2021.10.25 - Community update

- New Developer and Community Hub https://manifoldfinance.github.io/hub/ Starting point and organizational checkpoint for providing updates and engaging with each other
- Reduce the effort to get updates and find historical updates
- Reduce the 'mission creep' that was happening (multiple different repos for documentation, hard to keep up if casual end-user)
- Reduce the effort of formulating and providing updates

Mitigating Security Issues

In the course of working with the sushi team, we caught a 'supply chain attack' npm package (this was actually apart of an unclaimed bug bounty) A repository is produced here: https://github.com/sambacha/openzeppelin-splinter

Through a mix of versioning package managers (yarn vs npm) a downstream dependency of the Sushiswap interface (also the Uniswap interface) could become compromised. The TLDR is:

the package name includes the versioning. it correctly maps to the proper openzeppelin-solidity however the package name was not taken until months afterwards

"openzeppelin-solidity-2.3.0": "npm:openzeppelin-solidity@2.3.0"

OpenMEV

- currently onboarding 3-4 new searchers to participate in the sushi process with us
- released flashbots fork of mev-geth, as flashbots has decided to not support eth_callBundle

RPC Method docker container & go-ethereum MEV

How does OpenMEV's infrastructure operate in suboptimal conditions?

This will be expanded in an article, and if you have additional questions please ask in discord, the forums or telegram. The quick summary is like this:

For state and error recovery, we use a weight system similar to this:GitHub - sambacha/web3-rpc-failover: web3 rpc provider connectivity failover

For predictive analytics (i.e. knowing ahead of time that we may have issues soon) you can see this repo here that explores Littles Law and backpressure/data plane routing: GitHub - sambacha/capacity: message queue capacity testing. This is especially relevant towards end-users as we measure how long it takes for a transaction to be mined vs. the parameters of the transaction when it was first sent. Here is some sample results for the simulation capacity repo to help you better understand what we are discussing:

client sends 5 requests per second and receives between 2.4 rps and 2.7 rps (HTTP 200) proxy oscillates between 4 and 6 in-flight requests origin processes between 2.7 and 3 requests per second origin has served requests with average latency 2.2 seconds origin has served 50% of requests (50th percentile) within 2 seconds origin has served 99% of requests (99th percentile) within 3 seconds with periodic spikes

Contracts and Staking

Contracts for deploying strategies and trades are in their preliminary acceptance phase, a.k.a Release Candidate. Contracts are located in the Manifold SDK monorepo here. The First deployed strategy will be the USD strategy for Fiat / Crypto On/Off ramp usage.

Staking and DAO beta testing will be in 3 weeks.

Developer and Searcher Services

We are working on providing a custom plugin module system that will allow end users to load or provide a custom go module that enables new and unique RPC methods for their own private usage. We have had a few people ask about this idea i general and plan on supporting it. In addition, users can purchase per RPC method requests to our service in a pay-for-what-you-use model. Pricing will be denominated in GxGas, the standard fixed unit of account of gwei. Gwei costs are paid via $FOLD.