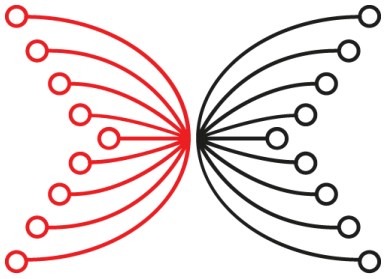
Security Issue Report



“Blocks from the near Future” bug

Severity: Low-Medium

Authors: Alexander Esgen

Date of Report: Feb 9, 2024

Table of Contents

1 Revision History

Version	Date	Change	Reason
1.0	2024-02-09	Created from Draft	Formalise Report
1.1	Feb 9, 2…	Expand on mitigation	Feedback by Karl Knutsson

2 Summary of Issue

Date of Issue Ongoing. The issue has existed since the removal of the isSelfIssued tiebreaker after the Vasil HF.
Criticality Level Medium. If unfixed and exploited, the issue could have resulted in a significant loss of chain density on mainnet (a Denial of Service attack). The actual impact was relatively small.
Context Blocks from the near future could cause nodes to mint orphan blocks for several minutes, distorting the chain production, and leading to potential gaps in the chain.
How was the Issue Detected The issue was detected by SPOs as a performance bug, manifested as a loss of block production up to several minutes. Karl Knutsson (CF) made the Consensus team aware of this by opening the GitHub issue #4251.
What Action was Taken ● The issue was investigated by the Consensus and Networking teams, including Cardano Foundation staff ● A fix was produced and deployed in node version 8.8
Potential Effect Investigation showed that the issue could potentially be exploited by an adversarial actor to lower the overall chain density.

Actual Effect

- Slight reduction in chain density (some “outages” of several minutes)

Ongoing Mitigations Needed, if any

Release node 8.8 or later to mainnet, monitor uptake, and advise SPOs to upgrade.

In case this issue is actively exploited before node 8.8 or later is widely deployed, advise SPOs to restart their nodes at certain points in time, see below.

Responsibility for Mitigations

Core Tech/IOI

3 Fixing the “blocks from the near future” DoS vector

3.1 High-level context

In all cardano-node versions prior to the (upcoming) cardano-node 8.8, the Consensus layer’s treatment of “blocks from the near future” has undesirable consequences when such an “early” block is propagated through the network. Specifically, in such an event, the network will with a good chance not extend the chain with a new block for several minutes; rather, all blocks minted until the situation resolves itself will be orphaned and lost.

So far, this situation has been triggered multiple times on the Cardano mainnet, recently, for example on the blocks with number 9361213 (no blocks for ~6.5 min) or 9861109 (no blocks for ~6 min). It seems unlikely that this was intentional; rather, it is probable that the responsible SPO did not configure their system to have an adequately synchronized clock.

It should be noted that it would be very easy for an attacker with a modest amount of stake to bring about this situation in many of the slots they are elected in. Without any response, this will lead to a significant decrease in the density of the chain. However, there is a relatively simple mitigation: SPOs should restart their nodes at certain points in time.

The purpose of this document is to describe how this attack vector has been resolved in the Consensus layer that is currently being integrated into the upcoming cardano-node 8.8 release, and how this fix has been validated to prevent this kind of situation from happening again on mainnet.

3.2 Technical details of the DoS vector

Background on admissible clock skew

A node considers a block to be *from the future* if its slot is ahead of the current slot. Ouroboros Praos mandates that all chains containing blocks from the future (at that time) are ignored during chain selection.
As Praos assumes that all nodes have access to perfectly synchronized clocks, this will never cause nodes to disregard blocks that have been minted by other honest nodes. In an actual real-world deployment, this assumption is unrealistic due to the imperfections of protocols like NTP as well as leap seconds.
Even in the real world, the magnitude of clock skew between honest (in particular well-configured) nodes can reasonably be assumed to be bounded; for this, an *admissible clock skew* of 5 seconds is currently being used[1]. At a high level, the implementation distinguishes between blocks from the future to be either from the *near* or the *far* future.

