MythX

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
57108bf6-255b-48ca-9da7-49a8144a8630	/contracts/paymentmodule.sol	0

Started	Tue Mar 22 2022 15:51:33 GMT+0000 (Coordinated Universal Time)
Finished	Tue Mar 22 2022 16:36:49 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Paymentmodule.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0
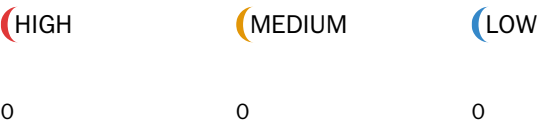
ISSUES

UNKNOWN   Arithmetic operation "++" discovered

SWC-101   This plugin produces issues to support false positive discovery within MythX.

Source file
/contracts/paymentmodule.sol
Locations

```
42   }
43   //add to list index
44   listedNFTList.push(tokenIds[0]);
45   }
```

UNKNOWN   Arithmetic operation "++" discovered

SWC-101   This plugin produces issues to support false positive discovery within MythX.

Source file
/contracts/paymentmodule.sol
Locations

```
54   }
55
56   function removeListNFT(uint256 tokenId) public virtual onlyOwner {
57   require(registeredPayment[tokenId].buyer == address(0), 'RegisterPayment exists for NFT');
58   //unlock token
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/paymentmodule.sol

Locations

```
60    tokenLock[listedNFT[tokenId].listedtokens[i]] = false;
61    }
62    //delete from index
63    for (uint256 i = 0; i < listedNFTList.length; i++) {
64    if (listedNFTList[i] == tokenId) {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/paymentmodule.sol

Locations

```
63    for (uint256 i = 0; i < listedNFTList.length; i++) {
64    if (listedNFTList[i] == tokenId) {
65    listedNFTList[i] = listedNFTList[listedNFTList.length - 1];
66    listedNFTList.pop();
67    break;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/paymentmodule.sol

Locations

```
66    listedNFTList.pop();
67    break;
68    }
69    }
70
71    delete listedNFT[tokenId];
72    }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/paymentmodule.sol

Locations

```
99   for (uint256 i = 0; i < listedTokens.length; i++) {
100  require(tokenIds[i] == listedTokens[i], 'One or more tokens are not listed');
101  }
102  return true;
103  }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/paymentmodule.sol

Locations

```
146  require(tokenIds[i] == listedTokens[i], 'List of token not match');
147  }
148  return true;
149  }
```

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

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

SWC-101

Source file

/contracts/paymentmodule.sol

Locations

```
66   listedNFTList.pop();
67   break;
68   }
69   }
70
71   delete listedNFT[tokenId];
72   }
```

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/paymentmodule.sol

Locations

```
36   require(!existsInListNFT(tokenIds), 'Already exists');
37   require(tokenIds.length <= _maxListingNumber, 'Too many NFTs listed');
38   listedNFT[tokenIds[0]] = ListedNFT({seller: seller, listedtokens: tokenIds, tokenType: tokenType, price: price});
39   //lock tokens
40   for (uint256 i = 0; i < tokenIds.length; i++) {
41   tokenLock[tokenIds[i]] = true;
```

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/paymentmodule.sol

Locations

```
43   //add to list index
44   listedNFTList.push(tokenIds[0]);
45   }
46
47   function existsInListNFT(uint256[] memory tokenIds) public view virtual returns (bool) {
48   if (listedNFT[tokenIds[0]].seller != address(0)) return true;
```

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/paymentmodule.sol

Locations

```
45   }
46
47   function existsInListNFT(uint256[] memory tokenIds) public view virtual returns (bool) {
48   if (listedNFT[tokenIds[0]].seller != address(0)) return true;
```

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/paymentmodule.sol

Locations

```
63    for (uint256 i = 0; i < listedNFTList.length; i++) {
64    if (listedNFTList[i] == tokenId) {
65    listedNFTList[i] = listedNFTList[listedNFTList.length - 1];
66    listedNFTList.pop();
67    break;
68    }
```

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/paymentmodule.sol

Locations

```
65    listedNFTList[i] = listedNFTList[listedNFTList.length - 1];
66    listedNFTList.pop();
67    break;
68    }
69    }
```

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/paymentmodule.sol

Locations

```
65    listedNFTList[i] = listedNFTList[listedNFTList.length - 1];
66    listedNFTList.pop();
67    break;
68    }
69    }
70
71    delete listedNFT[tokenId];
72    }
```

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/paymentmodule.sol

Locations

```
89    //check if NFT(s) are even listed
90    require(listedNFT[tokenIds[0]].seller != address(0), 'NFT(s) not listed');
91    //check if seller is really a seller
92    require(listedNFT[tokenIds[0]].seller == seller, 'Submitted Seller is not Seller');
93    //check if payment is sufficient
94    require(listedNFT[tokenIds[0]].price <= payment, 'Payment is too low');
```

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/paymentmodule.sol

Locations

