This is a follow up of

Time as a Public Service in Byzantine context,

Time attacks and security models and Sensor Fusion for BFT Clock Sync.

We have discussed already that time setup vulnerabilities pose a threat to overall beacon chain protocol security properties (eth2.0-specs issue #1592, Time attacks and security models).

However, the attacks are perhaps not very realistic, if implemented directly, since the problems will be detected by administrators and necessary counter-measures will be taken. But such attacks can aid more sophisticated and/or stealth attacks.

One direction of such opportunistic behavior can be de-anonymization. As an attacker can adjust clocks of vulnerable nodes, it can affect their behavior which is noticeable within some period - i.e. if such a node is validating, then it should attest blocks each epoch and there will be late or early attestations, depending on how the attacker has shifted the clock. It also affects other validator duties like block proposal and attestation aggregation. However, attestation are to be issued each epoch, so it's easier to start from analyzing attestations. A validator becomes proposer very rare, so it's probably not very useful from de-anonymization perspective.

De-anonymization attack and setup

We assume there is a node or set of nodes, some of them are participants of beacon chain protocol (though extensions to other protocols are possible), i.e. validating nodes, others being non-validating nodes. An attacker cannot control nodes directly, however it can control clocks of some nodes.

Each node contains zero or several validators. Each validator possesses a distinct private/public key pair, which means its behavior can be distinguished from other nodes, by monitoring its activity. E.g. someone (not necessarily an attacker) can send a block to the nodes. Then some nodes will attest it according to the beacon chain protocol. As signing keys are different, nodes responses will differ too. During an epoch, all correctly functioning validator nodes should participate (become attesters). Therefore, monitoring node activity allows to determine which validators are associated with the node.

In reality, it would be quite difficult for an attacker to monitor messages of many nodes. There are also non-validating nodes. We therefore consider the case, when attacker power is limited, i.e. we assume an attacker can exploit vulnerabilities of NTP setup (e.g. one, two, three) of some nodes. And the goal is to use this weakness to recover validator IDs/public keys associated with the node.

For simplicity, we assume that association between validators and nodes doesn't change over time. Though, in practice this can be a counter-measure against such attacks, we concentrate on a simpler static (association between validators and nodes) case.

So, an attacker can probe a node by shifting its clock. It also needs access to outcomes, to be able to infer whether the node is validating or not and which validator IDs/public keys are associated with the node. We assume that an attacker only need to observe beacon chain blocks, so it doesn't need to participate in p2p-graph, generate messages, own validators, monitor transient traffic, etc.

De-anonymization attack sketch and its parameters

Clock shift direction

The basic idea is simple: an attacker can shift a node's clock and observes how properties of block stream changed. We concentrate on attestations (included in beacon blocks), as they are to be produced each epoch by each validator. So, if a node's clock is slow (and the node is validating), then attestations from the node will be included in beacon blocks later, i.e. inclusion_delay

of PendingAttestation will be greater, on average, than if the node clock is normal.

The other possibility is that the node's clock is hastened, then validators on such node will be issuing attestations which refer to stale head blocks. I.e. the slot of the block with root beacon_block_root

of corresponding AttestationData will differ from slot

(of the AttestationData

) more on average, than if the node clock is normal.

Clock shift duration

One parameter of attack, which can vary is duration of a clock shift. Typically, a clock shift should be introduced for a short period of time, e.g. an epoch or two. A typical NTP synchronization period is 1024s, while an epoch duration is 384s, so such attacking clock shifting is higher frequency than typical NTP clock sync and likely attract no attention, unless specific monitoring code is added.

Another possibility is gradual introduction of time shit, e.g. slowing clock down at rate around 100ppm (which we consider as a nominal drift of typical oscillators used in RT clocks).

Clock shift magnitude

Another variable is a magnitude of clock shift. Huge time shifts will result in a more prominent validator behavior, which is easier to distinguish. E.g. if a validator clock is set to be more than one epoch late, then its attestation won't be included in blocks (and even won't be re-send by transient nodes).

