



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**

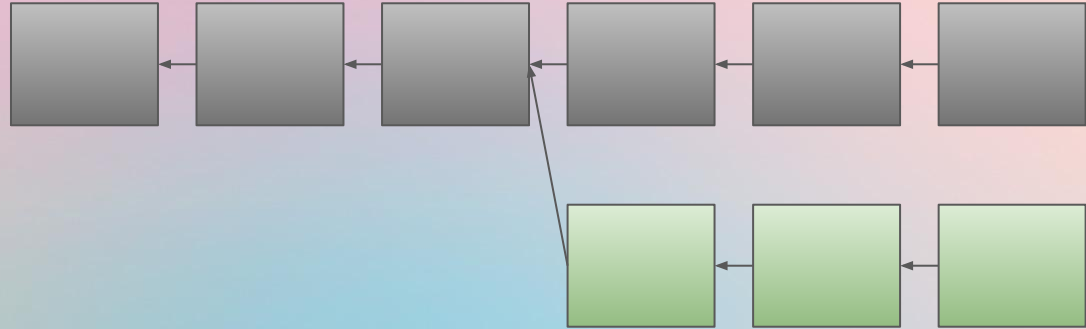


Testing Strategies

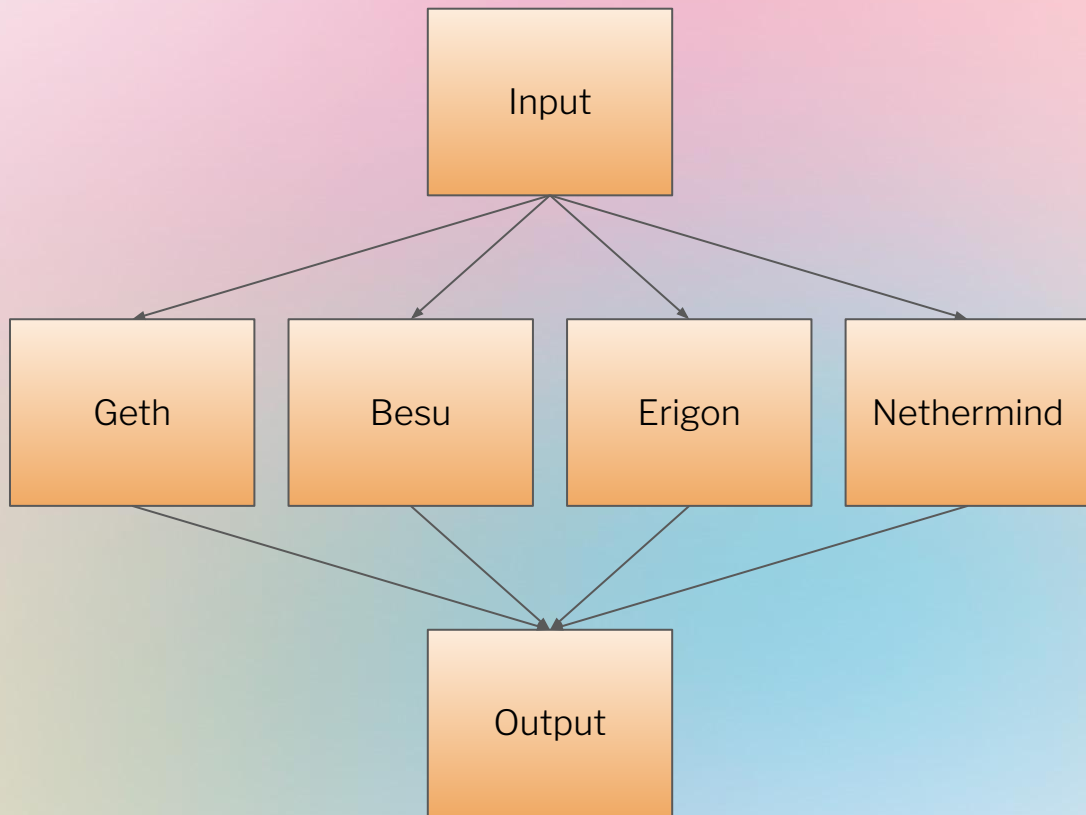
Shadowforks

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.

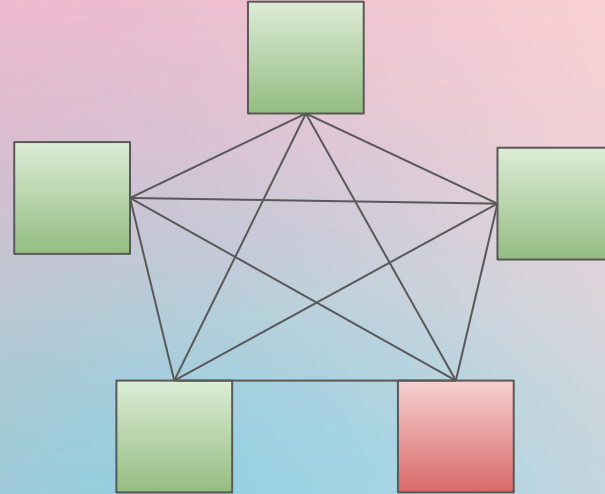


Differential Testing



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.

