Protec and Attac: Programmatic Execution Layer Consensus Tests

ethereum/execution-spec-tests

danceratopz

Protocol Tester, Ethereum Foundation

Section 1

Why do we need a testing framework?

Client Diversity and Testing

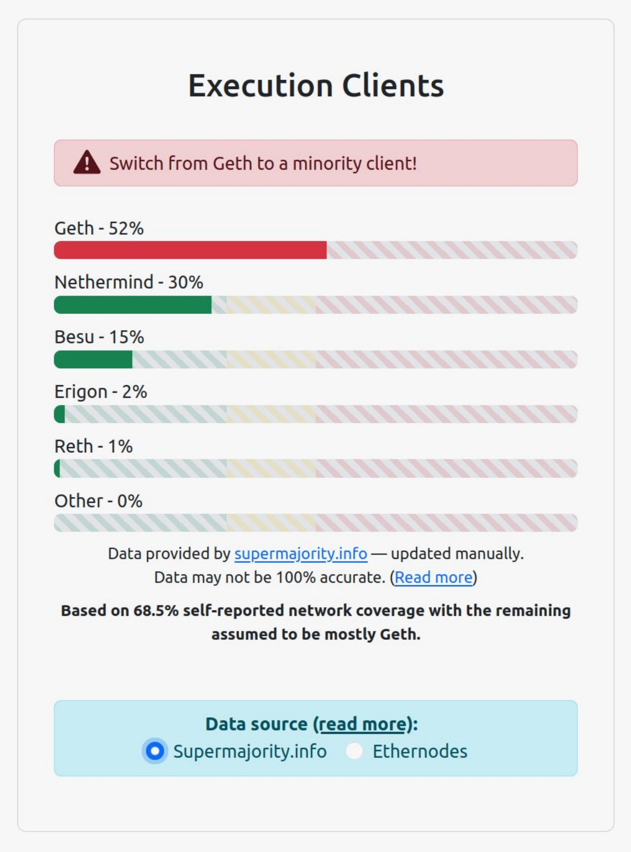
Client diversity is a foundation of Ethereum protocol development.

Client diversity makes the network more resilient against attacks and software bugs.

Ethereum has ≥9 EVM implementations - in ≥8 different languages!

..but it does add complexity to testing.

The ethereum/execution-specs-tests (EEST) team maintains a set of **client-independent Execution Layer (EL) consensus tests**.



Execution Clients

⚠ Switch from Geth to a minority client!

Geth - 52%

Nethermind - 30%

Besu - 15%

Erigon - 2%

Reth - 1%

Other - 0%

Data provided by supermajority.info — updated manually.
Data may not be 100% accurate. (Read more)

Based on 68.5% self-reported network coverage with the remaining assumed to be mostly Geth.

Data source (read more):
● Supermajority.info ○ Ethernodes

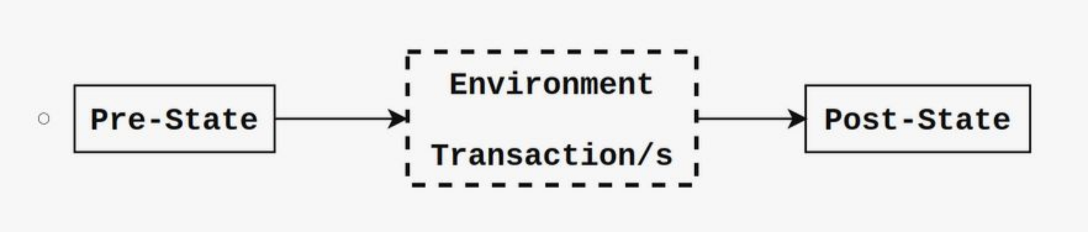
Source: clientdiversity.org, 2024-11-07

Execution Layer Consensus Tests

Every client must handle state changes identically.

