Kim NFT marketplace

Security Review

Cantina Managed review by:
Cccz, Security Researcher
Chinmay farkya, Associate Security Researcher

November 26, 2024

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

KIM is a decentralized exchange (DEX) protocol that uses a mathematical formula to price assets, facilitating trading without needing a traditional order book.

From Nov 12th to Nov 19th the Cantina team conducted a review of kim-nft-marketplace on commit hash c2489cfe. The team identified a total of **23** issues in the following risk categories:

- Critical Risk: 2
- High Risk: 2
- Medium Risk: 7
- Low Risk: 6
- Gas Optimizations: 2
- Informational: 4

3 Findings

3.1 Critical Risk

3.1.1 Dutch auctions can be used to drain honest buyers' escrowed funds from the market

Severity: Critical Risk

Context: Auction.sol#L131-L145

Description: When a buyer bids for a dutch auction, the flow is `bidForNFT()` → `_bidForAuction()` where all the required validations are done.

Even though the `bestBidForListing[listingId]` is not required in the case of a dutch auction (because there is only one valid bid and it is immediately settled), the bid amount is stored at line 141.

This creates a problem for the listing's accounting because this is the only auction settlement flow where the `bestBidForListing[listingId]` is not deleted. The logic in `bidForNFT()` just concludes the sale and transfers the payment as well as NFT to the seller and buyer respectively.

Imagine this situation:

- A seller creates a dutch auction to start at 100 and end at 200.
- A buyer comes in and bids at the current price, the auction gets settled and everyone gets what they deserve.
- Because the value in `bestBidForListing[listingId]` still exists, the buyer can come in again before `listingEnd` i.e. timestamp 200 to cancel their bid by calling `cancelBidForNFT()`.
- The bid gets cancelled now and they get back full refund of their `bidAmount`, making the market insolvent.

This can be used by an attacker to drain the market of honest buyers' escrowed funds by repeatedly opening dutch auctions, settling it themselves and then cancelling their bids to also get a full refund of their bid.

Recommendation: The logic for dutch auctions does not ever use `bestBidForListing[listingId]`. This issue can be fixed by not storing the bid value for dutch auctions. Change the code to:

```
if (listing.listingType != ListingType.DUTCH_AUCTION) {
    _validateBuyItNowBidAndEnglishAuctionBid(currentBid, bidAmount, listing.reserve);
    (refund, refundUser, amountToEscrowOrPay) =
↪   _calculateRefundAndEscrowAmountBuyItNowBidsAndEnglish(bidAmount, currentBidder, currentBid);
    isEscrow = true; // Save the bid to accept later

    // Save the bid based on the listing type : only for BUY_IT_NOW_WITH_BIDS and English auctions
    bestBidForListing[listingId] = Bid({
            user: msg.sender,
            amount: bidAmount
        });
} else {
    amountToEscrowOrPay = _validateDutchAuctionBid(bidAmount, listing);
    isEscrow = false; // If the bid is validated immediately end the auction and settle
}
```

Also, the `settleAuction()` flow shall delete `bestBidForListing[]` for consistency in case of English and Blind Auctions.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.1.2 Completed listings can be re-entered

Severity: Critical Risk

Context: KimNFTMarketplace.sol#L395-L402

Description: In the protocol, when the listing is completed by being sold or cancelled, nothing is changed about the listing's state, and when re-entering the listing, as long as the contract has the NFT of the listing, the listing is considered valid, even if the NFT belongs to another listing. This is because of the weak check in `_isListingValid()`.

Consider the following scenario:

1. Alice lists NFT A for sale.
2. Bob buys NFT A and continues to sell it in the protocol.
3. Alice calls `cancelListing()` to cancel her completed listing, and since the contract owns NFT A (which actually belongs to Bob), the listing is still considered valid, so in `cancelListing()` the NFT A is returned to Alice, not Bob.

The attacker can exploit this to steal the NFTs of other users in the contract. This problem also affects all other ways of settling the auctions (`buyItNow()`, `settleAuction()`).

Recommendation: It is recommended to set the listing's state when the listing is complete (sold or canceled) and check that state when entering the listing. A reconfiguration of the function `_isListingValid()` is required.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The protocol reimplemented the `_isListingValid()` function to use `isNoLongerEscrowed` to track listing closure and thus determine whether the listing is valid.

