TLDR: Any Ethereum node could lie to an optimistic rollup coordinator about the existence of state so long as the fictional state is valid. A coordinator would only discover this when attempting to send a transaction or switching Ethereum providers.

Optimistic rollups operate by storing block information using calldata on the Ethereum network. This allows any individual to operate an Ethereum node and download the state of the rollup. In practice some individuals will not operate their own Ethereum node and instead rely on hosted solutions like Infura.

Validity vs Existence

It's possible to construct a valid

rollup chain quickly and cheaply. All that is necessary is taking transactions, constructing a state and storing it somewhere other than the Ethereum network. When a rollup coordinator receives state history from an Ethereum node the coordinator can determine if the data is valid

by replaying the state transitions; however the coordinator cannot determine whether the state exists

on chain. The best it can do is ask the Ethereum node if the state exists.

A Contrived Attack

Imagine an Ethereum node service named Untrust. Untrust provides low cost infrastructure for Ethereum dapps. Untrust could, if they chose to, look at a specific rollup and create a fictional history. They could even create a semi-fictional history by pulling the calldata from some real transactions and mixing them with fictional ones. By doing so they would create a completely different current state hash, but this would only be discovered if the rollup operator using Untrust tried to submit a block (because it would be fraudulent).

This type of attack isn't particularly powerful; the attacker couldn't forge signatures, couldn't directly steal funds, but could lie about their own activity in the rollup.

Imagine Untrust becomes more sophisticated; they decide to attack a rollup called MoneyMover. When a request for the MoneyMover address comes in they instead return information from a node running a completely separate Ethereum network. In this separate Ethereum network Untrust mirrors most of the transactions (to keep balances looking similar) but injects their own transactions when they want.

Now say that Untrust finds a website that uses the MoneyMover rollup for payment. This website runs their own MoneyMover coordinator that connects to Untrust to access the Ethereum network. Untrust could create fictional transactions in their mirror network that would be interpreted as valid by the MoneyMover coordinator (so long as the state transitions are valid). Because the MoneyMover coordinator is not connected to any peers its only way of determining what exists on chain is by asking the Ethereum node (in this case operated by Untrust).

Untrust forges a single fictional transaction on its mirror network and the MoneyMover coordinator operated by the website detects it as valid payment. Untrust now has access to the website without having paid. The website will only discover this fictional payment when they go to submit a transaction, or switch to another Ethereum provider.

Actual Risk

This attack is only relevant when the coordinator operator is not running their own Ethereum node. There are a few cases that are problematic:

1. A downstream consumer of coordinator data has no way of knowing if the data they are receiving exists; they cannot trust the eth node the coordinator is using because the consumer is not running the node themself.

2. Fraud provers are particularly prone to this attack because they only submit transactions when fraud is detected. A malicious eth node could strip invalid transactions from the blockchain data and simply return a different state hash; the fraud prover would have no way of knowing the state hash is fictional and would never submit fraud claims.

3. An eth node operator lies to a rollup coordinator to trick them into either submitting an invalid state transition or submitting an invalid fraud claim. Once this happens the malicious eth node operator could act as the valid counterparty and collect the staked funds.

Solving

Solution 1

In the context of a proof of work chain the block data could be requested by a coordinator and checked for sufficient difficulty. Checking for at least

half the current difficulty should make most attacks financially inefficient.

In a proof of stake chain this becomes less viable as a malicious eth node could provide signatures from addresses that are not staked. They could do this by also lying about the staked amounts when queried.

Solution 2

A less elegant solution would involve reputable sources continuously signing and publishing a list of rollup state hashes known to exist on chain. This list could be published on a service like IPFS. Coordinator nodes could include a preset list of public keys that are known to be reputable (the rollup creators, EF, etc).

A coordinator would know that the rollup data is truthful if

1. All state transitions are valid
2. All state hashes are signed by a reputable source

This would ensure that the rollup data is valid

and non-fictional

and would allow a rollup node to use any Ethereum node to synchronize data.