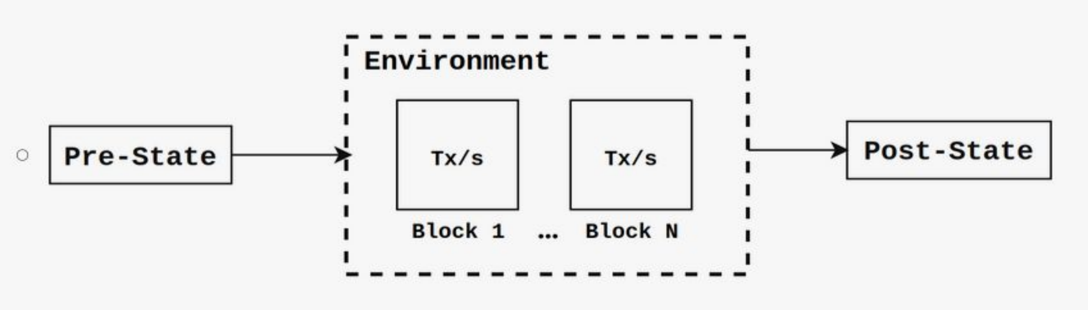
State tests validate

- Opcodes,
- Gas usage,
- Smart contract interactions,
- Transaction rejections.



Blockchain tests validate

- Block progression,
- Block rejections,
- Fork transitions.



Section 2

Status Quo at The Merge

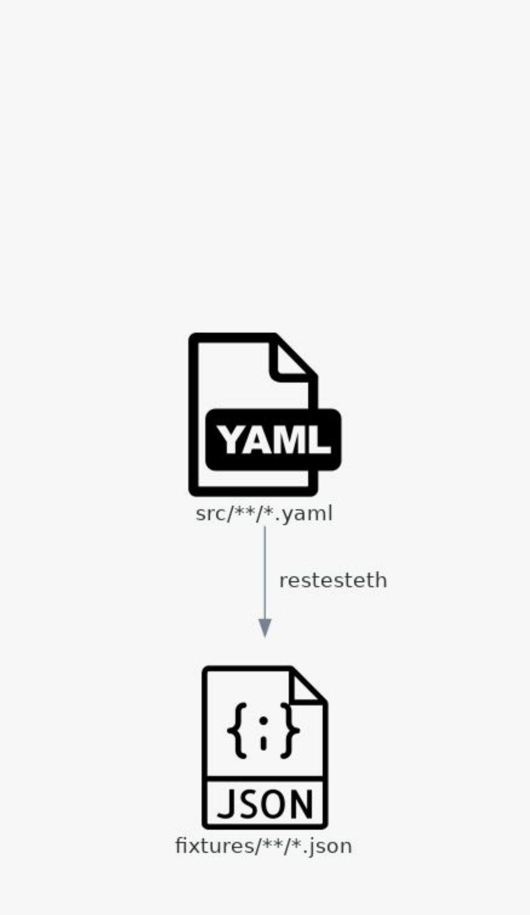
Generating Tests for forks ≥Frontier from ethereum/tests

ethereum/tests contains 4697 tests for Cancun.

Test cases are defined in YAML files, hand-written.

ethereum/retesteth generates JSON test fixtures from the YAML files using a reference EVM implementation.

The process of generating the JSON is known as **“filling the tests”**.



Generating Tests for forks ≥Frontier from ethereum/tests

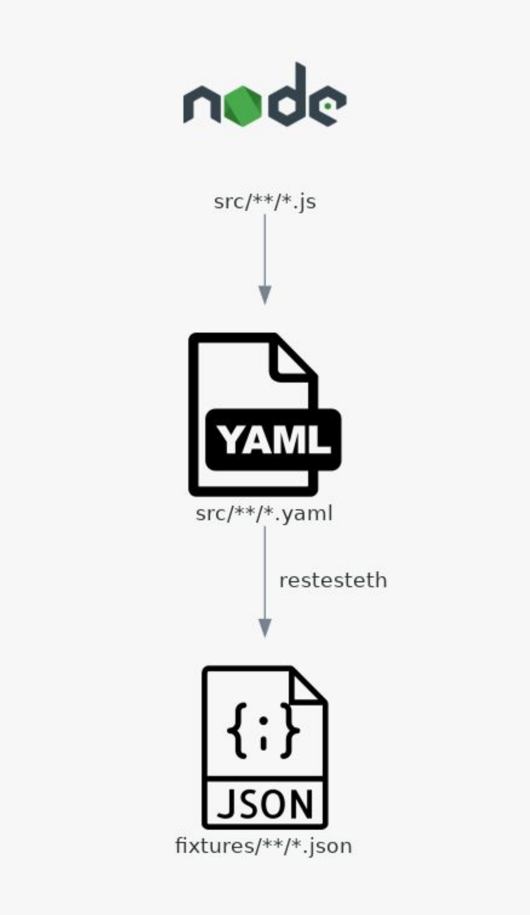
ethereum/tests contains 4697 tests for Cancun.