3.2 High Risk

3.2.1 Allowing last-minute bidding cancellations will always make the auction settle at the reserve price

Severity: High Risk

Context: KimNFTMarketplace.sol#L252-L264

Description: The protocol allows users to cancel their bids at the last minute, and even at the `timestamp == listingEnd` (when `settleAuction()` is also allowed).

For English auctions, an attacker could use a high bid to discourage other users from bidding, then cancel their bids at the last minute(or last block time) and bid a lower one. Another thing they could do is frontrun the `settleAuction()` call to cancel their bid at `timestamp == listingEnd` in order to conclude the auction as null.

Since bids cannot be lower than the reserve price, the attacker can always make the English auction settle at the reserve price, or make them conclude as null.

Recommendation: It is recommended to disallow users from cancelling their bids before the auction closes for certain buffer period.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The new implementation is as follows, canceling bids is no longer allowed during the MIN_LENGTH_BID_EXPIRY period before the auction ends.

But `BUY_IT_NOW_WITH_BIDS` bids can still be canceled at any time, it is recommended to allow canceling of BUY_IT_NOW_WITH_BIDS bids only after they expire.

```
uint256 listingEnd = listings[listingId].end;
if (listingEnd > 0 && listings[listingId].listingType != ListingType.BUY_IT_NOW_WITH_BIDS) {
    // Ensure a buffer exists near the end of the auction where cancellations cannot happen but also gives a
↪   window for snipe bids to enter which is good for sellers
    bool isEnded = (block.timestamp + MIN_LENGTH_BID_EXPIRY) > listingEnd;
    if (isEnded) revert AuctionEnded();
}
```

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.2.2 Escrow check for `ERC1155` tokens can DOS all auction interactions

Severity: High Risk

Context: KimNFTMarketplace.sol#L76-L79, KimNFTMarketplace.sol#L395-L408

Description: All interactions with a listing/auction involve checking that the NFT is still escrowed with the marketplace contract in order to ensure that the listing is still valid. This logic is present in `_isListingValid()`. For ERC1155 NFTs, this is the check employed:

```
isMarketplaceStillHoldingNFT =
    _get1155NftBalanceOfUser(listing.nft, listing.tokenId, address(this)) == listing.amount;
```

This check can return false and DOS the whole auction process (including `settleAuction()`, `cancelBidForNFT()`, `cancelListing()`, `bidForNFT()`, `buyItNow()`, `acceptBidForNFT()`, `rejectBidForNFT()`): which is basically all ways of either settling/ cancelling the auction or cancelling the bid to retrieve funds.

Imagine this situation:

- Seller A lists a ERC1155 `tokenID` for sale: amount X.
- Seller B lists the same ERC1155 `tokenID` for sale: amount Y.
- Now all auction interactions will be permanently DOS'ed as none of these listings can be closed in any way.

This is because the `balanceOf(address(this)` will actually be greater than any one listing's value (will be X + Y).

Both the tokenIDs as well as any existing bids of the first listing (before the second listing was created) will be stuck forever. This can happen for all auction types.

Recommendation: This way of validating a listing comes with many problems. It should be reconfigured to use a state variable in the listing struct that indicates if the listing was finalized or not. Doing that will solve this problem for ERC1155 tokens as well.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The protocol reimplemented the `_isListingValid()` function to use `isNoLongerEscrowed` to track listing closure and thus determine whether the listing is valid. These changes solved the issue with ERC1155 escrow check as well because listing validity no longer depends on the balanceOf `tokenID`.

3.3 Medium Risk

3.3.1 `buyItNow()` can be used to conclude sale of a `BUY_IT_NOW_WITH_BIDS` listing at a price lower than reserve

Severity: Medium Risk

Context: KimNFTMarketplace.sol#L110-L135

Description: Current logic in `buyItNow()` allows both `BUY_IT_NOW` and `BUY_IT_NOW_WITH_BIDS` listing types to be settled. This flow just performs checks on the sale window, refunds any existing bids, and straightaway concludes the sale transferring payment and NFT to the seller and buyer respectively.

