Killing ETH

Finding Consensus Issues on Layer 1

Marius van der Wijden
EF

Consensus

The Ethereum network is run by 4 execution layer clients and 5 consensus layer clients.

It's critically important that all implementations do the same thing.

If one implementation deviates from the spec, a chain split can occur.

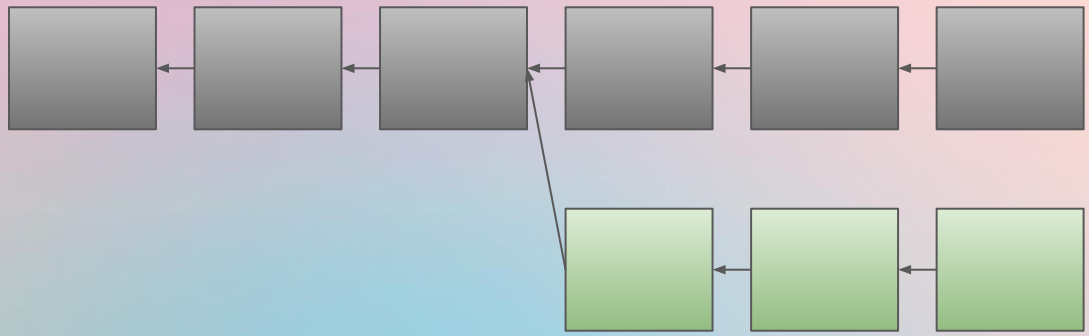
Consensus issues are differences between implementations and the spec