Test cases are defined in YAML files, hand-written.

🚀 **2021: Partial programmatic generation!**

ethereum/retesteth generates JSON test fixtures from the YAML files using a reference EVM implementation.

The process of generating the JSON is known as “**filling the tests**”.



Section 3

A Framework for Programmatic Tests

The new framework: ethereum/execution-spec-tests

2021-10: interop greece, how to test AA?
⟶ **programmatic test generation!**

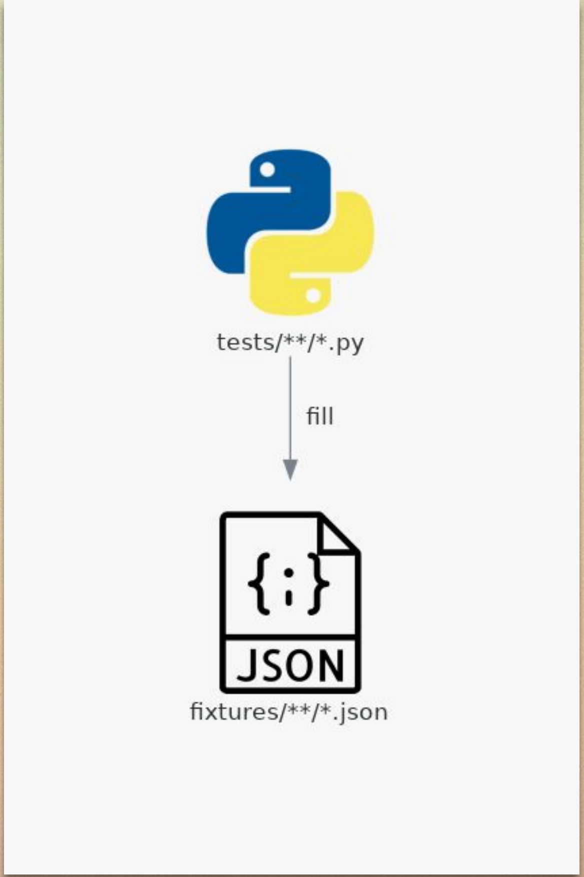
2021-10: cd testing-tools && git init

2021-12: sleep 💤💤💤

2022-10: devcon 6: “testing-tools”!?