This works great for a `BUY_IT_NOW` listing. But for `BUY_IT_NOW_WITH_BIDS` this creates several problems:

- `BUY_IT_NOW_WITH_BIDS` type needs to check if the sale price (bid) is at least above `reserve` price. The reserve check is bypassed if someone calls `buyItNow()`.
- As per the specs, the price parameter for `BUY_IT_NOW_WITH_BIDS` listing has to be greater than reserve. The problem is that if `buyItNow()` is called, the sale gets concluded with the `listing.price` as the payment, which can be lower than reserve as this is not validated in `listNFT()`.

The impact is that an attacker can get the NFT for a payment lower than reserve.

Recommendation: In `listNFT()`, ensure that `BUY_IT_NOW_WITH_BIDS` type listings always have a `price >= reserve`. Modify the docs to reflect the same.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.3.2 Bidder can be griefed by the seller for a `BUY_IT_NOW_WITH_BIDS` listing that has `listingEnd` set

Severity: Medium Risk

Context: KimNFTMarketplace.sol#L252-L259

Description: `BUY_IT_NOW_WITH_BIDS` listings can have start and end time optionally. If they do have them, the start and end times are validated against `block.timestamp` to ensure the bids can't be cancelled after the listing ends.

This creates a problem for the buyer. Imagine this situation:

- User A creates a listing of `BUY_IT_NOW_WITH_BIDS` type with `listingEnd = 100`.
- A buyer comes and bids for the NFT before `listingEnd`.
- Now during the auction window the buyer can easily cancel their bid.
- After the `listingEnd`, if there exists a bid that was not cancelled, the seller (user A) has complete power to accept/ reject the bid.

The problem is that the buyer can no longer cancel their bid, so the entire power to settle the auction (either accept/reject) lies with the seller and if they wish, they can keep bidder's funds stuck at this point, griefing them from the ability to recover their funds.

This issue does not occur with other auction types (for English and Blind auctions, anyone can call `settleAuction()`), (for Dutch and Buy_it_Now auctions, it settles immediately during the bid).

The buyer's funds will be stuck until the seller wishes to cancel listing/ accept or reject the offer, though the seller's NFT will be stuck too.

Recommendation: Change the code to:

```
if(listings[listingId].listingType != ListingType.BUY_IT_NOW_WITH_BIDS) {
    if (block.timestamp > listingEnd) revert AuctionEnded();
}
```

This will allow the buyer to cancel their bid for `BUY_IT_NOW_WITH_BIDS` type even after the `listingEnd`.

KIM Exchange: Fixed in commit d58a0089.

Cantina Managed: In an edge case if the listing window gets completed, and the last remaining highestBid gets cancelled after that, the auction will result in nothing for the seller (except they can get their own NFT back). This is annoying but is considered as an acceptable risk as the seller has all the time to accept the highestBid if he wants to and settle the listing before the listing ends.

Nevertheless, the issue pointed out above has been fixed now.

3.3.3 Blind auctions cannot be cancelled before auction starts

Severity: Medium Risk

Context: KimNFTMarketplace.sol#L160-L162

Description: In the protocol, the maker is allowed to cancel the auction before it starts.

```
if (listing.start > 0 && block.timestamp >= listing.start && block.timestamp <= listing.end) revert
↪  AuctionNotFinished();
```

For blind auctions, however, due to the incorrect check, the maker is not allowed to cancel the auction before it starts.

```
if (listing.listingType == ListingType.BLIND) {
    if (block.timestamp <= (listing.end + BLIND_BID_REVEAL_WINDOW)) revert AuctionNotFinished();
}
```

Recommendation: It is recommended to change the code as follows to allow blind auctions to be cancelled before they start.

```
- if (block.timestamp <= (listing.end + BLIND_BID_REVEAL_WINDOW)) revert AuctionNotFinished();
+ if (block.timestamp >= listing.start && block.timestamp <= (listing.end + BLIND_BID_REVEAL_WINDOW)) revert
↪  AuctionNotFinished();
```

KIM Exchange: Fixed in PR 8.

