potuz:

Builder collusion

In this scenario the proposer and builder of N

are colluding to reorg the proposer of N+1

. The proposer sends his block late and does not reveal the attestations until very late (say when the block for N+1

is being produced), and the builder sends his payload supporting this block. The proposer of N+1

will see the PTC vote beforehand. So assuming he has seen the PTC vote but not the attestations for it and honest validators voted for N-1

, he will reveal his block based on N-1

if

1-\beta > RB

1−β>RB1-\beta > RB

After the reveal happens, his block will be reorged if

RB + \beta > PB + 1 - \beta \Leftrightarrow RB > PB + 1 - 2 \beta

RB+β>PB+1−β⇔RB>PB+1−2βRB + \beta > PB + 1 - \beta \Leftrightarrow RB > PB + 1 - 2 \beta

Combining with the above we get

PB < \beta

PB<βPB < \beta

Thus as long as PB \geq \beta PB≥βPB \geq \beta

, we can prevent ex-anti reorgs. That is, under the assumption of builder's collusion, the situation reverts to the current ex-anti reorg analysis pre-ePBS.

This is not the best attack. If it was, the (block, slot) fork-choice would obviate the need for proposer boost, but it doesn't. The proposer of N could send their block so as to target a 50/50 split between their block and the block of N-1 (votes for the empty slot), which essentially nullifies the votes of slot N, except for the \beta

votes controlled by proposer. Without RB

, we are back to the normal proposer boost math where we need PB > \beta

to prevent an ex-ante reorg of length 1. With RB

, we need PB > \beta + RB

, and there shouldn't be any good parameter space left.

Another issue is, PB = 20

instead of PB = 40

makes ex-ante reorgs of length > 1 much easier, and makes balancing attacks easier.