Testing Strategies

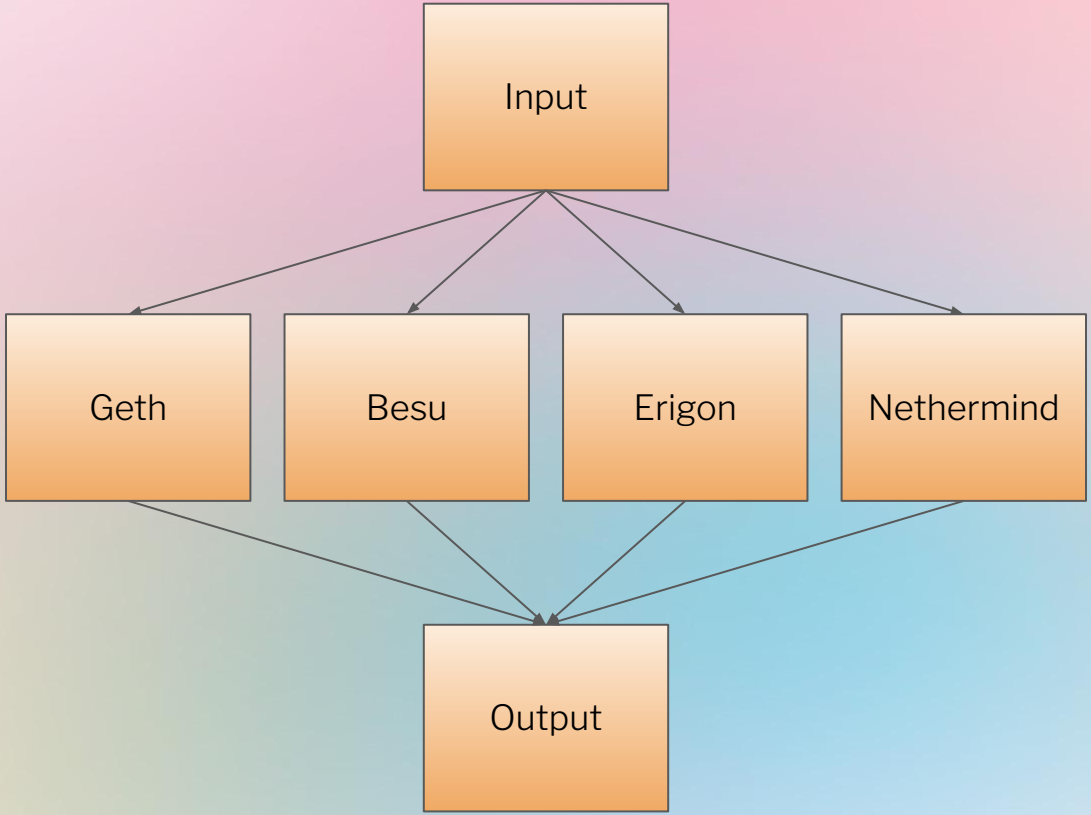
Shadowforks

We configure a subset of nodes with the new rules. At the fork point, the nodes create their own chain (with the same chainID). Transactions from the main chain are replayable.

Shadowforking allows for very effective tests since we have the same state and transaction load as mainnet.

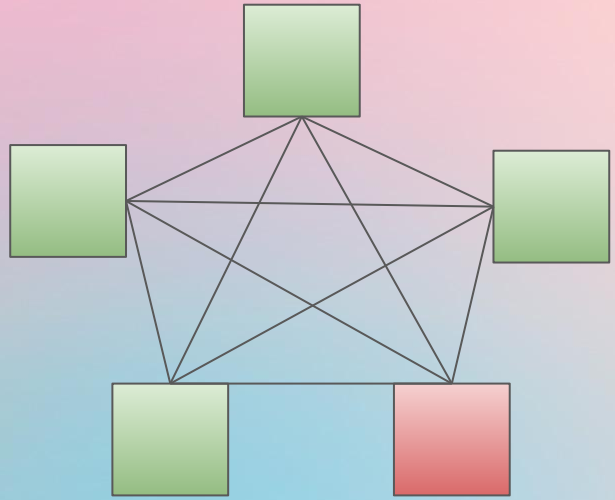


Differential Testing



Malicious Nodes

Malicious nodes can insert or alter transactions, change header fields, send really big values or nil values. Malicious CL nodes can double vote or fake signatures etc. They might send spoofed or fuzzed network packages in order to cause mayhem.



Testing Tools

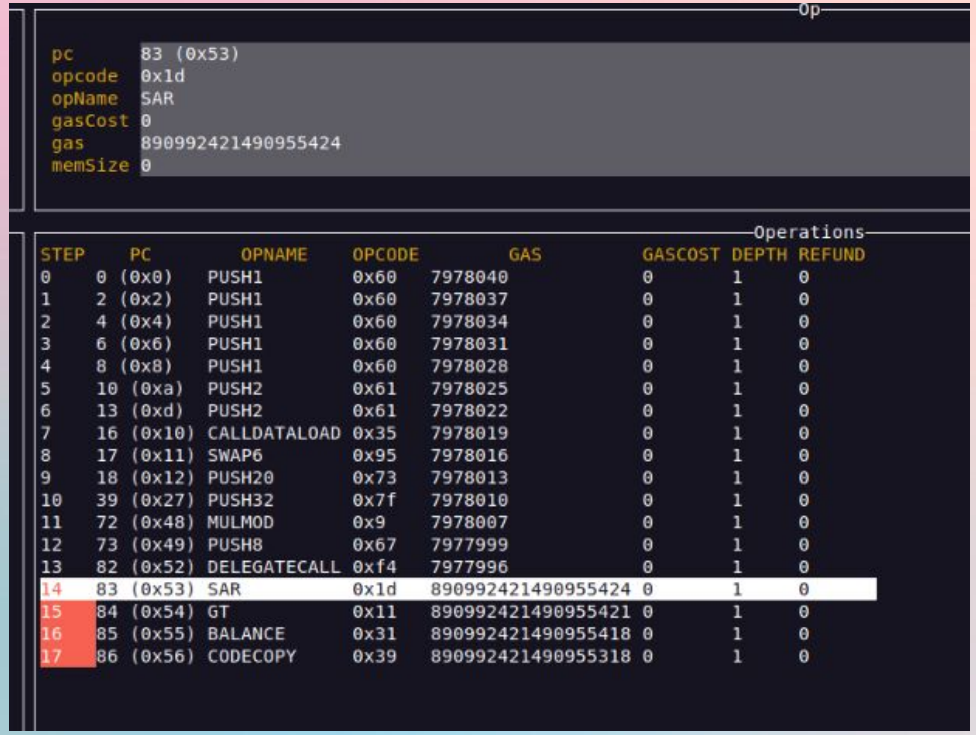
Name	Layer	Purpose
goevmlab	EL	Toolkit for EVM testing and fuzzing
Hive	EL/CL	Regression testing
FuzzyVM	EL	Differential EVM fuzzing
ethereum/tests	EL	Set of >48.000 state tests
Malicious CL client	CL	Malicious CL node
Merge-bad-block-gen	EL	Malicious EL node
Kurtosis	EL+CL	Nightly testnets
Antithesis	EL+CL	Deterministic fuzzing
tx-fuzz	EL	Send interesting transactions
merge-fuzz	EL	Fuzzes the engine api
Beacon fuzz	CL	Fuzzes the beacon client