```
91    //check if seller is really a seller
92    require(listedNFT[tokenIds[0]].seller == seller, 'Submitted Seller is not Seller');
93    //check if payment is sufficient
94    require(listedNFT[tokenIds[0]].price <= payment, 'Payment is too low');
95    //check if token type supported
96    require(_isSameString(listedNFT[tokenIds[0]].tokenType, tokenType), 'Payment token does not match list token type');
```

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/paymentmodule.sol

Locations

```
94    require(listedNFT[tokenIds[0]].price <= payment, 'Payment is too low');
95    //check if token type supported
96    require(_isSameString(listedNFT[tokenIds[0]].tokenType, tokenType), 'Payment token does not match list token type');
97    //check if listed NFT(s) match NFT(s) in the payment and are controlled by seller
98    uint256[] memory listedTokens = listedNFT[tokenIds[0]].listedtokens;
```

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/paymentmodule.sol

Locations

```
95    //check if token type supported
96    require(_isSameString(listedNFT[tokenIds[0]].tokenType, tokenType), 'Payment token does not match list token type');
97    //check if listed NFT(s) match NFT(s) in the payment and are controlled by seller
98    uint256[] memory listedTokens = listedNFT[tokenIds[0]].listedtokens;
99    for (uint256 i = 0; i < listedTokens.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/paymentmodule.sol

Locations

```
98    uint256[] memory listedTokens = listedNFT[tokenIds[0]].listedtokens;
99    for (uint256 i = 0; i < listedTokens.length; i++) {
100   require(tokenIds[i] == listedTokens[i], 'One or more tokens are not listed');
101   }
102   return 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/paymentmodule.sol

Locations

```
100   require(tokenIds[i] == listedTokens[i], 'One or more tokens are not listed');
101   }
102   return true;
103   }
104
105   function addRegisterPayment(
106   address buyer,
107   uint256[] calldata tokenIds,
```

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/paymentmodule.sol

Locations

```
103  }
104
105  function addRegisterPayment(
106  address buyer,
107  uint256[] calldata tokenIds,
```

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/paymentmodule.sol

Locations

```
110  ) public virtual onlyOwner {
111  require(registeredPayment[tokenIds[0]].buyer == address(0), 'RegisterPayment already exists');
112  registeredPayment[tokenIds[0]] = RegisteredPayment({buyer: buyer, boughtTokens: tokenIds, tokenType: tokenType, payment: payment});
113  }
```

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/paymentmodule.sol

Locations

```
110  ) public virtual onlyOwner {
111  require(registeredPayment[tokenIds[0]].buyer == address(0), 'RegisterPayment already exists');
112  registeredPayment[tokenIds[0]] = RegisteredPayment({buyer: buyer, boughtTokens: tokenIds, tokenType: tokenType, payment: payment});
113  }
```

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/paymentmodule.sol

Locations

```
138  require(registeredPayment[tokenIds[0]].buyer == buyer, 'RegisterPayment not found');
139  //check if payment is sufficient
140  require(registeredPayment[tokenIds[0]].payment == payment, 'Payment not match');
141  //check if token type supported
142  require(_isSameString(registeredPayment[tokenIds[0]].tokenType, tokenType), 'TokenType not match');
```

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/paymentmodule.sol

Locations

```
140    require(registeredPayment[tokenIds[0]].payment == payment, 'Payment not match');
141    //check if token type supported
142    require(_isSameString(registeredPayment[tokenIds[0]].tokenType, tokenType), 'TokenType not match');
143    //check if token list are same
144    uint256[] memory listedTokens = listedNFT[tokenIds[0]].listedtokens;
```

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/paymentmodule.sol

Locations

```
142    require(_isSameString(registeredPayment[tokenIds[0]].tokenType, tokenType), 'TokenType not match');
143    //check if token list are same
144    uint256[] memory listedTokens = listedNFT[tokenIds[0]].listedtokens;
145    for (uint256 i = 0; i < listedTokens.length; i++) {
146    require(tokenIds[i] == listedTokens[i], 'List of token not match');
```

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/paymentmodule.sol

Locations

```
144    uint256[] memory listedTokens = listedNFT[tokenIds[0]].listedtokens;
145    for (uint256 i = 0; i < listedTokens.length; i++) {
146    require(tokenIds[i] == listedTokens[i], 'List of token not match');
147    }
148    return true;
```

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/paymentmodule.sol

Locations

```
149  }
150
151  //https://hackmd.io/@o70I-dRsSdeopRewqTLbnw/r1NDumcBt#Removing-the-Payment-entry-in-registeredPayment-after-successful-transfer-new-in-v12
152  function removeRegisterPayment(address buyer, uint256 tokenId) public virtual onlyOwner {
153  require(registeredPayment[tokenId].buyer == buyer, 'RegisterPayment not found');
```

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/paymentmodule.sol

Locations

```
149  }
150
151  //https://hackmd.io/@o70I-dRsSdeopRewqTLbnw/r1NDumcBt#Removing-the-Payment-entry-in-registeredPayment-after-successful-transfer-new-in-v12
152  function removeRegisterPayment(address buyer, uint256 tokenId) public virtual onlyOwner {
153  require(registeredPayment[tokenId].buyer == buyer, 'RegisterPayment not found');
```