At the same time, such problems are much easier to notice. E.g. it's highly likely validator operators will monitor attestation inclusion metrics.

However, such attack strategy can work if the clock shift period is short.

Time shifts of lower magnitudes (several slots) will result in prominent behavior too. I.e. attestation delayed by several slots will be included several slots later (and validator will receive fewer rewards) - that can be determined from block stream.

Statistical testing

Clock discrepancy can occur for reasons other than attacks, e.g. NTP clock synchronization can be missing. It can also result in attestations included with delays. A node can stop working for some reason or a fault can happen. It's especially important, if an attacker introduces clock shifts of low magnitudes.

As an attacker can control clock, then it can (statistically) discriminate whether attestation inclusion metrics are sensitive to the node clock shifts or not. Distribution of inclusion_delay

and its descriptive statistics like mean, median, mode, etc directly affected by clock shift in case of validating nodes.

So, an attacker can sample inclusion_delay

(of all validators) when clock (of some node or some nodes) is shifted and when it's not shifted.

Then a statistical test like Mann-Whitney U test can be used to test if a particular validator behaves different when node clock is shifted.

There are couples of issues to consider.

First, fishing bias should be accounted for - given lots of validators, there is a probability that some irrelevant validator (i.e. associated with a node which clock has not been shifted) will respond to a clock shift just by chance. That can be handled by proper design of experiment/randomization and/or selection of statistic parameters (e.g. adjust significance level appropriately).

Second, there are lots of samples when clocks are normal (assuming most validator clocks work fine when not under attack). So, experiment design should consider sampling "under attack" data mostly. The longer attack, the more chances it will be detected by node operators. So, optimization of design may be quite useful to make such attack stealthy.

We won't consider statistical details anymore, since our goal is to describe such attack and counter-measures on a high level.

Divide-and-conquer de-anonymization for multiple nodes

Assume an attacker can control clocks of k

nodes. A straightforward but slow approach would be to test them one by one, however, it requires $O(kmt)$

time, where m

is amount of clock shift experiments to test individual node and t

is the time required to do one experiment. If there are lots of nodes with vulnerable NTP/clock setups, it can take a lot of time. E.g. if t

is two epochs (one epoch for on and one for off) and one want to make 10 experiments per node, then it requires 128 minutes per node.

A simple divide-and-conquer approach can be used to speed up the process. E.g. an attacker can shift clocks for all nodes it can control first, then select validators that demonstrate statistically significant dependency on the clock shift on/off state.

Then, the set can be split in two parts and the same experiment performed on each part separately (it can be combined actually, e.g. first part is on, second is off and vice versa).

The step can be repeated by splitting the set differently.

Basically, given k nodes, determining which validators belong to which node requires O(log(k))

steps and thus O(log(k)*m*t)

time. For example, for 10K nodes it should require something like 2000-4000 minutes, which is quite reasonable time.

Of course, a real attack can be more complex depending on circumstances (like desired level of stealthiness).

Clock sync as a counter-measure

As a clock synchronization protocol can be implemented, which fuses huge amount of reference and local clocks and thus limits ability of an attacker to affect nodes' time, it's interesting whether such de-anonymization approach is still possible?

Filtering using local clocks

Let's first consider a very simple countermeasure: filter reference clock readings, which deviate too much from local clock reading.

E.g. one can assume that a local clock can drift within 100ppm. So, if an attacker tries to shift the node's clock with a big value, it is easily detected (assuming that correct initial local clock offset is known).

That means an attacker need more time to introduce a significant clock shift (which will be enough to affect attestation inclusion metrics). For example, an attacker want to shift clock by -1 slot (i.e. slow it down by 12 seconds, using current beacon chain protocol settings). Assuming 100ppm shift incurred by attacker cannot be distinguished from natural local clock drift, slowing the clock will take 120000 seconds (e.g. about 1.5 day). The on/off cycle thus will take about 3 days. Note that during the period, there will be lot of under attack

samples. So, assuming 10K nodes, it can be enough to do 14 (log(10000)+1

) such cycles, using the above divide-and-conquer approach, i.e. about 42 day. Though randomization may require to do that several times (to possible counteract spurious correlations or fishing bias).