2022-10: ⟶ ethereum/execution-spec-tests

2023-03: EEST Shanghai release: 194 tests



Pytest and Cancun

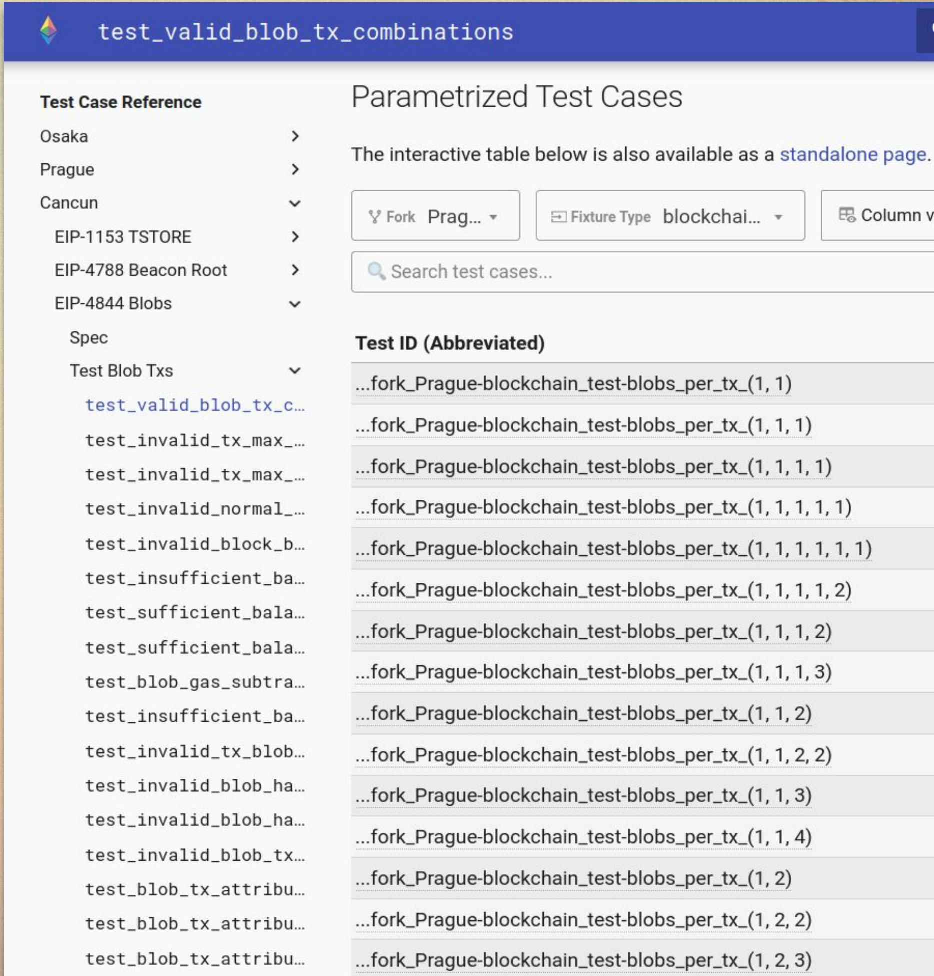
Shanghai showed how easy it can be to implement test cases in Python.

We then re-wrote the framework to use, pytest:

- Extendable via plugins.
  - ...large existing ecosystem.
  - ...allowing a custom modular architecture.
- Easier debugging
- Test cases easily parameterizable

This has greatly accelerated test development.

`2024-03: EEST Cancun release 2412`



test_valid_blob_tx_combinations

Test Case Reference

- Osaka
- Prague
- Cancun
  - EIP-1153 TSTORE
  - EIP-4788 Beacon Root
  - EIP-4844 Blobs
    - Spec
    - Test Blob Txs
      - test_valid_blob_tx_c...
      - test_invalid_tx_max_...
      - test_invalid_tx_max_...
      - test_invalid_normal_...
      - test_invalid_block_b...
      - test_insufficient_ba...
      - test_sufficient_bala...
      - test_sufficient_bala...
      - test_blob_gas_subtra...
      - test_insufficient_ba...
      - test_invalid_tx_blob...
      - test_invalid_blob_ha...
      - test_invalid_blob_ha...
      - test_invalid_blob_tx...
      - test_blob_tx_attribu...
      - test_blob_tx_attribu...
      - test_blob_tx_attribu...

Parametrized Test Cases

The interactive table below is also available as a standalone page.

Test ID (Abbreviated)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1, 1)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1, 1, 1)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1, 1, 1, 1)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1, 1, 2)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1, 2)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 1, 3)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 2)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 2, 2)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 3)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 1, 4)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 2)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 2, 2)
...fork_Prague-blockchain_test-blobs_per_tx_(1, 2, 3)

Contributor First Repository

EIP authors and implementers can use EEST to:

- Write tests,
- Generate tests (from a devnet tx hash),
- Check coverage.

Prague:

