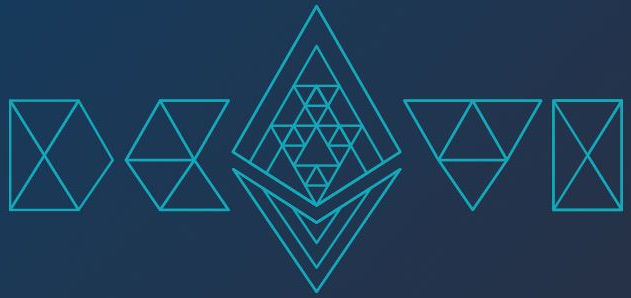


Testing the Merge

Testing Ethereum with Parallel Universes

David Searle

Head of EMEA

Doctor Strange

Primary protector of Earth
….adventures in bizarre worlds
and twisting dimensions

Cher

Goddess of Pop
….If I could find a way

David

25 years in tech industry
….now professional Bug Hunter!

My house

Things I’m selling on eBay

gpu

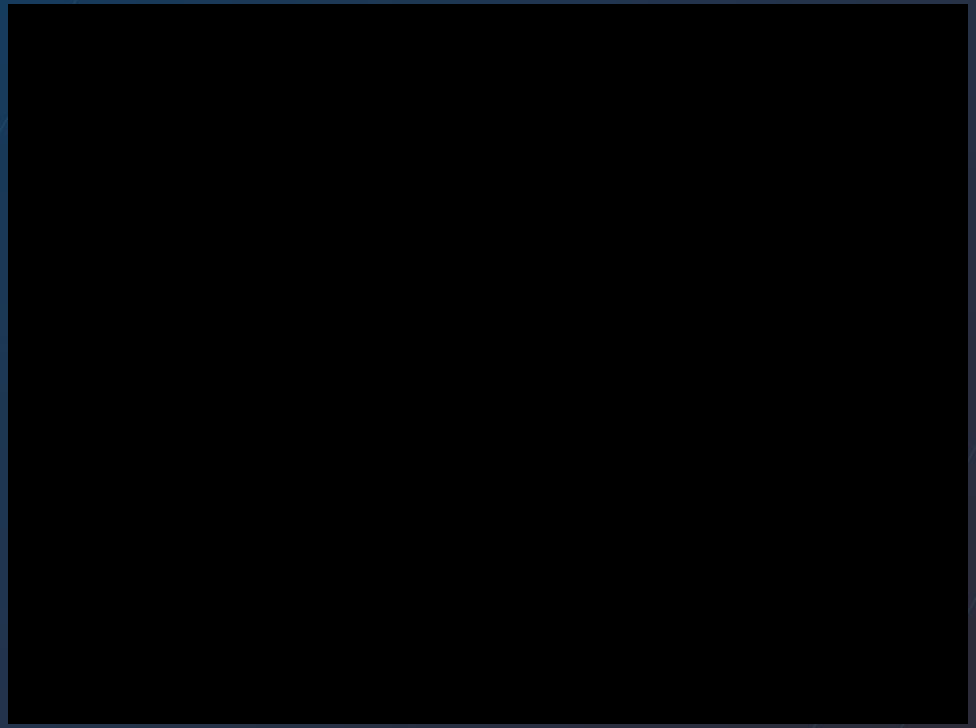
NVIDIA GeForce RTX 3090 Founders Edition 24GB GDDR6X GPU - FE
Pre-owned
£389.00
1 bid · 6d 12h
+ £20.00 postage

MSI NVIDIA Geforce GTX 1660 SUPER VENTUS XS OC - Graphics card GPU
Refurbished
£170.00
or Best Offer
Free next day postage
Get it by Tue 4 Oct
27 watchers
115 mi

Nvidia GTX Titan Xp 12GB GDDR5X GPU
Pre-owned
£100.00
1 bid · 6d 11h
Free postage
Click & Collect
125 mi

The Day of the Merge

Or this?