Anyway, while such a simple counter-measure makes the attack longer, the time remains more or less reasonable.

Calibrating local clock can probably increase necessary duration of the attack even more, rendering it less useful (e.g. validators can migrate to different nodes during the period the attack is conducted).

Clock sync protocol as a counter-measure

When nodes fuses readings from lots of reference and local clocks, it becomes much more difficult to affect time. We consider two cases of attacker power limitations:

- an attacker can control reference clocks of a third minority
- an attacker can directly control a third minority

Where "third minority" means less than one third of validators (validator balances).

NB

As validator distribution accross nodes can be highly non-uniform, 1/3 of nodes can be quite different from 1/3 of validating power. However, we ignore the distinction. Basically, we assume an attacker is not powerful enough to break beacon chain protocol or clock sync protocol directly.

Reference clock case

BFT clock sync protocol (e.g. here) can limit the amount of clock shift which can be introduced by an attacker. Though some shift is still possible. E.g. if an attacker introduces a negative shift to reference clocks of < 1/3 nodes (e.g. nodes which use compromised NTP servers, or an attacker exploits NTP vulnerabilities to send fake NTP packets), then clock sync outcome of all nodes will be affected, since they exchange with clock readings. However, the effect will be limited by other uncertainty factors (correct reference clock accuracy, network delays variations, local clock drift).

Basically, if there is no attack, the uncertainty factors will tend to cancel each other, resulting in a low error (relative to the common time standard, that reference clocks are synchronized to). An attacker is able to introduce a shift in the same direction for all nodes, so that the nodes are not able to distinguish the shift from random factors.

The other aspect is that such attack shifts all clocks (synchronized by the same clock sync protocol) at the same time. That means while an attacker can affect clocks in a limited way, it cannot affect metrics we consider above, as all clocks are slowed down or hastened (relative to a time standard) and the clock shift doesn't grow with time, as the majority of reference clocks stay correct.

In order to implement the discriminative attack described above, an attacker should be able to manipulate message flow (e.g. introduce delays) of selected nodes.

Direct node control case

If an attacker can control < 1/3 of nodes directly, the situation is different. First, the task becomes to associate validator IDs with other nodes, which are not controlled by the attacker.

Second, an attacker can send different clock readings for different (honest) node subgroups. I.e. it can send negatively shifted clocks to some nodes and positively shifted to others. That means a discrimination becomes possible.

However, as a clock sync protocol limits the magnitude of shifts that an attacker can introduce - it's close to sum uncertainties of network delays, reference clock accuracy and uncertainty due to local clock drift . So, statistical properties of inclusion_delays

for attesters under attack

and under normal conditions will be much closer and difficult to discriminate by statistical methods.

Though, it will be quite possible, given enough number of experiments. Again, under normal conditions (no attack), uncertainty factors tend to cancel each other, so clock discrepancy will be quite low (e.g. tens of milliseconds), while under attack, one can expect hundreds of milliseconds of discrepancies (e.g. see simulation experiment here).

So, under attack clock disparity between nodes is about several percent of slot duration. This means inclusion_delay

for validators under attack will be greater on average than the inclusion_delay

for the same validators under normal conditions. To answer the question how it affect the de-anonymization attack duration requires a separate investigation.

However, if an attacker can control significant amount of nodes directly, then its opportunities to de-anonymize are much broader, e.g. a significant portions of nodes in p2p-graph is controlled by the attacker, which means it has many opportunities to monitor transient messages and/or manipulate message flow (e.g. delay messages and/or send messages earlier - those that are to be sent by validators controlled by the attacker).

Therefore, the assumption that an attacker can directly control significant portion of nodes leads to a de-anonymization problem which differs significantly from the problem that we discussed here, in the write up. And thus deserves a separate train of investigation. See for example, Packetology: Validator Privacy.