Local reputation

By thomas.huet@acinq.fr

Overview

We introduce the following scores in [0, 1]:

- Reputation for our local peers. We compute it based on past behavior.
- Confidence in HTLC: When a peer relays a HTLC, they signal how confident they are that it will succeed. This needs to be added to the protocol.
- Occupancy of our channels.

When receiving an HTLC, we compute our confidence that it will succeed as *our_confidence* = *incomming_reputation* × *incoming_confidence*
We relay the HTLC if and only if *our_confidence* > *outgoing_channel_occupancy*. If we do relay, we send our own confidence to the next node.

Slots and liquidity

Jamming attacks can target

- HTLC slots. There is a limited number of slots per channel and an attacker can try to fill all of them (usually with very small amounts).
- Liquidity. An attacker can send large payments to lock all of the liquidity that we have for this channel.

To deal with these two dimensions, we can duplicate the scores and relay the HTLC only if the slot score is higher than the slot occupancy **and** the liquidity score is higher than the liquidity occupancy.

Computing reputation

We want our confidence in HTLCs to be well calibrated. Which means that the reputation of a node should depend on how well their confidence is calibrated, but not necessarily on the quality of the HTLCs they send us. If a node sends bad HTLCs but marks them as low-confidence, it shouldn't decrease their reputation.

Bad HTLCs

A "good" HTLC is one that succeeds fast. Anything else is a "bad" HTLC:

- HTLCs that fail. It is expected that there will always be failing HTLCs, even in a healthy network, however they don't pay us fees so they are bad.
- HTLCs that take too long to resolve, even if they succeed eventually. The maximum delay before considering an HTLC "bad" should be randomized so that it's not possible to game the system by being just below the maximum delay.

Some honest HTLCs will be considered "bad" but it's OK. The confidence that we compute is just used to prioritize HTLCs, if we are not under attack, even very low confidence HTLCs will be relayed.

Instead of classifying HTLCs as either good or bad, we could also give them a continuous score (*is_success* - K × *time_to_resolve*), however I'm not sure the added complexity is worth it.

The basic formula

For a given peer, we consider all incoming HTLCs in a given window in the past. The window should be parameterized by a time duration and a minimum number of HTLCs so that it's never empty (except when the channel was just created but even then we could add a dummy "good" HTLC).
For every HTLC, we define

- *w*, the weight of this HTLC. If we're considering slot jamming, all HTLCs have a weight of 1, if we're considering liquidity jamming, the weight corresponds to the amount being sent or the fees (if we have channels with different fees, using the fees instead of just the amount prevents an attacker from "buying" reputation on the low-fees channels and "spending" it on the high-fees channels).
- *c*, the confidence given by the previous node.
- *s*, the success value: 0 for "bad" HTLCs and 1 for "good" HTLCs.

The reputation of the peer is computed as
reputation = sum(*w* × *s* / *c*) / sum(*w*)

This formula is derived directly from our target *confidence* × *reputation* = *success_probability*: If our peer always uses the same confidence *c*, we get *reputation* = (sum(*w* × *s*) / sum(*w*)) / *c* which is exactly what we wanted as (sum(*w* × *s*) / sum(*w*)) is our best estimate of the success probability.

More robust computation

However this formula is too naive, it can lead to a reputation higher than 1. Even more problematic, it is too easy to gain reputation by sending HTLCs you know will succeed but marking them as extremely low confidence.
~~We only want the score of a "good" HTLC to offset the scores of "bad" HTLCs that have a lower or equal confidence. To fix that, we need to iterate over HTLCs by order of confidence (and "bad" HTLCs first if they have the same confidence) and never allow the partial sum(*w* × *s* / *c*) to be higher than the partial sum(*w*).~~ This is still not enough, what we really want is the reputation to be a function of the confidence. That way a node can easily build a high reputation for the low confidence HTLCs but it would be a lot more costly for it to be trusted for the high confidence HTLCs. When we receive a new HTLC, we should compute the confidence of the previous peer considering only the past HTLCs that have the same confidence. The problem with this approach is that we may lack data, so we may need to also consider HTLCs with different confidence scores. I suggest that if we have enough HTLCs with the same confidence to compute a meaningful score we do so but if the sample size is too small, we also take into account the HTLCs with higher confidence.

Another detail is that we should count all currently unresolved HTLCs (including the one that triggered this reputation computation) as "bad". That way, if an attacker starts an attack with a lot of HTLCs, only the first one will benefit from the still high reputation and the next ones will be more and more penalized.

Discrete confidence

Some people expressed concern that using a continuous confidence score would leak the distance from the sender since the confidence would decrease by a (potentially predictable) amount at each hop. To mitigate this, we can use a small number of classes instead of a continuous score, for instance we can use a binary score: high / low confidence. This is easily done by rounding the continuous confidence score to the desired precision.

Spec changes

The only spec change required is a way for a node to communicate its confidence in an HTLC to the next node. The formula to compute the reputation is just my recommendation but any node can choose to do whatever they want without impacting the rest of the network.

Choices

- Do we use two confidence scores (one for slot jamming and one for liquidity jamming) or a single one (that would be the minimum of the two)?
- Do we use continuous scores, binary scores or something in between?

Any additional bit of information is potentially a privacy leak but could enable a better decision downstream.

Deployment

The change can be backward-compatible by using an odd TLV: legacy nodes will just ignore the confidence score and upgraded nodes can use a default confidence of 1 (or anything else, each node can choose for itself) when they receive HTLCs from legacy nodes.
I believe that this reputation scheme would provide some jamming protection even if only a small portion of the network implements it. The benefits would improve as more nodes implement this scheme.

Discussion

Negative feedback loop

If the reputation of a node is low, we will mark their HTLCs as low-confidence which increases their chances of being rejected and in turn will decrease the reputation of the node.
I believe that this would only happen during times of congestion, most of the time, even 0-confidence HTLCs will be propagated which would break this feedback loop.

Reputation for next node

I believe it makes more sense to assign a reputation to the previous node in the route than to the next one because if every node in the route does the same, it will eventually penalize the

sender. And the attacker is usually the sender. However in some cases it may be useful to also assign a reputation to the next node, or even to pairs of previous / next nodes.