contracts/royaltymodule.sol

Locations

```
232 }
233
234 function getBalance(uint256 tokenId, address subaccount) public view virtual returns (uint256) {
235 (bool found, uint256 subaccountIndex) = findSubaccountIndex(tokenId, subaccount);
236 if (!found) return 0;
```

UNKNOWN

Out of bounds array access

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

SWC-110

Source file

/contracts/royaltymodule.sol

Locations

```
247 (bool subaccountFound, uint256 subaccountIndex) = findSubaccountIndex(tokenId, subaccount);
248 require(subaccountFound, 'Subaccount not found');
249 require(_royaltysubaccounts[_tokenindextoRA[tokenId]][subaccountIndex].royaltyBalance >= amount, 'Insufficient royalty balance');
250 _royaltysubaccounts[_tokenindextoRA[tokenId]][subaccountIndex].royaltyBalance -= amount;
251 }
```

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

/contracts/royaltymodule.sol

Locations

```
251 }
252
253 //https://hackmd.io/@o70I-dRsSdeopRewqTLbnw/r1NDumcBt#Update-RA-ownership-with-payout-to-approved-address-from
254 //Used in RoyaltyBearingToken._safeTransferFrom(address, address,uint256, bytes)
255 //for transfer royalty account ownership after tranfer token ownership
```

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

/contracts/royaltymodule.sol

Locations

```
262 (bool found, uint256 index) = _findSubaccountIndex(royaltyAccount, seller);
263 require(found, 'Seller subaccount not found');
264 require(_royaltysubaccounts[royaltyAccount][index].royaltyBalance == uint256(0), 'Seller subaccount must have 0 balance');
265
266 //replace owner of subaccount
267 _royaltysubaccounts[royaltyAccount][index].accountId = buyer;
268 }
```

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

/contracts/royaltymodule.sol

Locations

```
268 }
269
270 // Find subaccount index by subaccount address
271 function _findSubaccountIndex(address royaltyAccount, address subaccount) internal view virtual returns (bool, uint256) {
272 //local variable decrease contract code size
273 RASubAccount[] storage subAccounts = _royaltysubaccounts[royaltyAccount];
```

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

/contracts/royaltymodule.sol

Locations

```
277 }
278 }
279 return (false, 0);
280 }
```

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

/contracts/royaltymodule.sol

Locations

```
298 remains -= result[i];
299 }
300 return result;
301 }
302 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

/contracts/royaltymodule.sol

Locations

```
130 address raAccountId = address(bytes20(keccak256(abi.encodePacked(tokenId, to, block.number))));
131 if (parentTokenId == 0) {
132 //Create Royalty account without parent
133
134 //create the RA subaccount for the to address
135 _royaltysubaccounts[raAccountId].push(RASubAccount({isIndividual: true, royaltySplit: 10000 - _royaltySplitTT, royaltyBalance: 0, accountId: to}));
```