Quora

What makes software testing difficult?

Steven de Guzman
All my jobs include ensuring quality · Author has **1.5K** answers and **7.1M** answer views

In an imperfect IT world, software testing is difficult due to the **unfair expectation...**

Sridevi Balla
6y

... The real issue is really **reporting bugs nicely**. Let me tell you a story where Wife is the Developer and the Husband is the QA...

Youji Hajime
Ex game tester,currently a compliance officer in completely unrelated field.
Author has **1.4K** answers and **7.5M** answer views · 6y

... Fundamentally, this is **impossible requirement** because “absence of evidence is not the evidence of absence (of error, bugs, breach, etc)”...

Richard Dutcher
Former Enterprise Programmer at ADP (company) (2004–2013) · 3y

... So, testers have to think about **all possible scenarios** where issues may arise and ensure they are handled by the code.

Werayut Nasawaeng
10++ years in Software Quality Domain · Author has **751** answers

... Software Testing is always never easy because we can't have 100% test coverage. We couldn't test **every cases possible**. And we will need to deal with time and resources constraints.

Charitha Kankanamge
Manager, Software Quality at Amazon · 6y

... features which have been implemented without **considering testability** in advance...

Ethereum Testing

- Unit testing
- Testnets
  - Shadow-forks and testnet merges
  - Hive, Kurtosis, Antithesis
- Fuzzing
  - Coverage guided fuzzing on multiple portions of all CLs
  - Modified CL and EL clients for local-testnets and attack-nets.
- Static Analysis
  - manual code audits of clients, code queries (semgrep, codeql..)

Distributed Systems…

So what did we do…

- **Auto-generate** network, container, and other faults
- Fuzz **large systems**, not just the individual applications
- Ran all software **deterministically, across all clients at the same time**
- Used strategies to seek **rare events** (code coverage, events, log messages, etc.) and explored them as much as possible – better than random
- Used a **great exploration tool** set