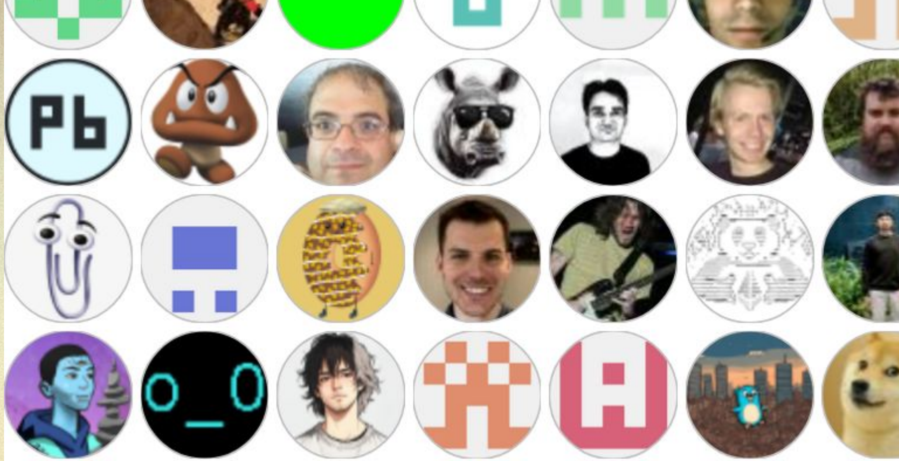
- ~2500 new test cases

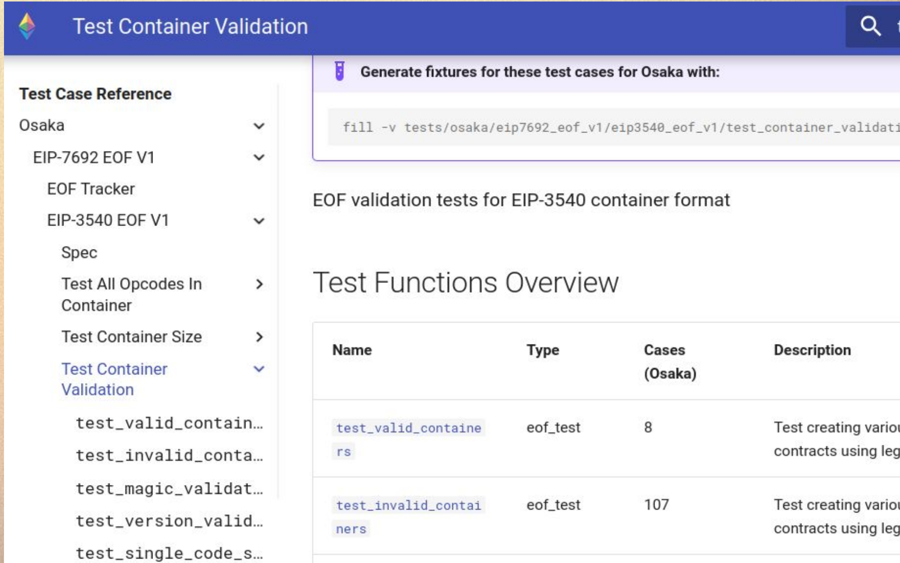
EVM Object Format (EOF):

- Added `eof_test` format.
- ~9000 new test cases for EOF.

Verkle:

- Existing tests (until Shanghai) can be filled for the Verkle Transition.
- Added the witness to the test fixtures.





Test Container Validation

Test Case Reference

Osaka

EIP-7692 EOF V1

EOF Tracker

EIP-3540 EOF V1

Spec

Test All Opcodes In Container

Test Container Size

Test Container Validation

test_valid_contain...

test_invalid_conta...

test_magic_validat...

test_version_valid...

test_single_code_s...

Generate fixtures for these test cases for Osaka with:

```
fill -v tests/osaka/eip7692_eof_v1/eip3540_eof_v1/test_container_validati
```

EOF validation tests for EIP-3540 container format

Test Functions Overview

Name	Type	Cases (Osaka)	Description
`test_valid_containers`	eof_test	8	Test creating variou contracts using leg
`test_invalid_containers`	eof_test	107	Test creating variou contracts using leg

