MythX

REPORT 62553069C5A1DE00191D065E

Created	Tue Apr 12 2022 07:55:21 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	62552e05ede4f21e388224d6

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
d81b32c2-de08-4ee7-bd13-fd676f42e547	/contracts/playerselfauction.sol	1

Analysis d81b32c2-de08-4ee7-bd13-fd676f42e547 MythX

Started	Tue Apr 12 2022 07:55:23 GMT+0000 (Coordinated Universal Time)
Finished	Tue Apr 12 2022 08:23:47 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Playerselfauction.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/playerselfauction.sol
Locations

```
91 uint256[] memory balances = IERC1155(_nftAddress).balanceOfBatch(addresses, _tokenIds);
92 for (uint256 i = 0; i < balances.length; i++) {
93 require(balances[i] > 0, "Sender does not own the NFT.");
94 }
95 IERC1155(_nftAddress).safeBatchTransferFrom(msg.sender, address(this), _tokenIds, balances, "");
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file
/contracts/playerselfauction.sol
Locations

```
95 IERC1155(_nftAddress).safeBatchTransferFrom(msg.sender, address(this), _tokenIds, balances, "");
96 } else {
97 for (uint256 i = 0; i < _tokenIds.length; i++) {
98 address owner = IERC721(_nftAddress).ownerOf(_tokenIds[i]);
99 require(owner == msg.sender, "Sender does not own the NFT.");
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
100 IERC721(_nftAddress).safeTransferFrom(msg.sender, address(this), _tokenIds[i]);
101 }
102 }
103 _;
104 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
163 uint256[] memory percentages = new uint256[](_defaultFee > 0 && _defaultFeeRecipient != address(0) ? _feeRecipients.length + 1 : _feeRecipients.length);
164 for (uint i = 0; i < _feeRecipients.length; i++) {
165 recipients[i] = _feeRecipients[i];
166 percentages[i] = _feePercentages[i];
167 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
166 percentages[i] = _feePercentages[i];
167 }
168 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
169 recipients[recipients.length - 1] = _defaultFeeRecipient;
170 percentages[percentages.length - 1] = _defaultFee;
171 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
168 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
169 recipients[recipients.length - 1] = _defaultFeeRecipient;
170 percentages[percentages.length - 1] = _defaultFee;
171 }
172 // Create the auction
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
176 nftAuctions[auctionHash].bidIncreasePercentage = _bidIncreasePercentage != 0 ? _bidIncreasePercentage : defaultBidIncreasePercentage;
177 nftAuctions[auctionHash].feeRecipients = recipients;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
176 nftAuctions[auctionHash].bidIncreasePercentage = _bidIncreasePercentage != 0 ? _bidIncreasePercentage : defaultBidIncreasePercentage;
177 nftAuctions[auctionHash].feeRecipients = recipients;
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
186
187 /**
188 * @notice External function used to setup an auction
189 * @dev Calls the internal function above to setup an auction
190 * @param _nftAddress address of the NFT contract (must be registered on the PlayerselfRegistry contract)
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
242
243
244 /** Sales **/
245 function _setupSale(
246 bytes32 saleHash,
247 address _nftAddress,
248 uint256[] memory _tokenIds,
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
246 bytes32 saleHash,
247 address _nftAddress,
248 uint256[] memory _tokenIds,
249 uint256 _buyNowPrice,
250 address _whitelistedBuyer,
251 address[] memory _feeRecipients,
252 uint256[] memory _feePercentages
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
259 uint256[] memory percentages = new uint256[](_defaultFee > 0 && _defaultFeeRecipient != address(0) ? _feeRecipients.length + 1 : _feeRecipients.length);
260 for (uint i = 0; i < _feeRecipients.length; i++) {
261 recipients[i] = _feeRecipients[i];
262 percentages[i] = _feePercentages[i];
263 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
262 percentages[i] = _feePercentages[i];
263 }
264 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
265 recipients[recipients.length - 1] = _defaultFeeRecipient;
266 percentages[percentages.length - 1] = _defaultFee;
267 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
264 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
265 recipients[recipients.length - 1] = _defaultFeeRecipient;
266 percentages[percentages.length - 1] = _defaultFee;
267 }
268 nftAuctions[saleHash].nftAddress = _nftAddress;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
268 nftAuctions[saleHash].nftAddress = _nftAddress;
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
271 nftAuctions[saleHash].feePercentages = _feePercentages;
272 nftAuctions[saleHash].buyNowPrice = _buyNowPrice;
273 nftAuctions[saleHash].nftSeller = msg.sender;
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
271 nftAuctions[saleHash].feePercentages = _feePercentages;
272 nftAuctions[saleHash].buyNowPrice = _buyNowPrice;
273 nftAuctions[saleHash].nftSeller = msg.sender;
274 nftAuctions[saleHash].whitelistedBuyer = _whitelistedBuyer;
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
344
345 function _getPortionOfBid(uint256 _totalBid, uint256 _percentage) internal pure returns (uint256) {
346 return (_totalBid * (_percentage)) / 1e18;
347 }
348
349 /** Bid functions **/
350 function makeBid(bytes32 hash) external payable auctionExists(hash) {
351 require(nftAuctions[hash].nftSeller != address(0), "Non-existing auction.");
352 require(_isAuctionOngoing(hash), "Auction ended.");
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
344
345 function _getPortionOfBid(uint256 _totalBid, uint256 _percentage) internal pure returns (uint256) {
346 return (_totalBid * (_percentage)) / 1e18;
347 }
348
349 /** Bid functions **/
350 function makeBid(bytes32 hash) external payable auctionExists(hash) {
351 require(nftAuctions[hash].nftSeller != address(0), "Non-existing auction.");
352 require(_isAuctionOngoing(hash), "Auction ended.");
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
347 }
348
349 /** Bid functions **/
350 function makeBid(bytes32 hash) external payable auctionExists(hash) {
351 require(nftAuctions[hash].nftSeller != address(0), "Non-existing auction.");
352 require(_isAuctionOngoing(hash), "Auction ended.");
```

UNKNOWN Arithmetic operation "/" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
350 function makeBid(bytes32 hash) external payable auctionExists(hash) {
351 require(nftAuctions[hash].nftSeller != address(0), "Non-existing auction.");
352 require(_isAuctionOngoing(hash), "Auction ended.");
353 require(!_isWhitelistedSale(hash) || nftAuctions[hash].whitelistedBuyer == msg.sender, "Only whitelisted buyer.");
354 require(nftAuctions[hash].nftSeller != msg.sender, "Bidding own auction?");
355 require(msg.value > 0 && (nftAuctions[hash].minPrice == 0 || msg.value >= nftAuctions[hash].minPrice), "Invalid payment.");
```