Cantina Managed: The new implementation is as follows, before the Blind auction starts, `blindBidsForListing[listingId].length == 0` so it doesn't go into that branch, and the outer check allows the Blind auction to be canceled before it starts:

```
if (listing.listingType != ListingType.BUY_IT_NOW && listing.listingType != ListingType.BUY_IT_NOW_WITH_BIDS) {
    // Ensure in the middle of an auction we cannot cancel the listing
    if (block.timestamp >= listing.start && block.timestamp <= listing.end) revert AuctionNotFinished();

    // When blind auction enforce the bid reveal window to finish if there are bids present
    if (listing.listingType == ListingType.BLIND && blindBidsForListing[listingId].length > 0) {
        if (block.timestamp >= listing.start && block.timestamp <= (listing.end + BLIND_BID_REVEAL_WINDOW))
↪   revert AuctionNotFinished();
    }
```

3.3.4 Buyer can grief attack seller in english and blind auction

Severity: Medium Risk

Context: KimNFTMarketplace.sol#L443-L455

Description: After the English auction or Blind auction ends, if there are any bidders, the auction cannot be cancelled by the seller and anyone can call `settleAuction()` to send the NFT to the bidder to send the tokens to the seller.

When the NFT is sent, the bidder's callback function will be called, which gives the bidder the opportunity to grief the seller by refusing to receive the NFT.

This means a malicious bidder can "extort" the seller. Consider that the bidder is a contract that refuses to receive NFTs by default. Only by sending 1 ETH to the contract will the key variables of the contract be changed so that it can receive NFTs.

Recommendation: If the NFT transfer fails, the `settleAuction()` should end up sending the payment to the seller while leaving the NFT escrowed. It is recommended to provide a public function that allows the bidder to claim the NFT later.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix will return the NFT to the seller if the bidder rejects the NFT, it fixes the issue:

```
bool nftTransferSucceeded = _concludeSale(bestBid.user, bestBid.amount, true, listingId, listing, true);
if (!nftTransferSucceeded) {
    // Here the buyer refused to accept the NFT. We will protect the seller's NFT by allowing the transaction
↪   to conclude with seller safely getting their NFT back
    isNoLongerEscrowed[listingId] = true;
    _transferNFT(
        listing.nft,
        listing.maker,
        listing.tokenId,
        listing.amount,
        listing.isSemiFungible,
        false
    ); // Reentrancy protection from _transferNFT

    // Try to refund the user
    _refundUserBid(bestBid.amount, bestBid.user, listing.currency, listingId);
}
```

3.3.5 Immediately expired bid will replace good bid

Severity: Medium Risk

Context: Auction.sol#L150

Description: For listings of type `BUY_IT_NOW_WITH_BIDS`, the bidder bids with an `expiryTimestamp`, and when the bid expires, the bid will not be accepted.

Since the protocol does not limit the minimum valid time, a bidder can offer a higher bid with a 1-second valid time, which will replace the current best bid and expire immediately, which may prevent the seller from accepting a good bid.

Recommendation: It is recommended to require `expiryTimestamp` to be greater than a minimum duration from the current timestamp.

KIM Exchange: Fixed in PR 8

Cantina Managed: The fix implements the recommendation.

3.3.6 Blind auction bid amount can't resist brute-force attack

Severity: Medium Risk

Context: Auction.sol#L254

Description: When bidders enter the blind auction, they make a blind bid and then reveal the bid amount. When making blind bid, the protocol saves a hash for the bidder, and when revealing the bid amount, it requires that the bidAmount provided by the bidder meets the following conditions:

```
keccak256(encodeUserBid(bidAmount_, from, tokenId, nft)) == savedHash
```

The from, tokenId, and nft are all public, only bidAmount will be made public when the bidder reveals it. However, considering that bidAmount is non-random and related to the market value of the NFT, it is not resistant to brute-force attack.

For example, for an NFT with a market price of 1000 USDC, the bidders' bidAmount may be in the range of $[0, 1000]$ USDC, and considering that USDC is 6 decimals, `1e10` calculations will be enough to brute-force all the bidAmount, and considering that the user's bid may be in the smallest unit of 0.01 USDC, the required calculations will be fewer.

Recommendation: It is recommended to introduce a user-supplied random nonce to increase the strength.

KIM Exchange: Fixed in PR 8.

By using `mapping(uint256 listingId => bytes32[] bids) public blindBidsForListing;`,we allow 2 possibilities:

1. Sending multiple blind bids from a single account. For example, blind bid for 999, 1000 and 1001, 1002 but we only reveal one of them later.
2. Sending a blind bid from a different account and then revealing a blind bid later with the correct from account by specifying the correct blind bid index.