Individually Tested

Consensus Clients


Lighthouse


Teku


Nimbus


Prysmatic Labs


Lodestar

Execution Clients


Go Ethereum


Nethermind


Hyperledger Besu


Erigon

Testnets & Shadow Forks

Consensus Clients


Lighthouse


Teku


Nimbus


Prysmatic Labs


Lodestar

Execution Clients


Go Ethereum

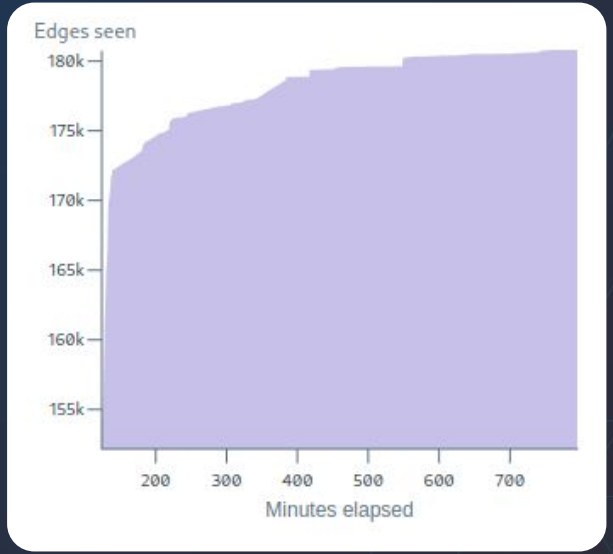

Nethermind


Hyperledger Besu

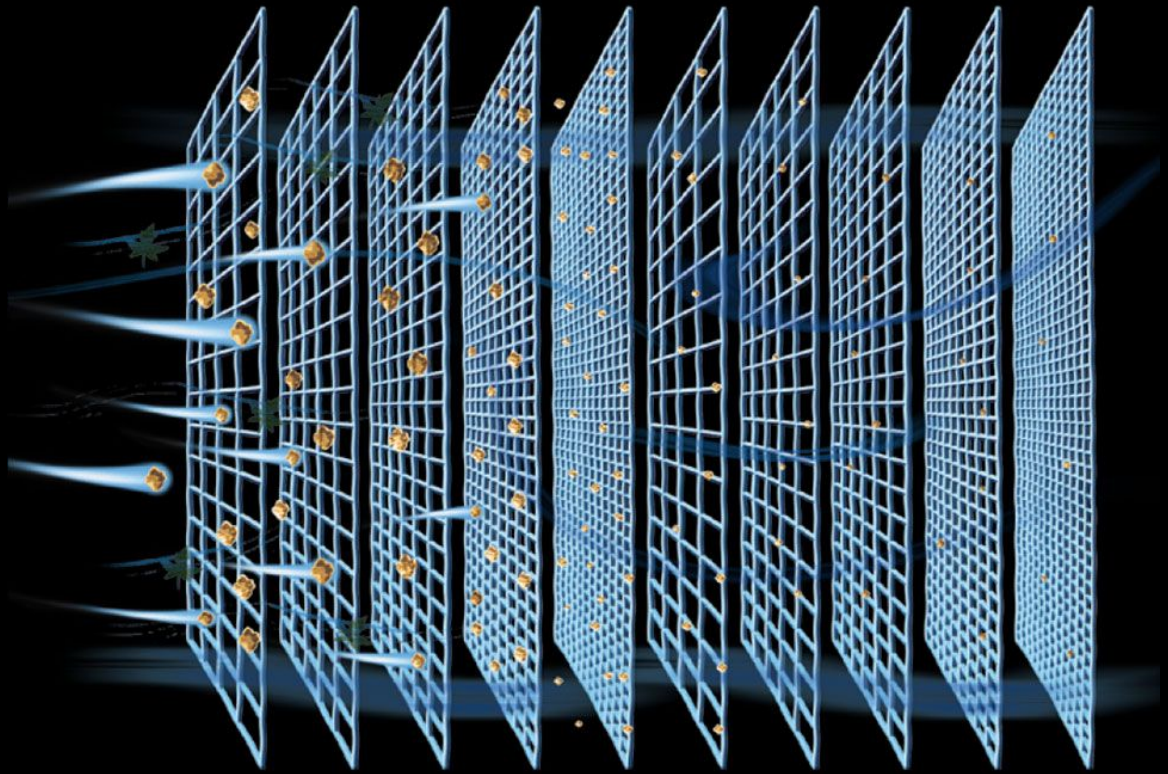

Erigon

One Simulation, many Parallel Universes

Test Branches	Test Hours (hrs)
858	536
Wall Clock (hrs)	**Edges Seen (unique)**
13	180,802