UNKNOWN Arithmetic operation "*" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
350 function makeBid(bytes32 hash) external payable auctionExists(hash) {
351 require(nftAuctions[hash].nftSeller != address(0), "Non-existing auction.");
352 require(_isAuctionOngoing(hash), "Auction ended.");
353 require(!_isWhitelistedSale(hash) || nftAuctions[hash].whitelistedBuyer == msg.sender, "Only whitelisted buyer.");
354 require(nftAuctions[hash].nftSeller != msg.sender, "Bidding own auction?");
355 require(msg.value > 0 && (nftAuctions[hash].minPrice == 0 || msg.value >= nftAuctions[hash].minPrice), "Invalid payment.");
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
373 }
374 emit HighestBidTaken(hash);
375 emit NftSold(hash, _nftHighestBidder, nftAuctions[hash].tokenIds, _nftHighestBid, nftAuctions[hash].nftAddress);
376 _payout(nftAuctions[hash].nftSeller, (_nftHighestBid - feesPaid));
377
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
374 emit HighestBidTaken(hash);
375 emit NftSold(hash, _nftHighestBidder, nftAuctions[hash].tokenIds, _nftHighestBid, nftAuctions[hash].nftAddress);
376 _payout(nftAuctions[hash].nftSeller, (_nftHighestBid - feesPaid));
377
378 address[] memory addresses = new address[](nftAuctions[hash].tokenIds.length);
```

UNKNOWN Arithmetic operation "-" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
381 }
382
383 IPlayerselfRegistry.NFT memory _nft = registry.getNFT(nftAuctions[hash].nftAddress);
384 if (_nft.supportsBatch) {
385 uint256[] memory balances = IERC1155(nftAuctions[hash].nftAddress).balanceOfBatch(addresses, nftAuctions[hash].tokenIds);
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
383 IPlayerselfRegistry.NFT memory _nft = registry.getNFT(nftAuctions[hash].nftAddress);
384 if (_nft.supportsBatch) {
385 uint256[] memory balances = IERC1155(nftAuctions[hash].nftAddress).balanceOfBatch(addresses, nftAuctions[hash].tokenIds);
386 IERC1155(nftAuctions[hash].nftAddress).safeBatchTransferFrom(address(this), _nftHighestBidder, nftAuctions[hash].tokenIds, balances, "");
387 } else {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
394
395 function _updateHighestBid(bytes32 hash) internal {
396 nftAuctions[hash].nftHighestBid = uint256(msg.value);
397 nftAuctions[hash].nftHighestBidder = msg.sender;
398 }
```

