Smart Contract issues and changes

Issue #	Description	Severity	Status
1	Overall architecture may imply major security risks	Warning	
2	No test coverage	Warning	Tested
3	Anyone can mint tokens from any batch at an arbitrary or zero price	Critical	Fixed
4	Any bidder can block subsequent bids from being accepted	Critical	Fixed
5	closeBid() does not close bidding	Critical	Fixed
6	Sale and Auction at the same time can cause bidder’s funds to be locked forever	Critical	Fixed
7	Sellers can be left without any payment if fees are set high	Major	Fixed
8	The contract owner can withdraw all funds at any time	Major	Fixed
9	Bids can be added even after auction end time	Major	Fixed
10	Token owner will lose funds for closing an auction with no bids	Major	Fixed
11	Auction end time can be set in the past	Minor	Fixed
12	Duplicate over-/underflow protection	Minor	Fixed
13	Use call() instead of transfer()	Minor	Fixed
14	Inefficient data-structures	Note	
15	Include revert messages	Note	Fixed

Warnings

1. Overall architecture may imply major security risks

The design of the system is based on an architecture that groups a large number of functionalities into a single smart contract. The decision to add token collection, auction, and sale functionality into the token smart contract, which also holds Ether value, increases the attack surface considerably. This design increases the likelihood of undetected issues.

Recommendation
Consider re-designing the protocol in a more modular fashion.

Fix

Going ahead with the same structure

2. No test coverage

The codebase lacks evidence of test coverage and the number of issues found indicates that testing has been very limited. This adds a risk of additional issues since there are a number of vulnerability types that are easy to detect in unit testing but harder to spot in a security audit.

Recommendation
Consider providing unit tests covering all functions and branches.

Fix

Tested on our end from remix on most cases

Critical Issues

3. Anyone can mint tokens from any batch at an arbitrary or zero price

The `mintToken()` function allows anyone to mint unclaimed tokens, even if the caller is not the batch creator. In addition, the payment for this operation is set by the caller and can be set as desired, even to zero. The fee deduction process indicates that minting should be payable operation, however, no minimum price is enforced.

Recommendation
Clarify the mint functionality and either enforce a minimum price or restrict this operation to the batch creator.

Fix

When openCloseMint is true, the caller can do open minting when they enter a value greater or equal to the price set by the batch creator. If openCloseMint is false, caller cannot mint because they are not the creator. Also have a require to put sufficient money to openmint.

4. Any bidder can block subsequent bids from being accepted

A bidder can block subsequent bids from being accepted with a Denial Service attack, winning the auction by default. This is due to the `transfer()` call in `addBid()` that pushes refunds out to the previous highest bidder. The bidder can call the `addBid()` function from a smart contract. This contract can be implemented to revert when receiving the refund payment, causing the whole transaction to fail. This will block any other users from placing a higher bid. In a similar manner, a bidder may block an auction to be closed. However, this is less critical, since there would not be an incentive to do so.

Recommendation
Do not push out refund payments. Instead, a pull pattern is recommended in which refunds have to be claimed in a separate transaction.

Fix

We use a userBalances mapping that has a wallet address mapped to whatever that person has to claim. Any user can claim his balances from withdrawuserbalance function.

5. closeBid() does not close bidding

The `closeBid()` function never sets the `bidding` flag to `false`, meaning that auctions remain open. This allows anyone to create a new bid for that auction for as low as 1 wei and call the `closeBidBuyer()` method to claim the token.

Recommendation
Set `bidding` to `false`.

Fix

Set bidding to false

6. Sale and Auction at the same time can cause bidder’s funds to be locked forever

A token on sale can be auctioned and vice-versa. This locks in the last bidder funds forever if an item on auction is sold.

Recommendation
Check the auction or sale status before initiating the other operation

Fix

Token owners cannot list tokens for sale on the same contract when the token is on for auction. If a token is purchased on OpenSea, we have a new function called beforeTokenTransfer where it would transfer the last bidder’s funds back to them when a token transfer happens and reset all the mappings to its default values.

Major Issues

7. Sellers can be left without any payment if fees are set high

There is nothing that prevents the total fees (royalties and operator fees) to be set to 100%, meaning sellers only receive any payment. This is even the case with the percentages used as examples in the comments in the code.

Recommendation
Limit fees.

Fix

We did not want to limit the seller to any royalty fee but we will be showing the royalties percentage to the buyers. After a sale or bid happens, token creators cannot change the royalties.

8. The contract owner can withdraw all funds at any time

The contract owner can withdraw funds at any time, leaving bidders and sellers without any means of receiving their proceeds.

Recommendation
Limit the owner's privileges to reduce funds.

Fix

Function to withdraw all funds removed

9. Bids can be added even after auction end time

The function `addBid()` does not check the auction end time and allows anyone to add a bid even after the set time.

Recommendation
Verify the bid end time.

Fix

Users cannot add bids after timer ended

10. Token owner will lose funds for closing an auction with no bids

In order for the owner to close a bid using the `closeBid()` method, it requires transferring the `bidPrice` to the bidder. In the case of an auction with no bidders, the owner can close the bid only if the specified `bidPrice` is transferred to zero address.

Recommendation
Avoid the transfer if no bidders are present for an auction.

Fix

Owner does not need to transfer any funds to a close auction that has ended with no bidders.

Minor Issues

11. Auction end time can be set in the past

The `startBid()` function does not check the start timestamp when setting up an auction, allowing for any time to be used, including start times in the past. In such an auction, the first bidder could just immediately close the auction and win it. In addition, there is no bound on the maximum duration of an auction,

Recommendation
Enforce auction end times to be in the future and consider adding a maximum duration for an auction.

Fix

Auction end times can only be in the future. Maximum duration is set to 31 days. This applies to both auction with and without countdown. If countdown is true, timer will only start after the first

bid whereas if countdown is false, auction will start immediately and end after we reach the endTimestamp.

12. Duplicate over-/underflow protection

The code uses the `safeMath` library for the protection of arithmetic operations. However, it also enforces Solidity version 0.8.0 or higher, which already provides overflow protection. This causes unnecessary code execution, resulting in a higher gas cost.

Recommendation
Remove `safeMath` or disable built-in overflow protection.

Fix

Not using safemath

13. Use call() instead of transfer()

The codebase uses the `transfer()` for ETH payments. However, since the Istanbul hard fork, the use of `transfer()` is discouraged, since the gas stipend forwarded may not be enough anymore for certain basic operations in smart contract wallets.

Recommendation

Consider using `call()` for ETH transfers. Additional reentrancy protection needs to be considered.

Fix

Using call()

Informational Notes

14. Inefficient data-structures

The code uses mappings that map keys to hatches. However, the hashes could act themselves as keys, resulting in unnecessary storage and indirections. In addition, the off-chain created hashes are not really necessary in the contract and

Recommendation
Remove `safeMath` or disable built-in overflow protection.

Fix

Still using mappings

15. Include revert messages

It is recommended to add revert messages to all possible revert conditions for better usability.

Fix

We added revert messages to the smart contract