ONYX LABS

Smart Contract Audit

Prepared For:
HeeDong
BBRC

Date:
3/17/23

Table of Contents

Summary

This report was prepared to summarize the findings of the audit performed for BBRC (HeeDong) of their Blind Auction, Deposit, and ERC721A Contract. The primary objective of the audit was to identify issues and vulnerabilities in the source code. This audit was performed utilizing Manual Review techniques and Static Analysis.

The audit consisted of:

- Static Analysis utilizing Slither Static Analyzer.
- Unit and Fuzz testing where appropriate.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Line-by-line manual review of the entire codebase by auditors.

The following report contains the results of the audit, and recommendations to guard against potential security threats and improve contract functionality.

We would like to thank the BBRC team for their business and cooperation; their openness and communication made this audit a success.

Overview

Project Summary

Project Name	HeeDong
Description	HeeDong is an ERC721 Mint, with a sale being performed prior to mint that includes a Blind Auction and Deposit/Reserve system
Blockchain	Ethereum
Language	Solidity
Codebase	BlindAuction.sol, Waitlist.sol, NFT.sol
Commit	N/A

Audit Summary

Delivery Date	3/17/23
Audit Methodology	Static Analysis, Manual Review, Unit Testing

Finding Summary

Finding Level	Total	Pending	Acknowledged	Resolved
Critical	1	0	0	1
High	1	0	0	1
Medium	5	0	0	5
Low	6	0	3	3
Informational	5	0	3	2

Audit Scope

ID	File
B	BlindAuction.sol
W	Waitlist.sol
N	NFT.sol
U	BlindAuction.sol (re-commit)

Findings

ID	Title	Category	Severity	Status
B-001	Tracking of total deposited value	Optimization	Low	Resolved
B-002	Msg.sender in verifyWonItemsCount	Functional	Low	Resolved
B-003	No max value in wonItemsCount	Security	Low	Acknowledged
B-004	Unused Variables	Quality	Informational	Acknowledged
B-005	Logical inaccuracy in withdraw	Functional	Low	Resolved
B-006	Lack of zero address check	Security	Medium	Resolved
B-007	Msg.sender in getUserBid	Functional	Low	Acknowledged
B-008	View function access control	Functional	Informational	Acknowledged
B-009	No min value in setFinalPrice	Security	Medium	Resolved
W-001	Unused Variables	Quality	Informational	Resolved
W-002	Tightly pack struct Reserve	Optimization	Medium	Resolved
N-001	Unused Variables	Quality	Informational	Resolved
N-002	MAX_SUPPLY not immutable	Optimization	Low	Acknowledged
N-003	Redundant URI strategy	Functional	Informational	Acknowledged
N-004	Use of totalSupply	Security	Medium	Resolved
N-005	Staked tokens still transferable	Functional	High	Resolved
U-001	Tightly pack struct Bid	Optimization	Medium	Resolved
U-002	userBid not committed to storage	Functional	Critical	Resolved

I’ve

B-001 | Tracking of total deposited value

Category	Severity	Location	Status
Optimization	Low	BlindAuction:165	Resolved

Description

In the function `commitBid`, msg.value is added to a global tracker `totalDepositedETH`.

This value is not used anywhere else in the contract; given this, and the fact that address(this).balance will return the total amount of ETH deposited into the contract, it is unnecessary

Recommendation

Remove `totalDepositedETH` from the contract.

B-002 | Msg.sender in verifyWonItemsCount

Category	Severity	Location	Status
Functional	Low	BlindAuction:137	Resolved

Description

Function `verifyWonItemsCount` uses msg.sender to verify the Merkle proof provided is valid. However, the function is marked public. Without the ability to input arbitrary addresses the function cannot be used for simple “sanity checks” of Merkle proofs.

Recommendation

Add an address parameter to `verifyWonItemsCount` and use it in place of msg.sender.

B-003 | No max value in wonItemsCount

Category	Severity	Location	Status
Security	Low	BlindAuction:227	Acknowledged

Description

In function `refund` the parameter `wonItemsCount` is used to verify the number of tokens won by a refunder. This value does not have a maximum, and could result in users being under-refunded if an error in the Merkle tree exists.

Recommendation

Add a maximum value to `wonItemsCount` based on the maximum number of tokens that can be won in the blind auction.

B-004 | Unused Variables

Category	Severity	Location	Status
Quality	Informational	BlindAuction:49	Acknowledged

Description

Variable `MAX_BID_QUANTITY` is defined in the contract but never used.

Recommendation

Remove `MAX_BID_QUANTITY`.

B-005 | Logical inaccuracy in withdraw

Category	Severity	Location	Status
Functional	Low	BlindAuction:260	Resolved

Description

In the function `withdrawSales`, local variable `withdrawed` is set equal to `sales` but should be set equal to `available`.