UNKNOWN Arithmetic operation "+" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
413 address prevNftHighestBidder = nftAuctions[hash].nftHighestBidder;
414 uint256 prevNftHighestBid = nftAuctions[hash].nftHighestBid;
415 _updateHighestBid(hash);
416
417 if (prevNftHighestBidder != address(0)) {
418 _payout(prevNftHighestBidder, prevNftHighestBid);
419 }
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
439 IPlayerselfRegistry.NFT memory _nft = registry.getNFT(nftAuctions[hash].nftAddress);
440 if (_nft.supportsBatch) {
441 uint256[] memory balances = IERC1155(nftAuctions[hash].nftAddress).balanceOfBatch(addresses, nftAuctions[hash].tokenIds);
442 IERC1155(nftAuctions[hash].nftAddress).safeBatchTransferFrom(address(this), nftAuctions[hash].nftSeller, nftAuctions[hash].tokenIds, balances, "");
443 } else {
```

UNKNOWN Arithmetic operation "++" discovered

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
447 }
448 _resetAuction(hash);
449 emit NftAuctionWithdrawn(hash, nftAuctions[hash].nftSeller);
450 }
451
452 /** Update methods **/
```

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

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
176 nftAuctions[auctionHash].bidIncreasePercentage = _bidIncreasePercentage != 0 ? _bidIncreasePercentage : defaultBidIncreasePercentage;
177 nftAuctions[auctionHash].feeRecipients = recipients;
```

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

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
176 nftAuctions[auctionHash].bidIncreasePercentage = _bidIncreasePercentage != 0 ? _bidIncreasePercentage : defaultBidIncreasePercentage;
177 nftAuctions[auctionHash].feeRecipients = recipients;
```

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

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

SWC-101

Source file

/contracts/playerselfauction.sol

Locations

```
268 nftAuctions[saleHash].nftAddress = _nftAddress;
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
271 nftAuctions[saleHash].feePercentages = _feePercentages;
272 nftAuctions[saleHash].buyNowPrice = _buyNowPrice;
273 nftAuctions[saleHash].nftSeller = msg.sender;
```

UNKNOWN

SWC-101

Compiler-rewritable "<uint> - 1" discovered

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

Source file

/contracts/playerselfauction.sol

Locations

```
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
271 nftAuctions[saleHash].feePercentages = _feePercentages;
272 nftAuctions[saleHash].buyNowPrice = _buyNowPrice;
273 nftAuctions[saleHash].nftSeller = msg.sender;
274 nftAuctions[saleHash].whitelistedBuyer = _whitelistedBuyer;
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.1"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

/contracts/playerselfauction.sol

Locations

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.1;
3
4 import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
```

