G0 Group

Security Review of

Gnosis EasyAuction

February 2021

Gnosis EasyAuction / February 2021

Files in scope

All solidity files in:

https://github.com/gnosis/ido-contracts/tree/0f9069ed360ec2fdadd490e50fd6b1c4232ff25d

Additional minor update introduced in the following commit has been also audited and doesn't introduce any issues:

https://github.com/gnosis/ido-contracts/commit/33b35e7e294b57ef7fcdd27672ac99f672b99336

Current status

All found issues have been fixed.

Report

Issues

1. Additional orders can be inserted among already processed orders

Severity: critical

`settleAuctionAtomically` allows to insert orders (multiple because of possible reentrancy) before `auctionData[auctionId].interimOrder` set in `precalculateSellAmountSum` this is very bad because these orders will be redeemable without being considered in the settlement, allowing for more auctioning tokens to be withdrawn than have been put in.

status - fixed

The issue is no longer present in

https://github.com/gnosis/ido-contracts/tree/ae40c2abd7fcf9b2de79f8dbd75e16e5187c0243

2. User with id 0 can create orders that will allow them to break the auction accounting

Severity: critical

Because there can be user with id `0` it's possible to enter and redeem `clearingPriceOrders` like this one . This allows for more tokens to be withdrawn than have been put in.

status - fixed

The issue is no longer present in

https://github.com/gnosis/ido-contracts/tree/ae40c2abd7fcf9b2de79f8dbd75e16e5187c0243

3. By careful order manipulation, it's possible to convince the system auction hasn't been fully filled while it has

Severity: critical

It's possible for this condition to pass even for fully filled auctions due to rounding on line leading to `currentBidSum` ending up equal as `minAuctionedBuyAmount` on line . This is bad because `volumeClearingPriceOrder` will be `0` leading to both auctioneer and participants being able to redeem the whole amount of auctioning tokens.

status - fixed

The issue is no longer present in

https://github.com/gnosis/ido-contracts/tree/ae40c2abd7fcf9b2de79f8dbd75e16e5187c0243

4. Potential minor DoS attack vector in precalculateSellAmountSum

Severity: minor

If `iterationSteps` in `precalculateSellAmountSum` is overshot, the call will throw, it would probably improve usability if it would just process as many orders as possible (or iterationSteps) and then return. This will make constructing calls easier, but will also simplify coordination between multiple callers. This will also prevent a DoS attack where after a transaction calling `precalculateSellAmountSum` is submitted, someone frontruns it with their own call that has `iterationSteps == 1` making the `iterationSteps` in the first call overshoot by one and the transaction fail.

status - acknowledged (dev's response follows)

We decided against it. Gas costs increase too much, by over 10%.https://github.com/gnosis/ido-contracts/pull/49#issuecomment-767565722 We believe that it is not ddos-able, as the precalculateSellAmountSum can be called in several tx's with iterationstep size of 1000. If the attacker wants to ddos, they have to run precalculateSellAmountSum themselves, such that less than 1000 iteration steps are left. But, in this case, the benign submitter can finish the calculation via settleAuction.

5. Auctioneers can achieve a strictly better outcome of some auctions by bidding

Severity: minor

In some cases it's beneficial to the auctioneer to place an additional order just before the auction is closed to reduce the price of the auction without reducing the amount of bidding token they receive, this concerns specifically this branch: . Instead of the price being raised so that the bids cover the whole auctioning amount, the auctioneer basically cancels part of the sale by buying the tokens themselves. It might be better to make this behavior the default so auctioneers don't have to monitor the state to achieve the optimal outcome.

status - acknowledged (dev's response follows)

We share your concern of this economic analysis. There could be situations in which the auctioneer would benefit from participating in the last second. Though we decided against it for 3 reasons:

- The biggest use-cases for the auctions are expected to be IDO's and liquidations. If projects end up with some of the tokens that they actually wanted to auction off, it will be a cumbersome experience, as they might have to restart another auction.
- If the default behavior would be changed, the auction might be less attractive to potential buyers and this might be counter productive for the auctioneer.
- Practically, we don't expect a significant price difference for the auctions due to this effect.

Notes

Following notes only concern gas efficiency and have no security implications. All notes have been addressed by the developer.

- https://github.com/gnosis/ido-contracts/blob/df739271af1d11a1faf240d8363035673a213513/contracts/EasyAuction.sol#L31 this condition is redundant since orderCancellationEndDate is `always <= auctionEndDate` so checking orderCancellationEndDate is enough
- https://github.com/gnosis/ido-contracts/blob/df739271af1d11a1faf240d8363035673a213513/contracts/EasyAuction.sol#L44 this should probably either be `block.timestamp >= auctionEndDate &&` or the condition in `atStageOrderPlacement` should include `auctionEndDate`
- the two for loops in `_placeSellOrders` could be merged into one
- https://github.com/gnosis/ido-contracts/blob/df739271af1d11a1faf240d8363035673a213513/contracts/EasyAuction.sol#L256 this should be a stack variable preloaded outside of the loop
- https://github.com/gnosis/ido-contracts/blob/df739271af1d11a1faf240d8363035673a213513/contracts/EasyAuction.sol#L570 this return value is unused
- `initialAuctionOrder` is loaded twice in the `settleAuction` call here: https://github.com/gnosis/ido-contracts/blob/df739271af1d11a1faf240d8363035673a213513/contracts/EasyAuction.sol#L397 and here: https://github.com/gnosis/ido-contracts/blob/df739271af1d11a1faf240d8363035673a213513/contracts/EasyAuction.sol#L573