Trophies

CVE-2022-36025

Found by holiman with go-evmlab

Issue with the gas-calculation in Besu. A CALL could underflow the returned gas which is a signed int32. Would result in a consensus issue on mainnet and DOS on Besu-only networks.



Death of Kintsugi

Invalid extradata

Nimbus was unable to sync because the bad block generator replaced the extradata in a block.

REVERT Opcode

Tx-fuzz created a transaction that triggered an issue in EthereumJS related to the REVERT opcode.

Three-way consensus split

A three-way consensus split between geth, teku-geth and besu/nethermind

Three-way consensus split

The fuzzer replaced the blockhash with its parenthash.

This block should be rejected on newPayload because the hash of the block doesn't match.

Besu did not have the check. Nethermind had the check but also cached the payloads and since the parenthash is in the cache, the payload was assumed valid.

Another block had the blocknumber set to 1.

Geth has a cache it checks to know whether we need to sync.

Teku executed all forks (~30) simultaneously which flushed the cache.

So we queried the DB (by parenthash and blocknumber - 1) which fails and triggers another sync cycle.

The sync violates some preconditions thus panic'ing and shutting down the node.

Hive

Hive found a bunch of issues in geth:

- Division by Zero in xchTxCfg
- TTD not checked on post-merge headers
- Timestamp > parent.Timestamp
- Modification of shallow copied TD
- Last PoW block must be the first one to pass TTD (twice)
- Bad blocks encountered during sync not available via engine API
- Multiple issues with return values and LatestValidHash
- ...

#TestingTheMerge

The easiest way to understand the Ethereum protocol is to start testing it.

Over 400 people got involved with #TestingTheMerge, sent transactions on the testnets, setup nodes, reporting issues and writing blog posts and documentations.

The community found the following issues in go-ethereum:

- TTD override not correctly applied to empty db
- TTD override overflows on values > Uint64Max