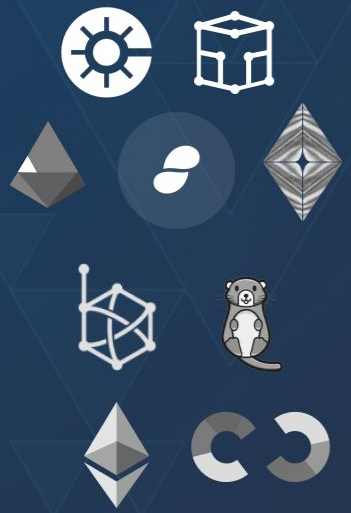


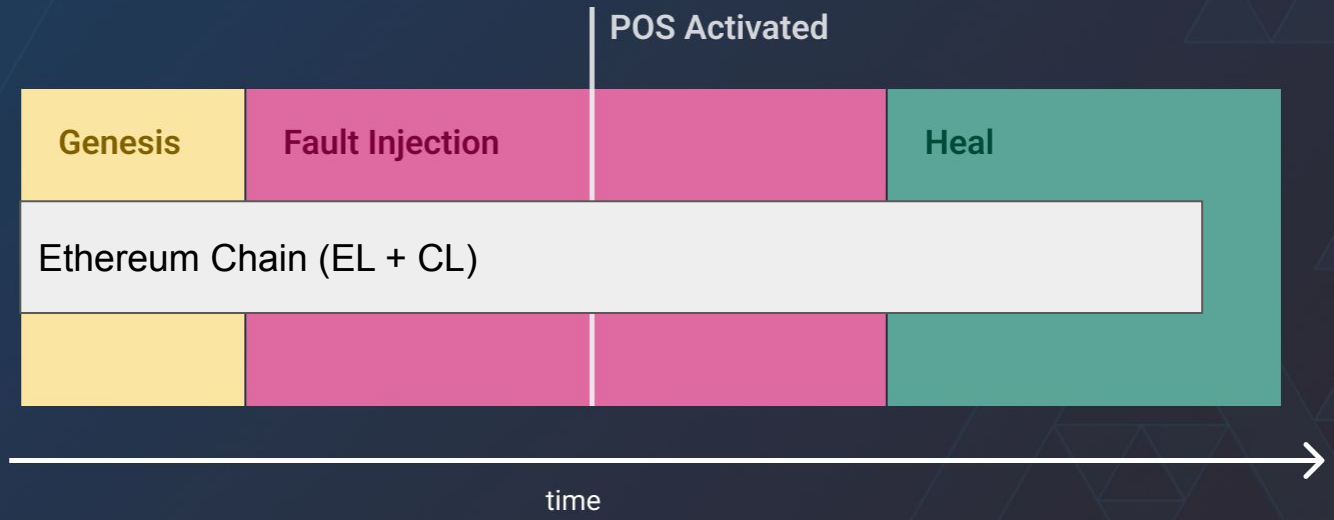
Reducing the risk of exploits at every step



 afepower.com/air-filter-media

Simulating the Merge





Faults

Network: Partitions, Delays, Drops

Nodes: Stops, Pause, Kills

Threads: Pause, Release

31 years of 24x7 nonstop testing

Over 50million edges

45 validated errors

33 logged bugs

How about an example?

Report on test suite “Ethereum - Multi-client--kilnv2__inst Fault Tolerance Experiment”

Conducted on Wednesday, July 27th, 2022 at 5:23 PM

Nightly experiment expected to run for 16 hours with the latest images in the registry matching the given tags. Client images are expected to have been instrumented as the experiment is set up to use a coverage strategy for exploring the states.

▶ Experiment configuration details

Test Results

Pass **At some point workload started successfully**

Pass **At some point there is at least one phase 3 success**

Pass **At some point geth reports leaving PoW stage**

Pass **Ultimately no panics**

Fail **Ultimately no seg faults**

Counterexamples

```
timestamp: 2402.57
source: service_nimbus-client-geth-0
rollout_id: 608
state_id: 119806
vtime: 10318950303345
text: SIGSEGV: Illegal storage access. (Attempt to read
from nil?)
```

Pass **Ultimately no impossible re-orgs**

Pass **Ultimately no nimbus panics**

Pass **Ultimately no lighthouse panics**

Pass **Ultimately no prysm panics**

Pass **Ultimately no lighthouse segfaults**

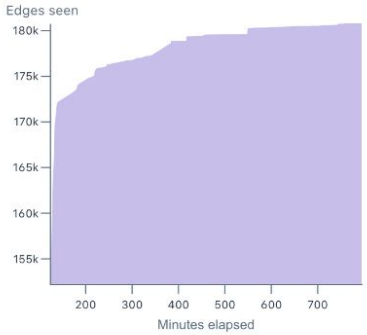
Pass **Ultimately no lodestar segfaults**