- A block is from the *near* future if the onset of its slot is ahead of the wall clock, but only by at most the admissible clock skew. Despite being from the future, these blocks are assumed to potentially have been minted by honest nodes. They cause the DoS vector that is discussed in this document.
- A block is from the *far* future if the onset of its slot is ahead of the wall clock by more than the admissible clock skew. By assumption, these blocks cannot have been minted by an honest node. These blocks are not relevant for the purpose of this document.[2]

Prior to cardano-node 8.8, the node processed a new incoming block from the near future like this:

1. The header and then block are downloaded via the ChainSync and BlockFetch mini-protocols.
2. The block is added to the ChainDB for chain selection.
    a. Before being validated, if the necessary preconditions are fulfilled (usually very likely), it is set as the tentative header for diffusion pipelining, and hence diffused through the network.
    b. *After* being validated, the in-future check is performed. As the block is from the near future, it is not selected, but rather added to a queue of blocks from the near future for later reconsideration.
3. Once another block is added to the ChainDB (either externally or because the node minted a block), this queue of blocks from the near future is reprocessed *before* the new block is considered. If the block is no longer from the future, it can be selected.

[1] As advised by the Network team, we will soon reduce this value to 2 seconds.
[2] Note that their handling pre-8.8 is also deficient and could also be used for a (different) DoS; this has also been fixed in 8.8.

Background on the chain order

For all non-experimental Shelley-based eras, the node uses the following chain order:

1. Compare block numbers, preferring higher values. If inconclusive, continue.
2. Tiebreaker based on opcert numbers, not relevant here.
3. Compare a specific[3] VRF value, preferring lower values.

Note that in the best case[4], the VRF value is uniformly random and can't be influenced by the issuer.

Before the Vasil HF, there was an additional tiebreaker ("isSelfIssued") that caused nodes to prefer blocks that they themselves minted. This was removed due to the need for an objective chain order for diffusion pipelining.

The DoS dynamics

Now consider what happens if a pool mints early (e.g. due to misconfiguration or being adversarial) on top of the currently best chain in the system. Suppose that the resulting block B has a strong (e.g. lower than the 90% percentile) VRF tiebreaker.

1. The block B is propagated throughout the entire network due to diffusion pipelining.[5]
2. B is not selected by any node, as it is from the near future, and is instead stored in the queue of blocks from the near future in every node.
3. Any node that is elected next will mint a *competitor* C to B (i.e. B and C have the same block number and parent). C is added to the ChainDB, which triggers chain selection for all blocks in the queue for blocks from the near future (i.e. just B). Hence, B is selected (if no longer from the future, but this is usually the case).
    a. If C has a better tiebreaker than B (unlikely, as we assumed B to have a strong tiebreaker), then it will be selected and the DoS resolved itself.
    b. Otherwise, C will not be selected, so C is orphaned and the network never learns of the existence of C. It is likely that the next elected node will *again* mint a losing competitor to B.
4. Note that no node in the network will adopt a chain of length blockNo(B) + 1 until one of the following two events happen:
    a. A node mints a competitor C that wins against B as in 3.a, and then someone mints on top of C (on average after 1/f = 20 slots).
    b. A single node mints twice; first, a competitor losing against B, and then a block on top of B.

It can take several minutes until one of the events described in 4. actually occurs. According to simulations by the Consensus team, the durations observed on mainnet are roughly what is to

[3] Which one is used exactly differs between TPraos and Praos, but this is not relevant here.
[4] In the worst case, an adversary is engaging in a procedure called *grinding* and can let luck be somewhat in their favor.
[5] Even without diffusion pipelining, it seems relatively easy for an attacker with a decent amount of stake to be considered as an upstream node for a large percentage of the entire stake.

be expected probabilistically. Note that the exact length of the block production outage (and hence the reduction in chain density) depends on various factors, in particular:

- The relative strength of the VRF tiebreaker of B (the lower, the longer).
- The stake distribution in the network (generally, the more widely the stake is distributed across smaller pools, the longer).

In addition, it is relevant how much stake is controlled by the parties with fast clocks as this guides *how often* this problem can occur.

3.3 Mitigation

In case this issue is actively exploited before node 8.8 or later is widely deployed, its impact on chain density can be limited by advising SPOs to restart their block-producing nodes at certain points in time. The queue of blocks from the near future is not persisted to disk, so restarting it will clear it (when the blocks in it are no longer from the future).

Concretely:

- SPOs shall restart their node such that it comes up again shortly before they are leading a slot. This is possible because the leader schedule is known in advance.
- SPOs shall restart their node a few seconds after they receive a block from the near future with a strong (low, for example lower than the 80% percentile) VRF tiebreaker.
- SPOs that operate a backup node can use it to perform a zero-downtime restart every few minutes to be on the safe side.