Shadowforks

Default Gas Limit

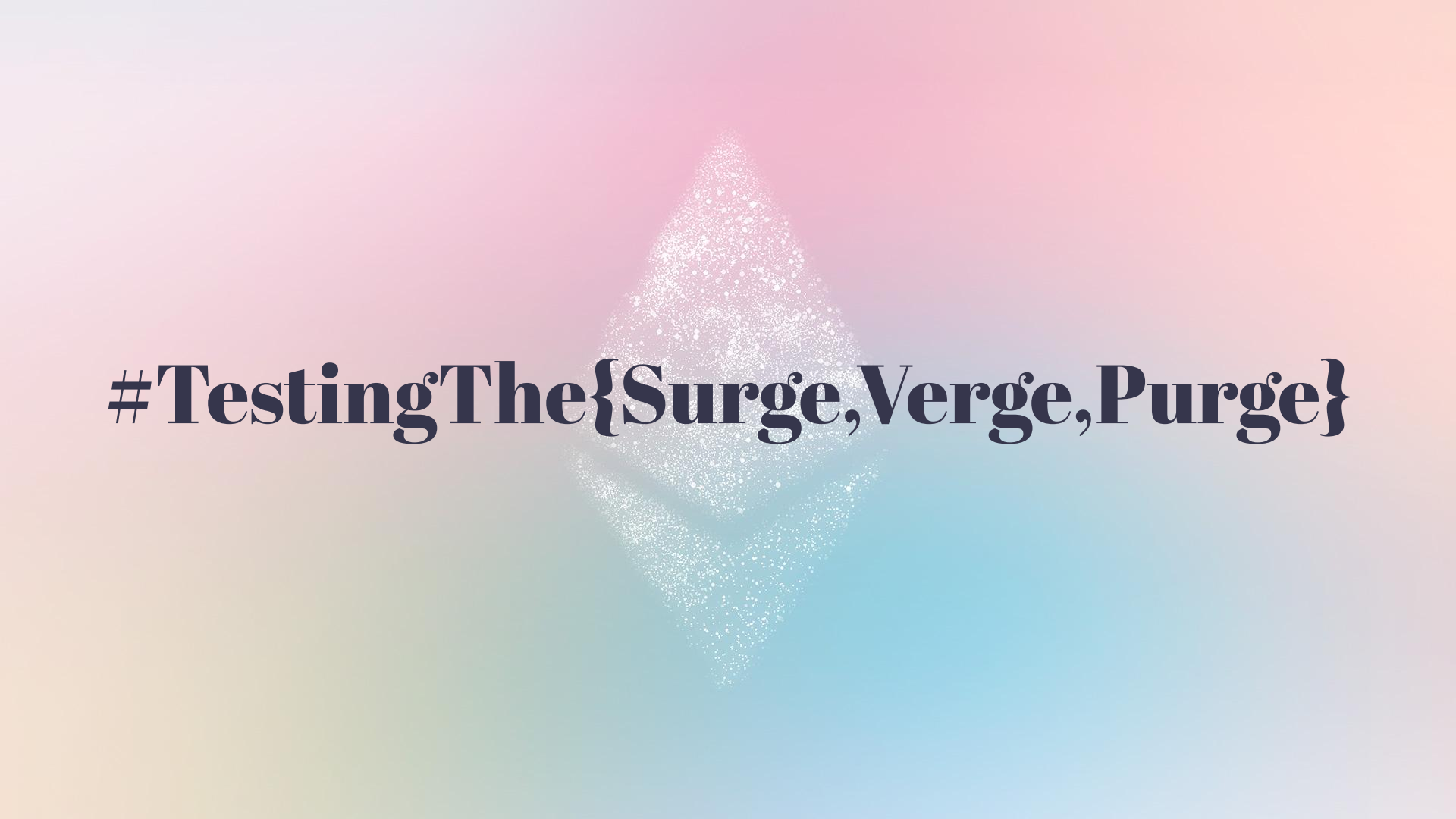
The default gas limit in geth was set to 8M, so after the first mainnet shadow fork the gas limit was quickly voted down by validators.

Memory blowup during reorgs

A different issue in geth forced nodes to do 600.000 block reorgs which exceeded available memory. Rewriting the reorg operation fixed this issue.

Endianness of Basefee

Prysm used the wrong endianness for the basefee, thus creating bad blocks on a shadowfork, when basefee > 255.


#TestingThe{Surge,Verge,Purge}

Special Thanks to

Peter and the go-ethereum team.
Mario and the ethereum/tests team.
Parithosh and the EF devops team.
Danny and the CL security team.
Marek, Lukasz and the Nethermind team.
Gary, Danno and the Besu team.
Andrew, Giulio and the Erigon team.
The Lighthouse, Prysm, Teku, Lodestar and Nimbus teams
The merge-devnet-debug group.
Everyone who participated in#TestingTheMerge.

And Martin Holst Swende who keeps this network alive.