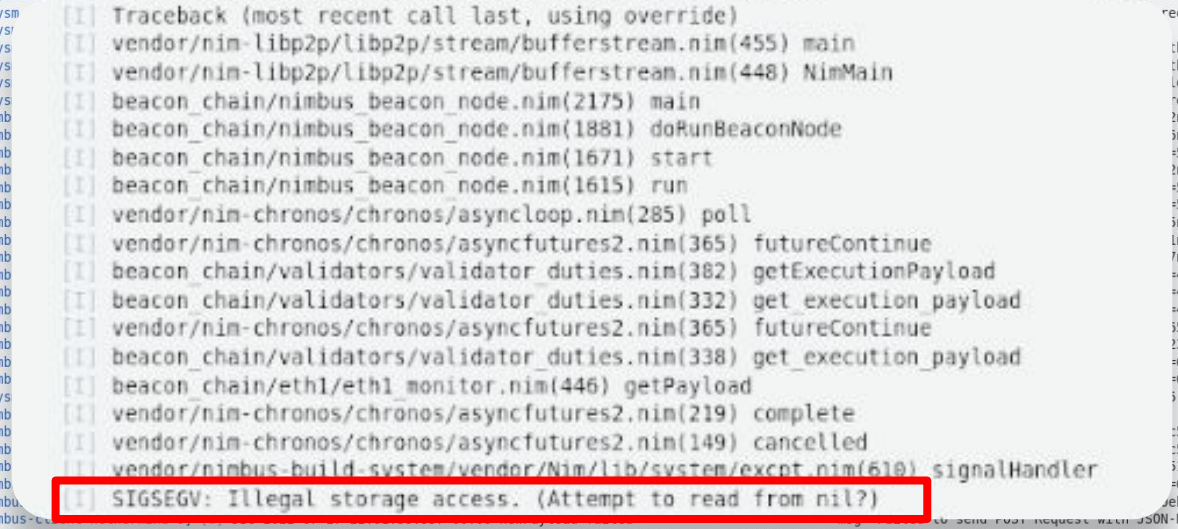
Fail **Ultimately no nimbus segfaults**

Pass **Ultimately no prysm segfaults**

Pass **Ultimately no teku segfaults**

Test Branches	Test Hours (hrs)
858	536
Wall Clock (hrs)	**Edges Seen (unique)**
13	180,802