This means guessing a blind bid is much more difficult now.

Cantina Managed: The fix makes it more difficult to brute-force blind bids, especially since bidder becomes uncertain. As an additional measure, listingID was added to the hashing to protect against pattern matching revealed hashes later for future blind auctions.

3.3.7 USDC Blacklisted user can break bidding

Severity: Medium Risk

Context: KimNFTMarketplace.sol#L236-L239

Description: When there is a higher bid, or the listing is canceled, or the listing is bought directly, if there is an existing bid, the bid will be returned to the bidder. For some tokens, refunds are not always successful.

For example, for USDC, if a bidder is added to the USDC blacklist after bidding, sending USDC to the bidder will revert the transaction.

Especially for `BUY_IT_NOW_WITH_BIDS` type listing, since there is no need to set a reserve price, an attacker can bid 0.01 USDC, after which the listing cannot be canceled and the bid cannot be rejected.

This will result in the seller's NFT being frozen in the contract or only able to accept the attacker's bid.

Note: this kind of DoS (due to the push funds approach) also affects `revealBlindBid()` and all other flows leading to `_deleteBidAndReturnFunds()`.

Recommendation: If USDC (or other token) transfer fails, save the bidder, refunded token and amount, continue with the subsequent logic, and provide a public function to allow the bidder to claim the transferred failed tokens later.

KIM Exchange: Fixed in PR 8.

Cantina Managed: This fix sends the refund to the DAO when bidder rejects it, which fixes the issue.

3.4 Low Risk

3.4.1 `acceptBidForNFT()` should allow accepting the bid when `highestBid > bidAmount`

Severity: Low Risk

Context: KimNFTMarketplace.sol#L267-L277

Description: The `acceptBidForNFT()` function is for the seller to accept open bids on a `BUY_IT_NOW_WITH_BIDS` listing. It takes expected `bidAmount` as an input by the seller to protect them from frontrunning attacks that try to cancel the previous bid and put up a lower bid : before the accept call executes.

Right now this call reverts if the expected `bidAmount` entered by the seller is not equal to the current bid stored in `bestBidForListing()`. But this also reverts in case someone puts up a higher bid before the accept call executes.

The call should execute even if the current bid is greater than the expected `bidAmount` input by the seller when calling `acceptBidForNFT()`, because in such a case it only profits the seller and does not cause any problems.

Recommendation: Change the code to:

```
if (highestBid < bidAmount) revert BidChanged();
```

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.4.2 Dutch auctions should be sold using `currentPrice`

Severity: Low Risk

Context: Auction.sol#L227-L231

Description: By the definition of Dutch auction, the sold price should be the price that decays over time.

Especially if the bidder's transaction is delayed, the `currentPrice` would be better for the bidder, however the current implementation will still use the bidder's `bidAmount` to sell the Dutch auction.

Recommendation: It is recommended that Dutch auctions be sold using `currentPrice`, and the bidder's `bidAmount` is more of a slippage control.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.4.3 Best bid is replaced when bids are equal

Severity: Low Risk

Context: Auction.sol#L187

Description: When bidding, `bidAmount >= currentBid` is required, which means that even if the bids are the same, the new bid will become the best bid.

And even if `bidAmount > currentBid` is required, the user can add 1 wei to become the best bid. Since the prices aren't differentiated, it might not be a good experience for the user, since their bids can be easily exceeded.

Recommendation: It is recommended to set the minimum bid increase amount, such as 101% of the current best bid.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.4.4 Inconsistent bidding timelines

Severity: Low Risk

Context: KimNFTMarketplace.sol#L155-L162

Description: The codebase lacks clear separation of time windows for bidding and settling the auctions.

1. For English auctions, at `timestamp == listingEnd`, it is possible to `settleAuction()` but it is not possible to `cancelListing()`. `cancelListing()` should be allowed as well because the auction has ended, it will still correctly revert if there is an existing bid. Also, `cancelBidForNFT()` should not be allowed.
2. For Blind auctions, at `timestamp == listingEnd + BLIND_BID_REVEAL_WINDOW`, it is possible to `settleAuction()` but not to `cancelListing()`.`cancelListing()` should be allowed for conceptual consistency of an auction getting concluded (which will still correctly revert if there is an existing revealed bid after the reveal period ends). Also, `revealBlindBid()` should not be possible.