Big thanks to the people that get up in the middle of the night to investigate consensus issues.

If you want to join the testing efforts contact mario.vega@ethereum.org

For when I inevitably have too much time on my hands

Backup Slides

If you find a bug (even an already fixed one) please don't trigger it on mainnet!

The bug that took down Infura

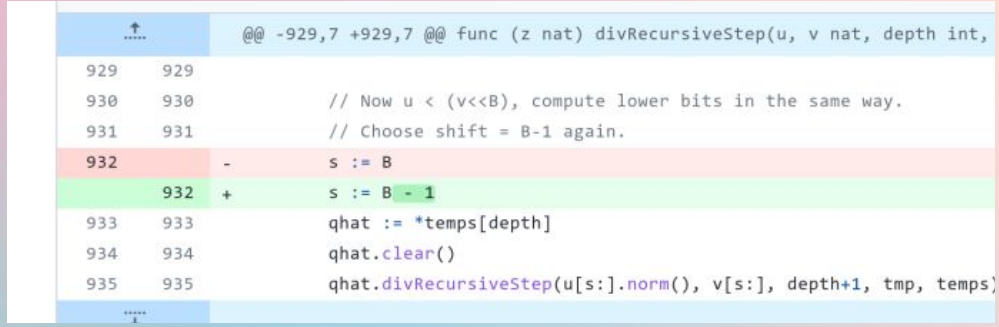
Found by John Youngseok Yang

Memory returned by RETURNDATACOPY was shallow copied and allowed modifications of the underlying array. The bug was discovered and quickly fixed but not announced. Someone found the bug and deliberately triggered it on mainnet, because only 1% of nodes were affected. These nodes happen to include infura.

Go math/big panic

Found by OSS-Fuzz

The ModExp precompile internally uses the golang standard library. If the modulo was larger than 6336 bits on 64-bit architectures, the standard library would panic, causing the geth node to crash.



```
@@ -929,7 +929,7 @@ func (z nat) divRecursiveStep(u, v nat, depth int,
929  929
930  930          // Now u < (v<<B), compute lower bits in the same way.
931  931          // Choose shift = B-1 again.
932       -       s := B
     932  +       s := B - 1
933  933          qhat := *temps[depth]
934  934          qhat.clear()
935  935          qhat.divRecursiveStep(u[s:].norm(), v[s:], depth+1, tmp, temps)
```

SMOD consensus flaw

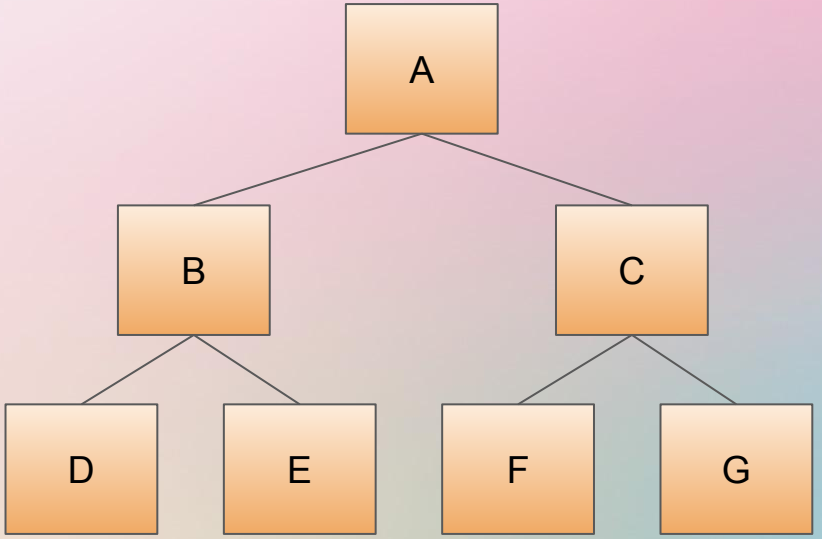
Submitted via bug bounty program

Nethermind failed to handle the case of int256.min (2^255) correctly. It would negate the result twice, resulting in 2 instead of -2.

```
pragma solidity ^0.8.7;

contract SMOD {
    event Result(int256);

    constructor() {
        int256 a = type(int256).min;
        int256 b = -3;
        int256 c = a % b;
        emit Result(c);
        // require(c == -2, "consensus error");
    }
}
```