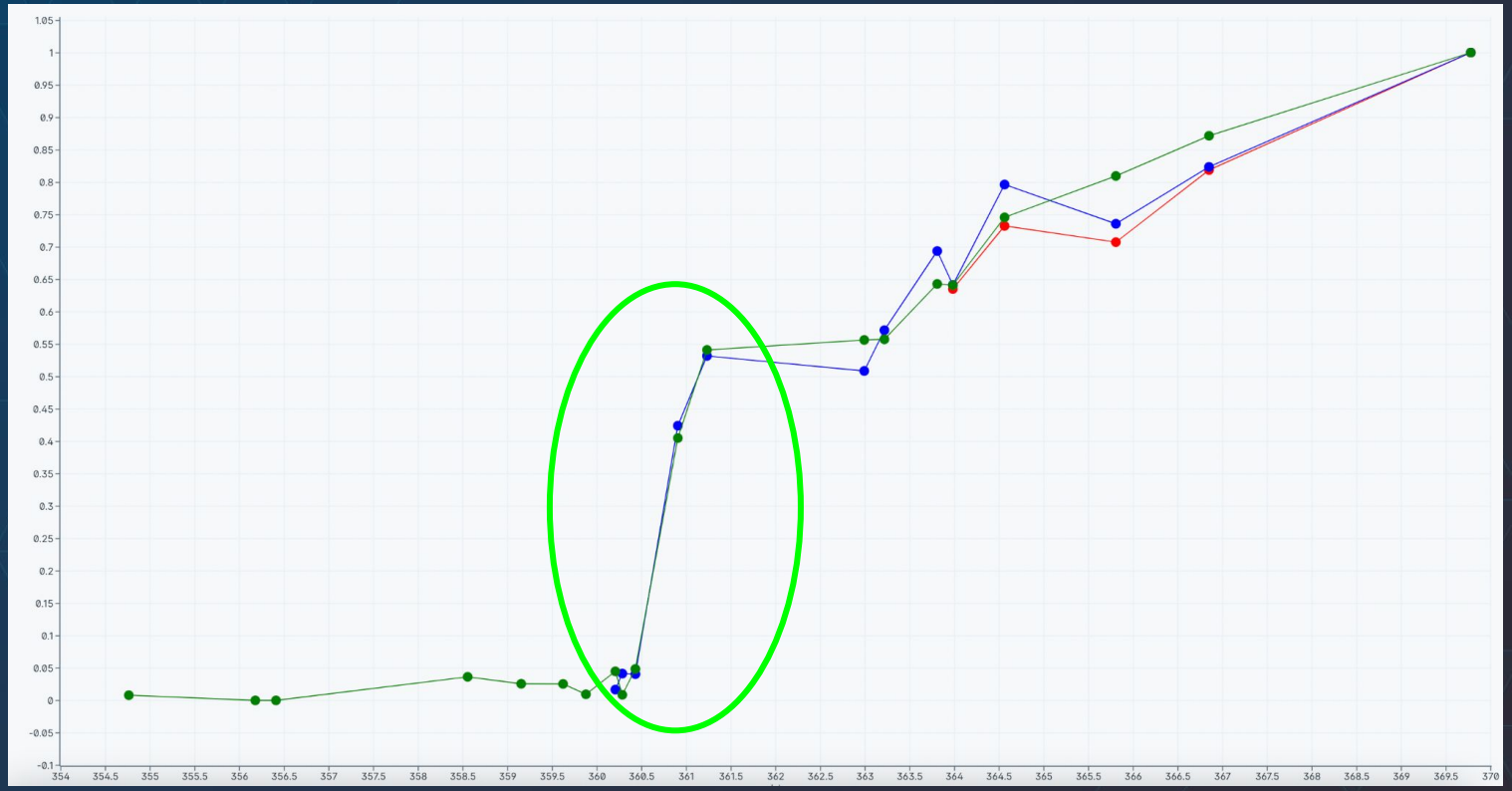

Traceback (most recent call last, using override)
vendor/nim-libp2p/libp2p/stream/bufferstream.nim(455) main
vendor/nim-libp2p/libp2p/stream/bufferstream.nim(448) NimMain
beacon_chain/nimbus_beacon_node.nim(2175) main
beacon_chain/nimbus_beacon_node.nim(1881) doRunBeaconNode
beacon_chain/nimbus_beacon_node.nim(1671) start
beacon_chain/nimbus_beacon_node.nim(1615) run
vendor/nim-chronos/chronos/asyncloop.nim(285) poll
vendor/nim-chronos/chronos/asyncfutures2.nim(365) futureContinue
beacon_chain/validators/validator_duties.nim(382) getExecutionPayload
beacon_chain/validators/validator_duties.nim(332) get_execution_payload
vendor/nim-chronos/chronos/asyncfutures2.nim(365) futureContinue
beacon_chain/validators/validator_duties.nim(338) get_execution_payload
beacon_chain/eth1/eth1_monitor.nim(446) getPayload
vendor/nim-chronos/chronos/asyncfutures2.nim(219) complete
vendor/nim-chronos/chronos/asyncfutures2.nim(149) cancelled
vendor/nimbus-build-system/vendor/Nim/lib/system/excpt.nim(610) signalHandler
SIGSEGV: Illegal storage access. (Attempt to read from nil?)


```
160 [service_prysm-client-nethermind-0]  [I] time="2022-05-27 11:29:36" level=debug msg="gRPC request finished." backend=[] duration=93.338072ms method="/et
161 [service_prysm-client-nethermind-0]  [I] time="2022-05-27 11:29:36" level=warning msg="Fee recipient is currently using the burn address, you will not b