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/playerselfauction.sol

Locations

```
91 uint256[] memory balances = IERC1155(_nftAddress).balanceOfBatch(addresses, _tokenIds);
92 for (uint256 i = 0; i < balances.length; i++) {
93 require(balances[i] > 0, "Sender does not own the NFT.");
94 }
95 IERC1155(_nftAddress).safeBatchTransferFrom(msg.sender, address(this), _tokenIds, balances, "");
96 } else {
```

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/playerselfauction.sol

Locations

```
96  } else {
97  for (uint256 i = 0; i < _tokenIds.length; i++) {
98  address owner = IERC721(_nftAddress).ownerOf(_tokenIds[i]);
99  require(owner == msg.sender, "Sender does not own the NFT.");
100 IERC721(_nftAddress).safeTransferFrom(msg.sender, address(this), _tokenIds[i]);
```

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/playerselfauction.sol

Locations

```
104 }
105
106 modifier auctionExists(bytes32 hash) {
107 require(nftAuctions[hash].nftSeller != address(0) && nftAuctions[hash].tokenIds.length > 0, "Auction does not exist.");
108 _;
109 }
```

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/playerselfauction.sol

Locations

```
109 }
110
111 modifier auctionNotExists(bytes32 hash) {
112 require(nftAuctions[hash].nftSeller == address(0), "Auction already exists.");
113 _;
```

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/playerselfauction.sol

Locations

```
168 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
169 recipients[recipients.length - 1] = _defaultFeeRecipient;
170 percentages[percentages.length - 1] = _defaultFee;
171 }
172 // Create the auction
```

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/playerselfauction.sol

Locations

```
168 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
169 recipients[recipients.length - 1] = _defaultFeeRecipient;
170 percentages[percentages.length - 1] = _defaultFee;
171 }
172 // Create the auction
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
```

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/playerselfauction.sol

Locations

```
170 percentages[percentages.length - 1] = _defaultFee;
171 }
172 // Create the auction
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
```

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/playerselfauction.sol

Locations

```
171 }
172 // Create the auction
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
```

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/playerselfauction.sol

Locations

```
172 // Create the auction
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
176 nftAuctions[auctionHash].bidIncreasePercentage = _bidIncreasePercentage != 0 ? _bidIncreasePercentage : defaultBidIncreasePercentage;
177 nftAuctions[auctionHash].feeRecipients = recipients;
```

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/playerselfauction.sol

Locations

```
173 nftAuctions[auctionHash].nftAddress = _nftAddress;
174 nftAuctions[auctionHash].tokenIds = _tokenIds;
175 nftAuctions[auctionHash].auctionBidPeriod = _auctionBidPeriod != 0 ? _auctionBidPeriod : defaultAuctionBidPeriod;
176 nftAuctions[auctionHash].bidIncreasePercentage = _bidIncreasePercentage != 0 ? _bidIncreasePercentage : defaultBidIncreasePercentage;
177 nftAuctions[auctionHash].feeRecipients = recipients;
```

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/playerselfauction.sol

Locations

```
264 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
265 recipients[recipients.length - 1] = _defaultFeeRecipient;
266 percentages[percentages.length - 1] = _defaultFee;
267 }
268 nftAuctions[saleHash].nftAddress = _nftAddress;
```

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/playerselfauction.sol

Locations

```
264 if (_defaultFee > 0 && _defaultFeeRecipient != address(0)) {
265 recipients[recipients.length - 1] = _defaultFeeRecipient;
266 percentages[percentages.length - 1] = _defaultFee;
267 }
268 nftAuctions[saleHash].nftAddress = _nftAddress;
269 nftAuctions[saleHash].tokenIds = _tokenIds;
```

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/playerselfauction.sol

Locations

```
266 percentages[percentages.length - 1] = _defaultFee;
267 }
268 nftAuctions[saleHash].nftAddress = _nftAddress;
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
```