3.4 The bug fix

The fix (implemented in #525) changes the handling of a block from the near future like this:

- When receiving a header from the near future in ChainSync, an artificial delay is introduced until the header is no longer from the future.
- Only then, it is validated and the corresponding block body is downloaded and added to the ChainDB for chain selection, where it is not considered to be from the future due to the previous artificial delay.

The DoS is avoided as the entire handling of blocks from the (near) future in the chain selection logic is now effectively dead code[6]. In particular, nodes won't mint orphan blocks in the particular way described above anymore.

This approach is consistent with Ouroboros Chronos: While the main goal of Chronos is to describe a permissionless alternative to clock synchronization via e.g. NTP, it also specifies a rule for how to handle blocks from the future; namely, by setting them aside until the local clock reaches the onset of the slot.

[6] The Consensus team will remove the corresponding code in the future, but intentionally kept the bug fix as minimal as possible, in particular to avoid larger attention.

It should be noted that the potential incentive effects of this change are not yet fully understood and are the subject of ongoing work by the IOG research team.

3.5 Validating that the bug fix addresses the DoS vector

While #525 introduced component-level tests to ensure that the new mechanism behaves as expected, it is desirable to have concrete evidence that a node sending blocks from the near future will now no longer result in the emergent DoS vector previously described.
To this end, we ensured that, comparing cardano-node 8.8.0-pre[7] with 8.7.3, we no longer see a significant decrease in chain density when a pool is sending blocks from the near future.

Concretely, we used scripts/babbage/mkfiles.sh with a minor modification:

- Three pools with equal stake.
- Active slot coefficient f = 1/10 and 0.1s slot length, so on average 1 block/second.
- We applied a small patch in order to easily shift the clock of a node. Concretely, the node respects the CARDANO_CLOCK_SHIFT environment variable (a duration in seconds), which will be added on top of the "real" wall clock time as given by the operating system. In our case, we sometimes shifted the clock of pool 3 by 0.1s (one slot), such that the clock of that node was fast, and blocks minted by it are considered to be from the near future by the other nodes.
- Basically no propagation delays, due to localhost networking.

We ran the local cluster for half an hour (a sufficiently long time to reduce the impact of the leadership schedule randomness) for each of the following configurations:

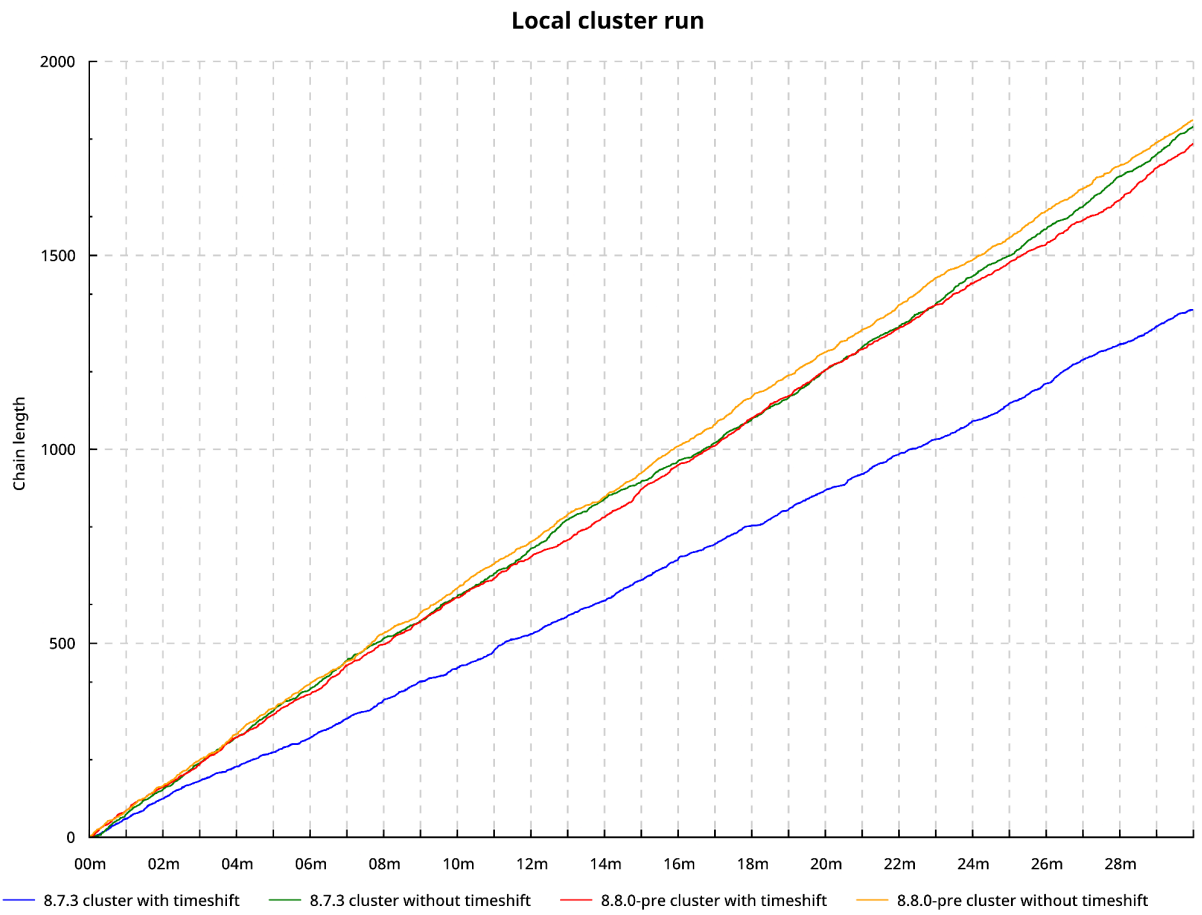
1. 8.7.3, no shifted clocks
2. 8.7.3, fast clock by 0.1s for just pool 3
3. 8.0.0-pre, no shifted clocks
4. 8.0.0-pre, fast clock by 0.1s for just pool 3

