TL;DR

In this post, I dig into the bouncing attack on Casper FFG, which is already known to potentially make a permanent liveness failure of FFG. I present specific cases where this attack can happen. Also, I describe how the choice of the fork-choice rule relates to this attack.

Prerequisites:

- Casper FFG paper

Background: Bouncing attack

In Casper FFG, the fork-choice rule must start from the latest justified checkpoint.

Alistair Stewart found that this introduces an attack vector where the attacker leverages the inconsistency between the latest justified checkpoint and the fork-choice due to accidental or adversarial network failure.

[

image

961×530 43.4 KB

](https://ethresear.ch/uploads/default/original/2X/0/08bd315f46dba836786e05b46898c27aa4586c8e.png)

Because of this, FFG cannot have liveness in eventually/partially synchronous model regardless of the choice of the fork-choice.

Then, looking deeper than the traditional network model in distributed systems, in what situation this attack happens in practice?

Setup

The protocol we discuss is as follows:

- Casper FFG, slot, epoch, and attestation-committee a la ETH2.0 are adopted
- The number of the total validators is n

, assume static validator set & homogeneous weight (stake) for simplicity

- The attacker controls t < n/3

validators

- The attacker is mildly rushing: the attacker can get votes from and deliver votes to honest validators with up to a small delay (e.g. a few slots, proportional to t

)

Notations:

- \mathrm{Epoch}(C)

: the epoch of a checkpoint C

- \mathrm{FFGVotes}(C)

: the number of Casper FFG votes (attestations) whose target is a checkpoint C

- t_C

: the number of votes by which the attacker can vote for C

later (t_C \le t

) * I.e. The number of the votes which the attacker saved up in \mathrm{Epoch}(C)

- I.e. The number of the votes which the attacker saved up in \mathrm{Epoch}(C)

Bouncing condition

First, we discuss bouncing condition i.e. a condition in which an attacker can do a bouncing attack even in fully synchronous network.

Bouncing condition: there exists a latest justified checkpoint C

and justifiable

checkpoint C'

such that (i) C'

is later

than and (ii) conflicting with C

.

- Justifiable

: 2n/3 - t_C \le \mathrm{FFGVotes}(C') < 2n/3

i.e. C'

is not justified yet but the attacker can justify C'

at an arbitrary time by himself

- Later

: \mathrm{Epoch}(C') > \mathrm{Epoch}(C)

This condition is sufficient to start bouncing attack; because if it is satisfied,

- Honest validators votes for a checkpoint C''

, which is a descendant of C

- C''

becomes justifiable

- The attacker justifies C'

before C''

is justified (This is where the rushing requirement for the attacker comes in) * In practice, the attacker publishes a block which contains the votes for C'

- In practice, the attacker publishes a block which contains the votes for C'

- C'

and C''

makes the bouncing condition satisfied again

[

image

746×507 15 KB

](https://ethresear.ch/uploads/default/original/2X/9/9530525f4da2a10b490be284b0e4acb9b1eacfd9.png)

How the bouncing condition is satisfied?

Since C

is justified, at least the majority of the honest validators voted for C

in \mathrm{Epoch}(C)

.

Also, since C'

is justifiable, at least the majority of the honest validators voted for C'

in \mathrm{Epoch}(C')

. (The proof is omitted for simplicity.)

Therefore, for the bouncing condition to be satisfied, the fork-choice must have switched from the chain of C

to the chain of C'

.

Below, we describe scenarios where the fork-choice switches and the condition is satisfied by ~2 epochs of network failure.

Case 1: Switch by saving (only in LMD)

In LMD GHOST, the attacker's votes from an epoch earlier than \mathrm{Epoch}(C)

can make the switch.

One example is a case where the attacker saved votes up and then published it.

[

image

869×477 36.4 KB

](https://ethresear.ch/uploads/default/original/2X/b/bcd31a7a4a61977bd58176a689eaf798d9d9bf71.png)

The exact condition that this happens are:

- There is a justifiable (but not justified) checkpoint C

, and there is no later justified checkpoint

- There is a checkpoint C'

later than and conflicting with C

- C'

is forked off in an epoch earlier than \mathrm{Epoch}(C)

- The attacker saved up t'

votes in an epoch earlier than \mathrm{Epoch}(C)

- In \mathrm{Epoch}(C)

, the votes are sufficiently split such that 2n/3 - t' \le \mathrm{FFGVotes}(C) \lt n/2

- For such a case to be possible, t' > n/6

- For such a case to be possible, t' > n/6

In simple terms, if the justification is delayed until the attacker succeeds to save up some votes and then in a later epoch the votes are sufficiently split, the attacker can start the bouncing attack.

Compared to the decoy-flip-flopping where the attacker also needs to succeed in saving, this is much stronger and easier to do successfully.

In FMD GHOST, which modified LMD GHOST so that only votes from the previous/current epoch are counted, this strategy does not work because the saved t'

votes cannot affect the fork-choice.

Case 2: Switch by biased message delay

Here, we consider a case where the network delay is biased (by accident or the attacker) for votes for C

and the fork-choice switches.

[

image

852×429 36.2 KB

](https://ethresear.ch/uploads/default/original/2X/8/899807c683a5dac3147d8af2a67bd0087dcd0867.png)

The exact condition that this happens are:

- There is a justifiable or justified checkpoint C

, and there is no later justified checkpoint

- There is a checkpoint C'

later than and conflicting with C

- Some of the votes for C

is delayed so that for honest voters, (i) C

is seen not to be justified yet and (ii) C'

wins the fork-choice * The necessary length of the delay depends on the slot allocation of the next epoch; the earlier slot the validators voted for C

and the attacker are allocated to, the quicker the fork-choice switches

- The necessary length of the delay depends on the slot allocation of the next epoch; the earlier slot the validators voted for C

and the attacker are allocated to, the quicker the fork-choice switches

In the above example, where there is no fork earlier than \mathrm{Epoch}(C)

, there is no difference between LMD and FMD.

However, when the fork is from an earlier slot and most of the validators voted for the same chain as the previous epoch, there is a case where the attack succeeds in FMD but not in LMD.

Implications

- A bouncing attack happens with ~2 epochs of network failure
- A bouncing attack works for every fork-choice in FFG, but fork-choice rules have their strengths and weaknesses
- We cannot conclude LMD vs FMD regarding bouncing attack since we have little knowledge about how these cases happen
- We cannot conclude LMD vs FMD regarding bouncing attack since we have little knowledge about how these cases happen