162 [service_prysm-client-nethermind-0]  [I] time="2022-05-27 11:29:36" level=debug msg="gRPC request finished." backend=[] duration=117.162228ms method="/e
163 [service_prysm-client-nethermind-0]  [I] time="2022-05-27 11:29:36" level=debug msg="gRPC request finished." backend=[] duration=22.352µs method="/ether
164 [service_prysm-client-nethermind-0]  [I] time="2022-05-27 11:29:36" level=error msg="Could not request attestation to sign at slot" error="rpc error: co
165 [service_prysm-client-nethermind-0]  [I] time="2022-05-27 11:29:36" level=debug msg="gRPC request finished." backend=[] duration=22.352µs method="/ether
180 [service_prysm-client-nethermind-0]  [I] panic: runtime error: invalid memory address or nil pointer dereference
181 [service_prysm-client-nethermind-0]  [I] [signal SIGSEGV: segmentation violation code=0x1 addr=0x0 pc=0x14a64f9]
182 [service_prysm-client-nethermind-0]  [I]
183 [service_prysm-client-nethermind-0]  [I] goroutine 266 [running]:
184 [service_prysm-client-nethermind-0]  [I] github.com/prysmaticlabs/prysm/beacon-chain/blockchain.(*Service).notifyForkchoiceUpdate(0xc000342540, {0x21d8c
185 [service_prysm-client-nethermind-0]  [I] /git/src/github.com/prysmaticlabs/prysm/beacon-chain/blockchain/execution_engine.go:193 +0x1519
186 [service_prysm-client-nethermind-0]  [I] github.com/prysmaticlabs/prysm/beacon-chain/blockchain.(*Service).notifyEngineIfChangedHead(0xc000342540, {0x21
```