Op

pc	83 (0x53)						
opcode	0x1d						
opName	SAR						
gasCost	0						
gas	890992421490955424						
memSize	0						

Operations							
STEP	PC	OPNAME	OPCODE	GAS	GASCOST	DEPTH	REFUND
0	0 (0x0)	PUSH1	0x60	7978040	0	1	0
1	2 (0x2)	PUSH1	0x60	7978037	0	1	0
2	4 (0x4)	PUSH1	0x60	7978034	0	1	0
3	6 (0x6)	PUSH1	0x60	7978031	0	1	0
4	8 (0x8)	PUSH1	0x60	7978028	0	1	0
5	10 (0xa)	PUSH2	0x61	7978025	0	1	0
6	13 (0xd)	PUSH2	0x61	7978022	0	1	0
7	16 (0x10)	CALLDATALOAD	0x35	7978019	0	1	0
8	17 (0x11)	SWAP6	0x95	7978016	0	1	0
9	18 (0x12)	PUSH20	0x73	7978013	0	1	0
10	39 (0x27)	PUSH32	0x7f	7978010	0	1	0
11	72 (0x48)	MULMOD	0x9	7978007	0	1	0
12	73 (0x49)	PUSH8	0x67	7977999	0	1	0
13	82 (0x52)	DELEGATECALL	0xf4	7977996	0	1	0
14	83 (0x53)	SAR	0x1d	890992421490955424	0	1	0
15	84 (0x54)	GT	0x11	890992421490955421	0	1	0
16	85 (0x55)	BALANCE	0x31	890992421490955418	0	1	0
17	86 (0x56)	CODECOPY	0x39	890992421490955318	0	1	0

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 $\text{basefee} > 255$.



#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

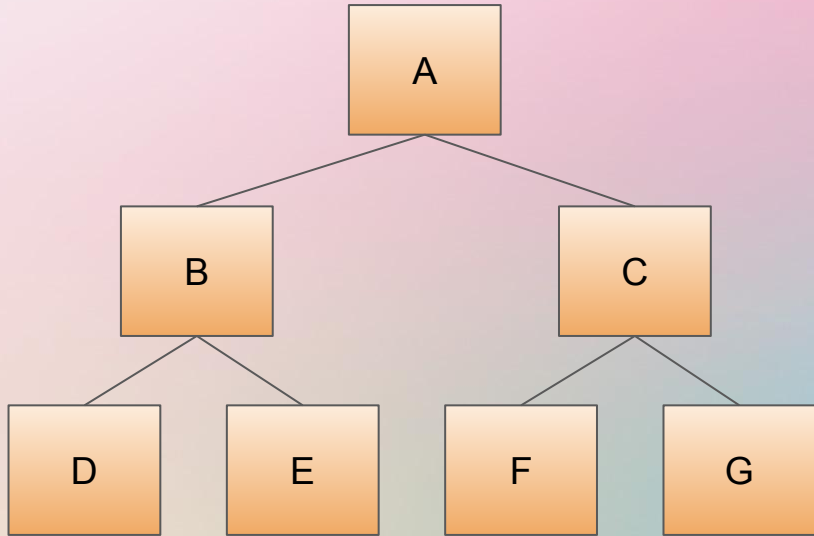
Submitted via bug bounty program

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.