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/playerselfauction.sol

Locations

```
266 percentages[percentages.length - 1] = _defaultFee;
267 }
268 nftAuctions[saleHash].nftAddress = _nftAddress;
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
```

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/playerselfauction.sol

Locations

```
268 nftAuctions[saleHash].nftAddress = _nftAddress;
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
271 nftAuctions[saleHash].feePercentages = _feePercentages;
272 nftAuctions[saleHash].buyNowPrice = _buyNowPrice;
273 nftAuctions[saleHash].nftSeller = msg.sender;
```

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/playerselfauction.sol

Locations

```
269 nftAuctions[saleHash].tokenIds = _tokenIds;
270 nftAuctions[saleHash].feeRecipients = _feeRecipients;
271 nftAuctions[saleHash].feePercentages = _feePercentages;
272 nftAuctions[saleHash].buyNowPrice = _buyNowPrice;
273 nftAuctions[saleHash].nftSeller = msg.sender;
274 nftAuctions[saleHash].whitelistedBuyer = _whitelistedBuyer;
```

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/playerselfauction.sol

Locations

```
373 }
374 emit HighestBidTaken(hash);
375 emit NftSold(hash, _nftHighestBidder, nftAuctions[hash].tokenIds, _nftHighestBid, nftAuctions[hash].nftAddress);
376 _payout(nftAuctions[hash].nftSeller, (_nftHighestBid - feesPaid));
377
378 address[] memory addresses = new address[](nftAuctions[hash].tokenIds.length);
```

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/playerselfauction.sol

Locations

```
374 emit HighestBidTaken(hash);
375 emit NftSold(hash, _nftHighestBidder, nftAuctions[hash].tokenIds, _nftHighestBid, nftAuctions[hash].nftAddress);
376 _payout(nftAuctions[hash].nftSeller, (_nftHighestBid - feesPaid));
377
378 address[] memory addresses = new address[](nftAuctions[hash].tokenIds.length);
379 for (uint256 i = 0; i < nftAuctions[hash].tokenIds.length; i++) {
380 addresses[i] = address(this);
```

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/playerselfauction.sol

Locations

```
383 IPlayerselfRegistry.NFT memory _nft = registry.getNFT(nftAuctions[hash].nftAddress);
384 if (_nft.supportsBatch) {
385 uint256[] memory balances = IERC1155(nftAuctions[hash].nftAddress).balanceOfBatch(addresses, nftAuctions[hash].tokenIds);
386 IERC1155(nftAuctions[hash].nftAddress).safeBatchTransferFrom(address(this), _nftHighestBidder, nftAuctions[hash].tokenIds, balances, "");
387 } else {
```

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/playerselfauction.sol

Locations

```
396 nftAuctions[hash].nftHighestBid = uint256(msg.value);
397 nftAuctions[hash].nftHighestBidder = msg.sender;
398 }
399
400 function _payout(address _recipient, uint256 _amount) internal {
401 // attempt to send the funds to the recipient
402 (bool success, ) = payable(_recipient).call{
```

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/playerselfauction.sol

Locations

```
439 IPlayerselfRegistry.NFT memory _nft = registry.getNFT(nftAuctions[hash].nftAddress);
440 if (_nft.supportsBatch) {
441 uint256[] memory balances = IERC1155(nftAuctions[hash].nftAddress).balanceOfBatch(addresses, nftAuctions[hash].tokenIds);
442 IERC1155(nftAuctions[hash].nftAddress).safeBatchTransferFrom(address(this), nftAuctions[hash].nftSeller, nftAuctions[hash].tokenIds, balances, "");
443 } else {
```

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/playerselfauction.sol

Locations

```
451
452 /** Update methods **/
453 function updateWhitelistedBuyer(bytes32 hash, address _newWhitelistedBuyer) external sellerOnly(hash) {
454 require(_isSale(hash), "Not a sale.");
455
456 nftAuctions[hash].whitelistedBuyer = _newWhitelistedBuyer;
```