♫ If I could turn back time ♫

Look back **n seconds** … and turn on packet capture

But how far should you look back?



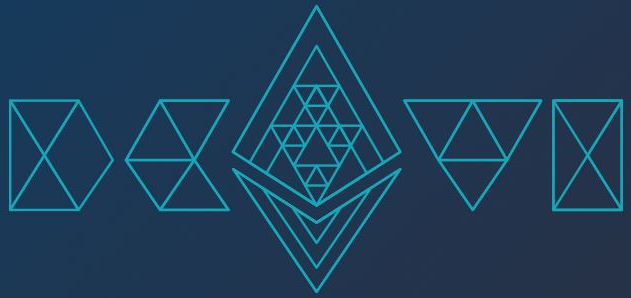
What's Next

- Post Merge Testing on Stable Branches
- EIP 4844 testing
  - Dank Sharding
  - Withdrawals
- Networking Testing Simulations (Gossip)
- API Fuzzing
- Byzantine / Malicious Client
- Merge code cleanup

Q & A

antithesis.com

tg: @signal1crypto



Thank you!

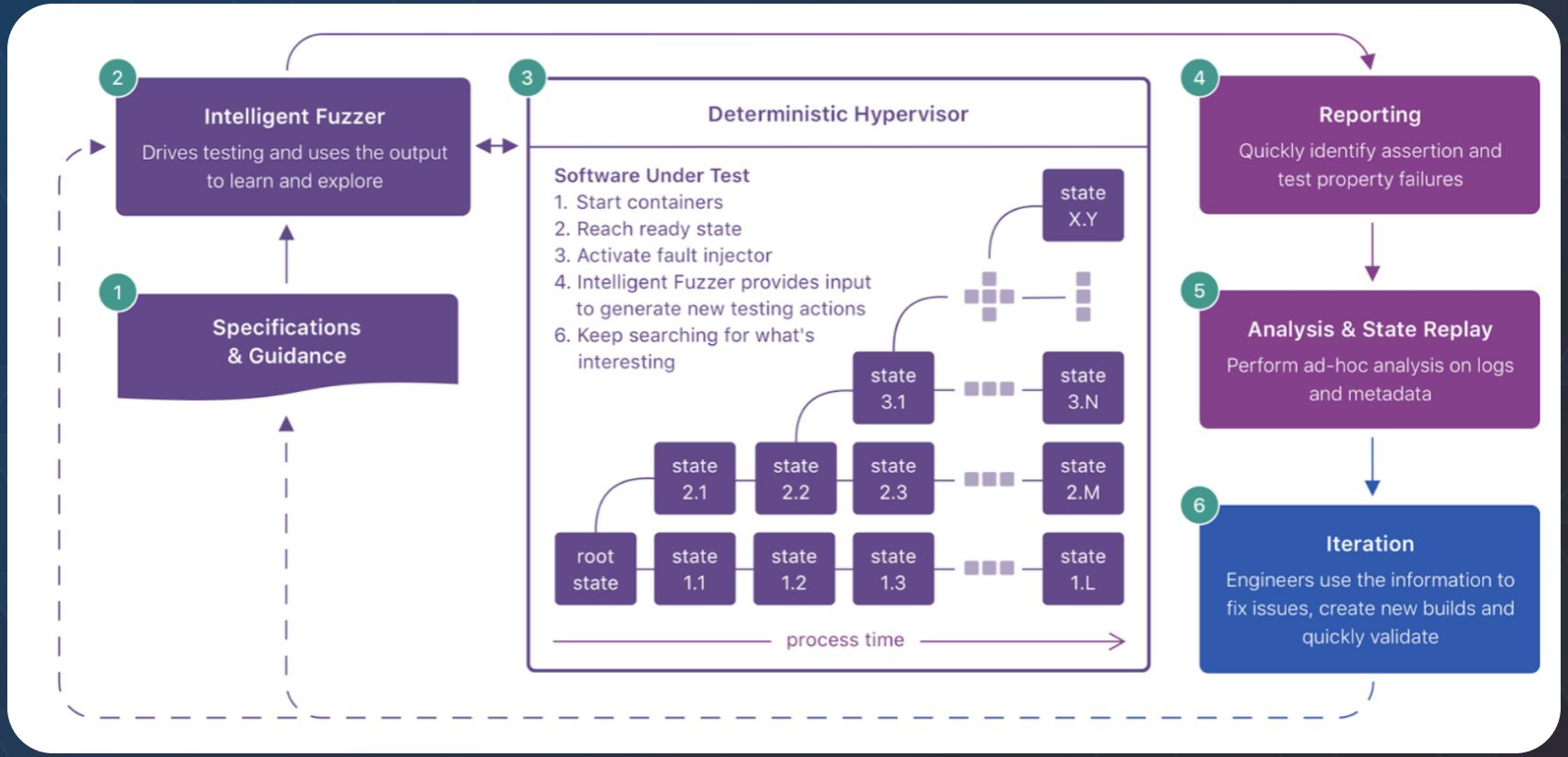
David Searle

European Field Operations at Antithesis

david.searle@antithesis.com

Fault Injection

- Network Faults
  - Events will either partition the network, or bring down individual nodes
  - Disruptions include latency, queuing packets until the end of event, or dropping all packets
- Thread pausing
  - Short pauses randomly inserted at any block of code
- Container faults
  - Pause (and resume) or kill (and restart) individual containers at randomized intervals
- User-defined scripts
  - The Fault Injector can be instructed to run scripts contained in the software or workload container


2
Intelligent Fuzzer
Drives testing and uses the output to learn and explore
1
Specifications & Guidance
3
Deterministic Hypervisor
Software Under Test
1. Start containers
2. Reach ready state
3. Activate fault injector
4. Intelligent Fuzzer provides input to generate new testing actions
6. Keep searching for what's interesting
root state
state 1.1
state 1.2
state 1.3
state 1.L
state 2.1
state 2.2
state 2.3
state 2.M
state 3.1
state 3.N
state X.Y
process time
4
Reporting
Quickly identify assertion and test property failures
5
Analysis & State Replay
Perform ad-hoc analysis on logs and metadata
6
Iteration
Engineers use the information to fix issues, create new builds and quickly validate

Reach vs Coverage