DoS via malicious snap request

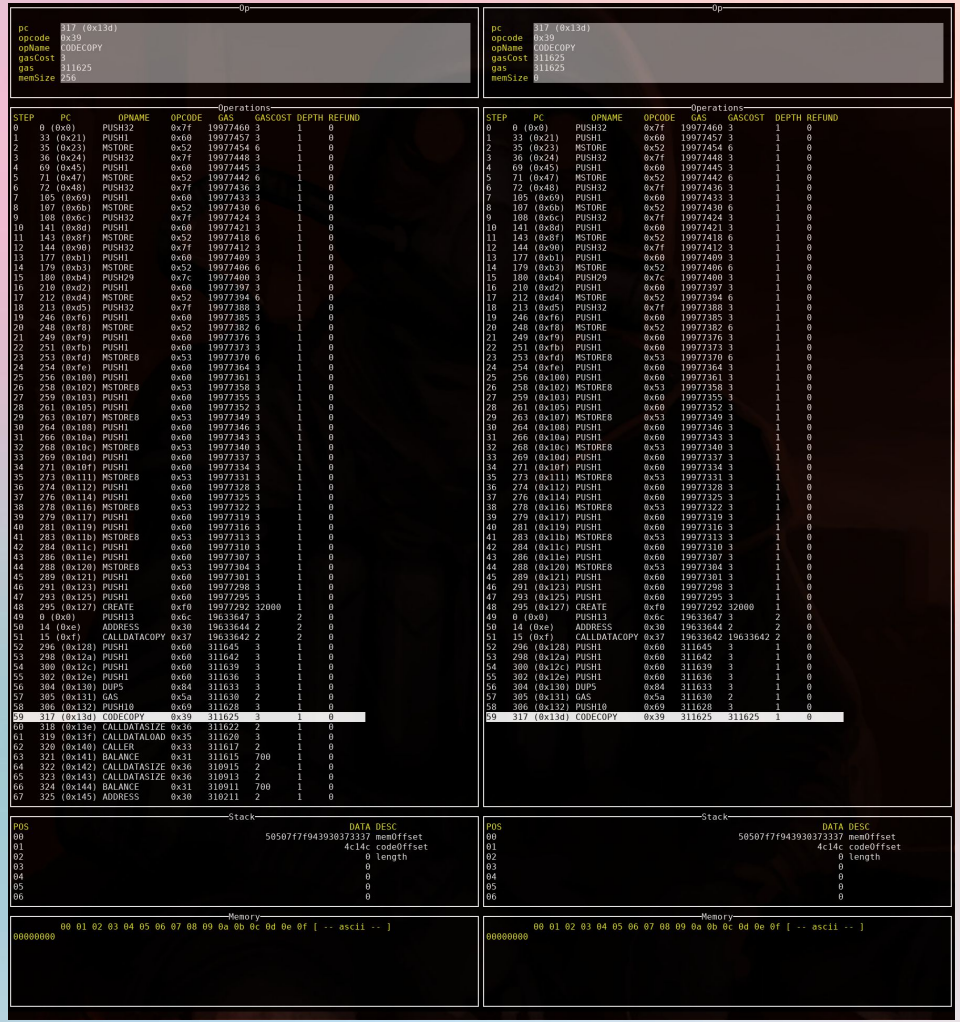
Found by Gary Rong and holiman

Geth 1.10.9 contained a fix for two panic in the snap handler. A carefully crafted `GetTrieNodes` package that requested a missing trie node could crash a geth node. Fuzzing it found a second panic if a non-existent account was requested.