Section 4

Running the Tests

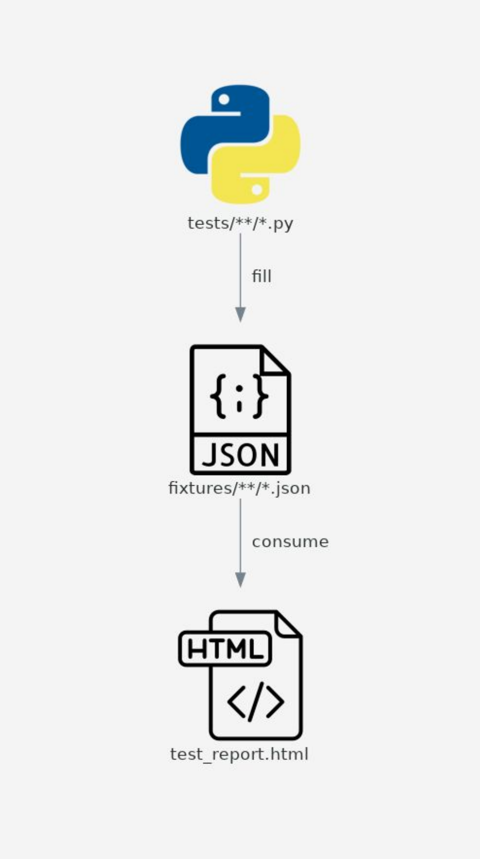
Closing the Loop: `consume (protec)`

Running a test against a client is a two-step process:

1. Generate/fill the tests: JSON.
2. Run the tests against a client

We added three ways to run the tests within EEST:

1. Consume direct (client's native consumer).
2. Consume rlp.
3. Consume engine.



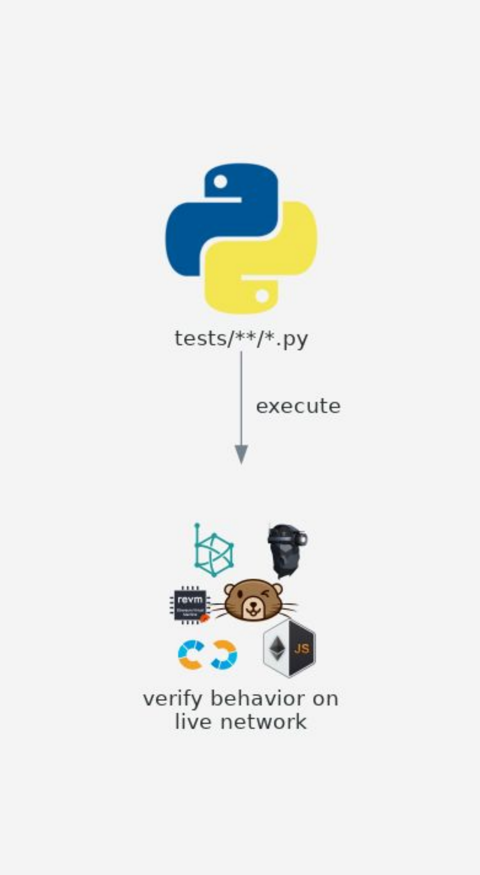
There is no Loop: `execute (attac)`

The next logical step was to extend the framework to allow execution of tests on live networks.

The execute command executes a test case by sending transactions to a clients RPC endpoint.

There is no filling and no JSON is generated.

It unlocks a completely new use case for the execution-spec-tests framework by allowing **execution of the tests in a production(-like) setting!**



The job's not finished

Improve the lives of client developers:

- Merge ethereum/tests into EEST.
- Provide tools to directly integrate into client team test flows.
- Faster test execution.
- Easier test debugging.

Improve the development cycle:

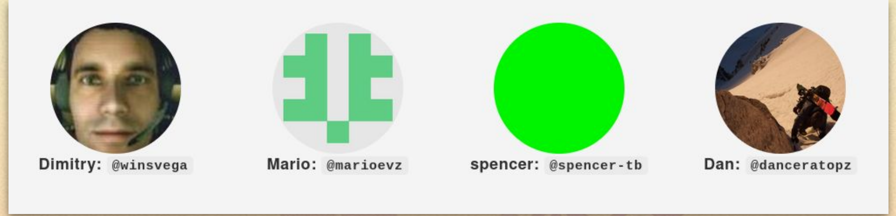
- Add EIP implementations to the spec first: EELS!
- Add EIP versioning to track implementations/definitions: EIP-7577.

Improve coverage:

- Ability to hook a fuzzer into test execution.
- Better documentation, more eyes on existing test coverage.

He protetc.
But he also attac.

Thank you!



Dimitry: `@winsvega` Mario: `@marioevz` spencer: `@spencer-tb` Dan: `@danceratopz`

Protocol Testers, Ethereum Foundation

ethereum/execution-spec-tests

contact us: repo → docs → getting_started → getting_help