Recommendation

Change `withdrawed = sales` to `withdrawed = available`

B-006 | Lack of zero address check

Category	Severity	Location	Status
Security	Medium	BlindAuction:101/285	Acknowledged

Description

Variable `financeWalletAddress` is used to determine which address receives funds withdrawn from the contract. This variable is set upon initialization and in the function `setFinanceWalletAddress`. Neither of these functions include a check to ensure that `financeWalletAddress` is not set to the zero address. This could lead to lost funds.

Recommendation

Add a check to ensure that `financeWalletAddress` cannot be set to the zero address.

B-007 | Msg.sender in getUserBid

Category	Severity	Location	Status
Functional	Low	BlindAuction:188	Acknowledged

Description

Function `getUserBid` uses msg.sender to return a user’s total bid. Without the ability to input arbitrary addresses the function cannot be used to check any user’s bid.

Recommendation

Add an address parameter to `getUserBid` and use it in place of msg.sender.

B-008 | View function access control

Category	Severity	Location	Status
Functional	Informational	Several	Acknowledged

Description

Several view functions are marked as `onlyOwner` functions. This pattern is atypical, and does not actually restrict anyone’s ability to read the function.

Recommendation

Remove the `onlyOwner` modifier from view functions.

B-009 | No min value in setFinalPrice

Category	Severity	Location	Status
Security	Medium	BlindAuction:315	Resolved

Description

The function `setFinalPrice` is used to set the final price of the auction, signifying what users will pay for each token. The final price should not be below the variable `MIN_BID_AMOUNT`, but `setFinalPrice` does not prevent the sender from entering a lower value.

Recommendation

Add a requirement that prevents a value lower than `MIN_BID_AMOUNT` from being entered into `setFinalPrice`.

W-001 | Unused Variables

Category	Severity	Location	Status
Quality	Informational	Waitlist:64	Resolved

Description

Struct `Transaction`, as well as variables `transactions` and `participantTransactions` are unused.

Recommendation

Remove `Transaction`, `transactions`, and `participantTransactions`.

W-002 | Tightly pack struct Reserved

Category	Severity	Location	Status
Optimization	Medium	Waitlist:82	Resolved

Description

Struct `Reserved` can be packed tightly by modifying the timestamp variable.

Recommendation

Change `Reserved.timestamp` to uint80 from uint256.

N-001 | Unused Variables

Category	Severity	Location	Status
Quality	Informational	NFT:38	Resolved

Description

Variable `MAX_TIME` is not used.

Recommendation

Remove the variable `MAX_TIME`.

N-002 | MAX_SUPPLY not immutable

Category	Severity	Location	Status
Optimization	Low	NFT:36	Acknowledged

Description

Variable `MAX_SUPPLY` is unlikely to change after deployment, but is not set as an immutable value.

Recommendation

Label `MAX_SUPPLY` as immutable and set it in the constructor.

N-003 | Redundant URI strategy

Category	Severity	Location	Status
Functional	Informational	NFT:40	Acknowledged

Description

The contract’s current URI strategy includes a `prerevealedURI` as well as a `baseURI`. As baseURI will need to be set at the time that the final artwork is revealed, the use of prerevealedURI is redundant, and adds additional steps to the reveal process.

Recommendation

Remove the variable `prerevealedURI` and simply set `baseURI` as the pre-revealed artwork until ready to change the artwork to its final form.

N-004 | Use of totalSupply

Category	Severity	Location	Status
Security	Medium	NFT:159	Resolved

Description

In the function `mint`, `totalSupply()` is compared to `MAX_SUPPLY`. Additionally, the contract includes a function to burn tokens. In ERC721A, any burnt tokens decrement the value of `totalSupply()`, which would allow additional tokens to be minted in the future if tokens are burnt.

Recommendation

Replace the use of `totalSupply()` with `_totalMinted()`.

N-005 | Staked tokens still transferable

Category	Severity	Location	Status
Functional	High	NFT:253	Resolved

Description

The contract includes an override of the function `transferFrom` that prevents tokens which are “staked” from being transferred from the owner’s wallet. However, `safeTransferFrom` has not been overridden with the same restriction, meaning that “staked” tokens could be transferred.

Recommendation

Apply the same override logic from `transferFrom` to `safeTransferFrom`.

U-001 | Tightly pack struct Bid

Category	Severity	Location	Status
Optimization	Medium	BlindAuction:60	Resolved

Description

Struct `Bid` can be more tightly packed by modifying the variables `createdAt` and `updatedAt` to be uint32 instead of uint256 and moving `amount` to the bottom of the struct.

Recommendation

Restructure `Bid` in accordance with the recommendations above.

U-002 | userBid not committed to storage

Category	Severity	Location	Status
Functional	Critical	BlindAuction:177	Resolved

Description

In the function `commitBid`, the updated userBid variable is not committed to storage, thus not saving the user’s bid data. This could result in erroneous loss of user funds, or a very difficult process of recreating all userBid data post auction.

Recommendation

Add `userBids[msg.sender] = userBid` after updating userBid with the user’s bid data.