COPY Opcodes

Found by FuzzyVM

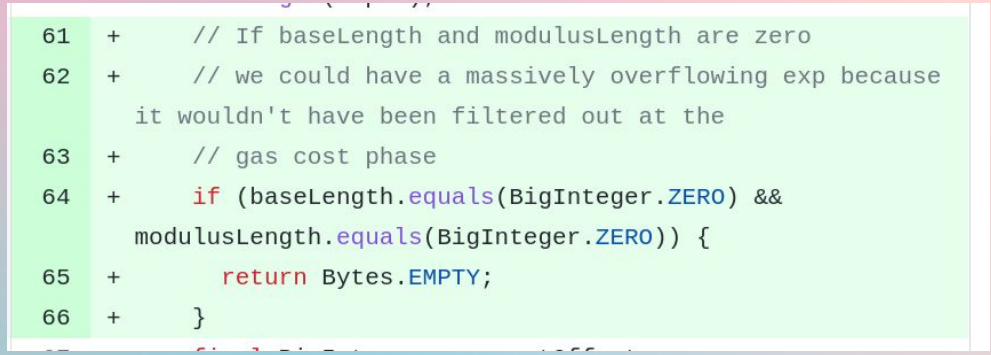
The *COPY opcodes (CALLDATACOPY, CODECOPY, EXTCODECOPY, RETURNDATACOPY) consume three items from the stack; destination, source and length of the data to copy. Nethermind halted execution if length was zero for them.



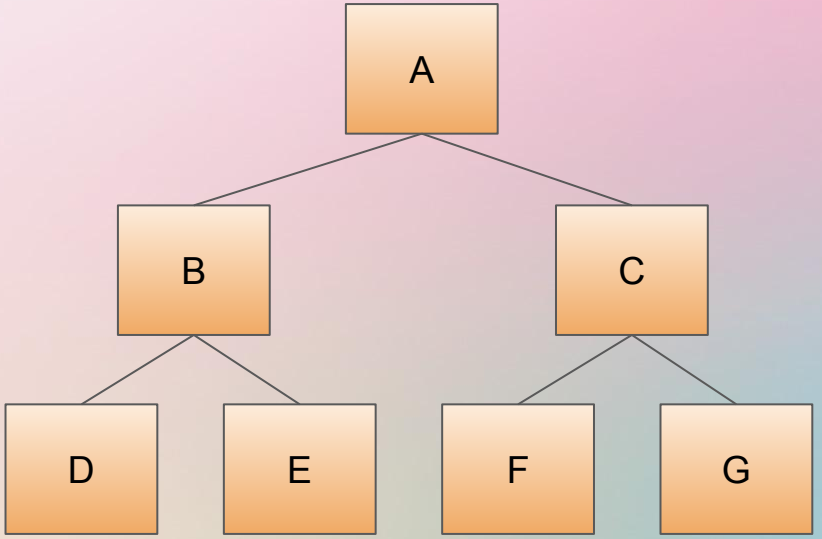
ModExp Crash

Found by FuzzyVM

The ModExp precompile consumes six parameters; base, modulus and exponent length as well as the base modulus and exponent themselves. Besu would read the other parameters even if base length and modulus length were zero, resulting in an overflow.



```
61 +     // If baseLength and modulusLength are zero
62 +     // we could have a massively overflowing exp because
   it wouldn't have been filtered out at the
63 +     // gas cost phase
64 +     if (baseLength.equals(BigInteger.ZERO) &&
   modulusLength.equals(BigInteger.ZERO)) {
65 +       return Bytes.EMPTY;
66 +     }
```



Geth v.1.10.22 bug

Geth 1.10.22 contained a regression that could corrupt the local state. The trie nodes are flushed to disk in the same order as they are inserted into the dirty cache in memory. The order in which we inserted them into the dirty cache was wrong (random) so we ended up with dangling trie nodes.

Thank you!

Marius van der Wijden
EF
marius@ethereum.org
@vdwijden