Based on the analysis above, we expected and indeed observed the following outcomes:

1. Perfect chain density of 1 block/second.
2. Significantly lower chain density due to the DoS vector.[8]
3. Perfect chain density of 1 block/second, as in 1.
4. Almost perfect chain density: Occasionally, pool 3 (the pool with the fast clock) will be elected in a slot s, and another pool P will be elected in s-1. Then pool 3 might not see the block that pool P minted due to its fast clock, and hence create an unnecessary competitor, resulting in an orphaned block.

Now consider the chain length in the different cluster scenarios over time:

[7] Concretely, commit ba233b0a0482bcbf4fd9b67173c16f52d6e74e86 from #5648
[8] Note that compared to mainnet, the amount of stake that is minting blocks from the future is very high (which increases the expected chain density drop), but the stake is much more centralized (which decreases the expected chain density drop).



(Actually, every cluster run consists of three lines, one for each pool, but this is not visible from this scale.)

Observations:

- The cluster runs without shifted clocks (green and orange lines) result in near-perfect chain density (in the end, 1800 blocks).
- The cluster run with one fast clock on 8.8.0-pre (red line) is only slightly worse than that.
- The cluster run with one fast clock on 8.7.3 results in significantly (~25%) reduced chain density.

Further manual inspections of the log files are also consistent with the analysis above. For example, trace messages indicating chain selection for future blocks occur only on 8.7.3, and the 8.7.3 cluster run with the shifted clock also features significantly more fork switches than the other ones.

We are therefore very confident that the block production outages observed on mainnet will stop as soon as 8.8 (or any later version) has been widely deployed.

Reproducing the local cluster runs

1. Check out the cardano-node repository at either 8.7.3 or 8.8.x (PR #5648 at time of writing).

Apply this patch to ouroboros-consensus. If you use Nix, you can add this snippet to “modules” in nix/haskell.nix:

```
Unset
packages.ouroboros-consensus.patches = [(pkgs.fetchpatch {
  url =
"https://gist.githubusercontent.com/amesgen/33b045472764b1ccc4d271749727221e/raw/ce1bce6ab0abd9da47b27a5a2f5cc5f44535ce15/cardano-clock-shift.diff";
  hash = "sha256-teiTKo5zOAgkwVzo95LUWJmwEJb9c/rg7SYMw8wsDYI=";
})];
```

2. Build cardano-cli and cardano-node and add them to the PATH.
3. Run the following bash commands, potentially commenting out `export CARDANO_CLOCK_SHIFT=0.1`

```
Unset
scripts/babbage/mkfiles.sh
export CARDANO_CLOCK_SHIFT=0
example/node-spo1.sh
example/node-spo2.sh
export CARDANO_CLOCK_SHIFT=0.1
example/node-spo3.sh
```

4. Inspect/plot the log files as desired. For example, the graph above shows the AddedToCurrentChain events over time.