Recommendation: We recommend to establish clear separation of time windows for bidding and settling flows

KIM Exchange: Fixed in PR 8

Cantina Managed: Now block.timestamp == end of the period is considered to be as auction window finished, so for English auctions,

- For English auctions, at `block.timestamp == listing.end`, `settleAuction()` and `cancelListing()` are allowed and bidding activity (including `cancelBidForNFT()`) is prohibited.
- For Blind auctions, at `block.timestamp == listing.end + BLIND_BID_REVEAL_WINDOW`, `settleAuction()` and `cancelListing()` are allowed and bidding activity (including `revealBlindBid()`) is prohibited.

This concludes into a clear timeline separation for interactions. Fixed.

3.4.5 Sale window may not be respected for `BUY_IT_NOW` and `BUY_IT_NOW_WITH_BIDS` listing types

Severity: Low Risk

Context: KimNFTMarketplace.sol#L110-L117

Description: The `listing.start` and `listing.end` values are optional for `BUY_IT_NOW` and `BUY_IT_NOW_WITH_BIDS` listing types, but they need to be validated if they do exist.

The `listNFT()` logic currently allows these two types of listings to be opened with `start == 0` and `endTime > 0`. Such a listing can be opened if

- A creator may by mistake enter `start == 0` or...
- Creator does not bother putting in a start value if he wants the sale to start immediately but puts up a rational value for the `listing.End`.

This creates a problem because the sale window is only checked in `buyItNow()` and `bidForNFT()` if `listing.start > 0`. So the checks are skipped in the cases mentioned above.

Now, these listings can be sold even after the `listing.end` timestamp which is outside of the restricted window, leading to unexpected results for the creator.

Recommendation: In `listNFT()`, add a check to make sure that if `newListing.end > 0`, the `newListing.start` should also be `> 0`.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.4.6 Blind auction should be allowed to be cancelled before reveal period if no bids were placed

Severity: Low Risk

Context: KimNFTMarketplace.sol#L160-L162

Description: `cancelListing()` only allows blind auctions to be cancelled after the `BLIND_BID_REVEAL_WINDOW` has passed. This is an extra time duration after the actual bidding window - whose end is marked by `listing.end`.

This creates a problem when no blind bids have been recorded during the bidding window but the seller will still need to wait an extra 24 hours (which is the current `BLIND_BID_REVEAL_WINDOW` period) with no interactions as there will be no hashes to be revealed in such a case.

This will keep the seller's NFT being stuck in the marketplace when its not required, potentially leading to opportunity cost.

Recommendation: Allow the seller to cancel a blind auction after `listing.end` if no bid hashes have been recorded.

KIM Exchange: Fixed in PR 8

Cantina Managed: The new implementation is as follows, when there are no bids in the Blind auction, it can be canceled after the auction ends.

```
if (listing.listingType != ListingType.BUY_IT_NOW && listing.listingType != ListingType.BUY_IT_NOW_WITH_BIDS) {
    // Ensure in the middle of an auction we cannot cancel the listing
    if (block.timestamp >= listing.start && block.timestamp <= listing.end) revert AuctionNotFinished();

    // When blind auction enforce the bid reveal window to finish if there are bids present
    if (listing.listingType == ListingType.BLIND && blindBidsForListing[listingId].length > 0) {
        if (block.timestamp >= listing.start && block.timestamp <= (listing.end + BLIND_BID_REVEAL_WINDOW))
↪   revert AuctionNotFinished();
    }
```

3.5 Gas Optimization

3.5.1 Unnecessary check in `_deleteBidAndReturnFunds()`

Severity: Gas Optimization

Context: KimNFTMarketplace.sol#L466-L467

Description: The line here that checks `currentBid.user` and reverts if it is == address(0) is unnecessary because all flows leading to the function `_deleteBidAndReturnFunds()` already check and ensure that it is only called when currentBid.user is != address(0).

