TL;DR

We propose an attacking strategy to censor specific transactions (e.g., fraud proofs in Layer2 protocols), in which it is difficult to identify the attacker.

Even if users can recover from this attack by a socially coordinated soft/hard fork, we cannot penalize the attacker without penalizing honest validators.

Background

The safety of fraud-proof-based Layer2 protocols (e.g., Plasma, Optimistic Rollup, state channels) depends on the assumption that if operators/players claim invalid off-chain state, a fraud proof can be submitted to and gets included in the chain within a pre-defined period (Footnote 1). Recently, @gluk64 brought up the security of these protocols for discussion on Twitter and ethresear.ch. Roughly speaking, an attacker can make a profit by cheating in such Layer2 constructions by 51% attack to censor fraud proofs, making the whole system (including Layer1) not incentive compatible, especially because the in-protocol reward compensates for the attacker's cost to run validators (Footnote 2).

The major countermeasures are:

- We can increase the cost of such censorship attacks by the"counter DoS attack." (cf. @adlerjohn's comment)
- TL;DR: In Ethereum's smart contract model, you don't know which contracts are called in executing a transaction without actually executing it, so DoS attack against the censorship is possible by sending a bunch of transactions to the attacker
- TL;DR: In Ethereum's smart contract model, you don't know which contracts are called in executing a transaction without actually executing it, so DoS attack against the censorship is possible by sending a bunch of transactions to the attacker
- We can recover from the attack via soft fork and also "wipes out" the attacker's stake in PoS. (cf@vbuterin's comment)

In this post, we describe a strategy of censorship attack for which these mechanism does not apply. This strategy works for any transactions other than fraud proofs in Layer2 protocols.

Description of the attack

We assume a PoS system like the eth2 beacon chain (Footnote 3), where a block proposer creates a block, a committee of validators vote for a block every slot (e.g., 6 seconds), and those votes are also interpreted as Casper FFG votes to finalize blocks. We expect similar strategies work for other consensus protocols or PoW systems.

Attacker's capability

In this post, we assume an attacker who controls more than the majority of stakes with a margin (e.g., 55% stakes). Because such an attacker controls more than 1/3 of the votes, blocks which is not favored by the attacker cannot be finalized by Casper FFG.

Attacker's strategy

1. If the main chain does not include a fraud proof, the attacker vote for the main chain.
2. If the main chain includes a fraud proof, the attacker creates and vote for a conflicting chain to fork off the fraud proof.

2.1 However, a small number of attacker's validators vote for the chain which includes the fraud proof. * These are "alibi votes" to make the attacker's validators behave similarly to honest validators

- The number of alibi votes must be sufficiently small (e.g., 5% of the votes of the slot) so that the block cannot win in the fork-choice.
- These are "alibi votes" to make the attacker's validators behave similarly to honest validators
- The number of alibi votes must be sufficiently small (e.g., 5% of the votes of the slot) so that the block cannot win in the fork-choice.

The attacker chooses either one of the below with a certain probability when creating a block.

1. Create a block without a fraud proof, filled with self-made transactions (e.g., A transaction that transfers tokens between accounts which the attacker controls) (Footnote 4).

2. Create a block which contains a fraud proof, along with self-made transactions.

4.1 The attacker publishes this block slightly after the next block is published so that most honest validators do not vote for the block

4.2 The attacker publishes alibi votes for this block (e.g., with 40% of the votes of the slot)

Also, the attacker has a strategy to encourage honest validators to create blocks without fraud proofs.

1. The attacker frequently broadcasts self-made transactions with high fees

“Security arguments” for attackers

Resistance to the “counter DoS attack”

When the attacker creates a block, he does not include any transactions created by other people. Therefore, the attacker verifies only the transactions included in honest validators’ blocks, which is already required to be a full node, so the attacker does not suffer from the above DoS attack.

Difficulty of detecting the attacker

In the above strategy, the attacker does not make equivocations or invalid messages. The differences between attackers and honest validators are only about how they create, publish, and vote for blocks that contain fraud proofs. To decrease the difference in the ratio of creating blocks with fraud proofs, the attacker increases the use of strategies 3 and 5 more. To decrease the difference of the ratio of voting for blocks with fraud proofs, the attacker increase alibi votes (strategies 2.1 and 4.2) and decrease honest validators’ votes for fraud proofs made by attackers (strategy 4.1).

Other than these, the strategies 2 and 4.1 leverages the nature of distributed systems that liveness faults cannot fundamentally be distinguished from a network failure. Further, there would be various network-level attacks by which the attacker can delay honest validators’ blocks and votes to make the honest validators look similar to the attacker.

Therefore, although we must refine the strategies (e.g., by parameterizing the attacker’s stake, the network conditions) to make these arguments more formal, we conjecture that it can become difficult to identify the attacker’s validators without making a false accusation.

Punishing the attacker by a soft/hard fork

If users coordinate a soft fork to recover from a censorship attack, the attacker can join the new chain soon to avoid being detected and any punishment on liveness failure (e.g., inactivity leak). Because it is difficult to identify who is the attacker, what we can do to slash the attacker’s stake is to punish all the suspicious validators with the risk of punishing honest validators, with a philosophy similar to the “penalizing both sides.”

Collateralization

One approach to make sure that only the attacker’s side gets penalized is to force anyone to deposit to be an operator of Plasma/Optimistic Rollup or participate in channels and slash the collateral by a fraud proof. If we assume socially coordinated soft fork is likely to succeed and the collateral is sufficiently high, such Layer2 protocols can potentially achieve incentive compatibility, with a trade-off of higher barriers to entry. If we want to avoid punishing too much on frauds due to honest mistakes, where it is expected that the fraud proofs smoothly submitted without being censored, we can limit the default amount of punishment, and punish a lot by a hard fork (irregular state change) in the case of censorship attack.

Footnotes

1. OTOH, fraud proofs of invalid state roots in sharding invalidate the chain without being included on-chain.This post proposes a clever technique about this.

2. Note that the attacker can benefit not only from a single Layer2 construction but also from multiple Layer2 constructions at the same time. This is why to cap the value of the off-chain asset does not solve the problem.

3. In eth2, Layer2 systems will be deployed on shard chains, not beacon chain, so the optimal attacking strategy might become different. We leave this for future discussion.

4. The fees of such self-made transactions go back to the attacker’s hand.