Recommendation: This check can be safely removed to save some gas.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.5.2 `bidAmount` validation can be optimized

Severity: Gas Optimization

Context: Auction.sol#L186-L189

Description: For English and `BUY_IT_NOW_WITH_BIDS` auctions, `_validateBuyItNowBidAndEnglishAuctionBid()` is called to validate that the bid amount is greater than the last bid as well as the reserve price.

This is the code:

```
function _validateBuyItNowBidAndEnglishAuctionBid(uint256 currentBid, uint256 bidAmount, uint256 reserve)
↪  internal pure {
    if (bidAmount < currentBid) revert BidTooLow();
    if (reserve > 0 && bidAmount < reserve) revert BidBelowReserve();
}
```

The reserve check is unnecessary if there is already a valid bid stored.

Recommendation: Check against reserve only if `currentBid == 0`. This will check the first bid against reserve and thereafter all new bids are required to be greater than currentBid so they will be automatically greater than reserve price.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.6 Informational

3.6.1 Modifier `onlyListing()` logic is wrong

Severity: Informational

Context: KimNFTMarketplace.sol#L70-L74

Description: The `onlyListing()` modifier is used on all user interactions to validate that the input `listingID` actually exists. It does so by checking if the `listingID` is greater than the actual array elements in the `listings` array.

But the current logic also considers array index == listings.length as a valid element, which can not exist as the array only has length - 1 elements.

This does not have much of an impact as the purpose is only to return a nice error instead of straight up array out of bound reverts.

Recommendation: Change the code to:

```
modifier onlyListing(uint256 listingId) {
    if (listings.length == 0) revert InvalidListing();
    if (listingId >= listings.length) revert InvalidListing();
    _;
}
```

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.6.2 `revealBlindBid()` should add premium check

Severity: Informational

Context: KimNFTMarketplace.sol#L308

Description: When bidders enter the blind auction, they make a blind bid and then reveal the bid amount.

When making blind bid, `bidForNFT()` will make standard check and premium check, while when revealing the bid amount, `revealBlindBid()` only make standard check, not premium check.

If the user reduces the allocation or sells the premium NFT before the reveal, he should not be allowed to reveal the bid.

Recommendation: It is recommended to add premium check in `revealBlindBid()`.

```
if (isPremiumNFT[listing.nft][listing.tokenId]) {
    _assertPremiumBuyerAllocatedXKIMAndOwnsAnyPremiumNFT(msg.sender, optionalPremiumTokenId);
}
```

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.6.3 Unnecessary `listing.start` check in `cancelListing()` can be removed

Severity: Informational

Context: KimNFTMarketplace.sol#L120-L125, KimNFTMarketplace.sol#L155-L157

Description: For English, Dutch and Blind auctions, the `cancelListing()` logic checks that the auction window is not currently running. However, this check also includes if `listing.start > 0`.

When listing an NFT for these types of auctions, `listNFT()` requires that the start time of listing is non-zero, so no English/ Dutch/ Blind auction can exist with a zero `listing.start`.

Recommendation: This check can be removed as its not required.

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.

3.6.4 `minXKimAllocationForStandardListings` can be equal to `minXKimAllocationForPremiumListings`

Severity: Informational

Context: KimNFTMarketplaceBase.sol#L225-L239

Description: In `_setMinXKimAllocationForStandardListings()`, `minXKimAllocationForStandardListings` can be equal to `minXKimAllocationForPremiumListings`, however in `_setMinXKimAllocationForPremiumListings()`, they are not allowed to be equal.

Recommendation: It is recommended to make them consistent. For example, change to

```
  function _setMinXKimAllocationForStandardListings(uint256 newMinXKimAllocationForStandardListings) internal {
      if (newMinXKimAllocationForStandardListings == 0) revert StandardErrors.ZeroValueArgumentSupplied();
-     if (minXKimAllocationForPremiumListings > 0 && newMinXKimAllocationForStandardListings >
↪  minXKimAllocationForPremiumListings) revert xKIMForStandardListingsCannotExceedPremium();
+     if (minXKimAllocationForPremiumListings > 0 && newMinXKimAllocationForStandardListings >=
↪  minXKimAllocationForPremiumListings) revert xKIMForStandardListingsCannotExceedPremium();
```

KIM Exchange: Fixed in PR 8.

Cantina Managed: The fix implements the recommendation.