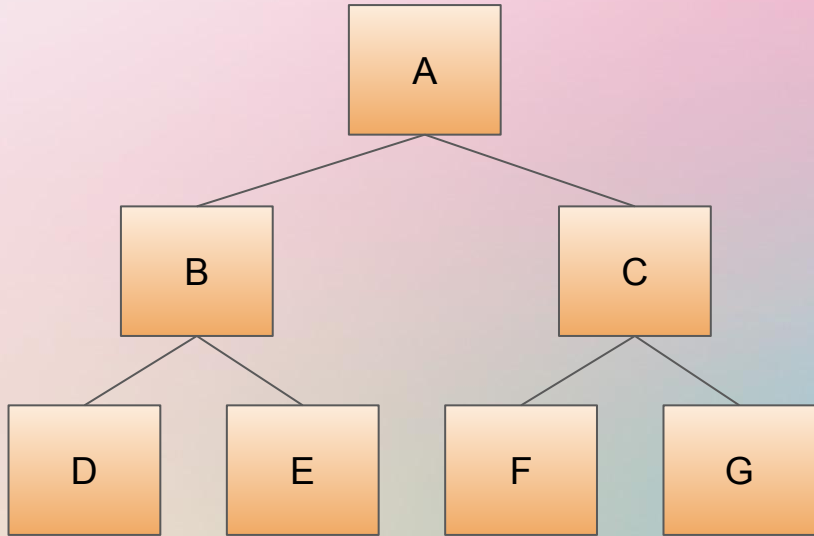
Op										Op									
pc					[17 (0x13d)]					pc					[17 (0x13d)]				
opcode					0x30					opcode					0x30				
opcode					CODECOPY					opcode					CODECOPY				
gasCost					3					gasCost					311625				
gas					311625					gas					311625				
memSize					0					memSize					0				
Operations										Operations									
STEP	PC	OPNAME	OPCODE	GAS	GASCOST	DEPTH	REFUND			STEP	PC	OPNAME	OPCODE	GAS	GASCOST	DEPTH	REFUND		
0	0 (0x0)	PUSH32	0x7f	19977460	3	1	0			0	0 (0x0)	PUSH32	0x7f	19977460	3	1	0		
1	33 (0x21)	PUSH1	0x60	19977457	3	1	0			1	33 (0x21)	PUSH1	0x60	19977457	3	1	0		
2	35 (0x23)	MSTORE	0x52	19977454	6	1	0			2	35 (0x23)	MSTORE	0x52	19977454	6	1	0		
3	36 (0x24)	PUSH32	0x7f	19977448	3	1	0			3	36 (0x24)	PUSH32	0x7f	19977448	3	1	0		
4	69 (0x45)	PUSH1	0x60	19977445	3	1	0			4	69 (0x45)	PUSH1	0x60	19977445	3	1	0		
5	71 (0x47)	MSTORE	0x52	19977442	6	1	0			5	71 (0x47)	MSTORE	0x52	19977442	6	1	0		
6	72 (0x48)	PUSH32	0x7f	19977436	3	1	0			6	72 (0x48)	PUSH32	0x7f	19977436	3	1	0		
7	105 (0x69)	PUSH1	0x60	19977433	3	1	0			7	105 (0x69)	PUSH1	0x60	19977433	3	1	0		
8	107 (0x6b)	MSTORE	0x52	19977430	6	1	0			8	107 (0x6b)	MSTORE	0x52	19977430	6	1	0		
9	108 (0x6c)	PUSH32	0x7f	19977424	3	1	0			9	108 (0x6c)	PUSH32	0x7f	19977424	3	1	0		
10	141 (0x8d)	PUSH1	0x60	19977421	3	1	0			10	141 (0x8d)	PUSH1	0x60	19977421	3	1	0		
11	143 (0x8f)	MSTORE	0x52	19977418	6	1	0			11	143 (0x8f)	MSTORE	0x52	19977418	6	1	0		
12	144 (0x90)	PUSH32	0x7f	19977412	3	1	0			12	144 (0x90)	PUSH32	0x7f	19977412	3	1	0		
13	177 (0xab)	PUSH1	0x60	19977409	3	1	0			13	177 (0xab)	PUSH1	0x60	19977409	3	1	0		
14	179 (0xab)	MSTORE	0x52	19977406	6	1	0			14	179 (0xab)	MSTORE	0x52	19977406	6	1	0		
15	180 (0xab)	PUSH29	0xc7c	19977400	3	1	0			15	180 (0xab)	PUSH29	0xc7c	19977400	3	1	0		
16	218 (0xd2)	PUSH1	0x60	19977397	3	1	0			16	218 (0xd2)	PUSH1	0x60	19977397	3	1	0		
17	212 (0xd4)	MSTORE	0x52	19977394	6	1	0			17	212 (0xd4)	MSTORE	0x52	19977394	6	1	0		
18	213 (0xd5)	PUSH32	0x7f	19977388	3	1	0			18	213 (0xd5)	PUSH32	0x7f	19977388	3	1	0		
19	246 (0xf6)	PUSH1	0x60	19977385	3	1	0			19	246 (0xf6)	PUSH1	0x60	19977385	3	1	0		
20	248 (0xf8)	MSTORE	0x52	19977382	6	1	0			20	248 (0xf8)	MSTORE	0x52	19977382	6	1	0		
21	249 (0xf9)	PUSH1	0x60	19977376	3	1	0			21	249 (0xf9)	PUSH1	0x60	19977376	3	1	0		
22	251 (0xfb)	PUSH1	0x60	19977373	3	1	0			22	251 (0xfb)	PUSH1	0x60	19977373	3	1	0		
23	253 (0xfd)	MSTORE8	0x53	19977370	6	1	0			23	253 (0xfd)	MSTORE8	0x53	19977370	6	1	0		
24	254 (0xfe)	PUSH1	0x60	19977364	3	1	0			24	254 (0xfe)	PUSH1	0x60	19977364	3	1	0		
25	256 (0x100)	PUSH1	0x60	19977361	3	1	0			25	256 (0x100)	PUSH1	0x60	19977361	3	1	0		
26	258 (0x102)	MSTORE8	0x53	19977358	3	1	0			26	258 (0x102)	MSTORE8	0x53	19977358	3	1	0		
27	259 (0x103)	PUSH1	0x60	19977355	3	1	0			27	259 (0x103)	PUSH1	0x60	19977355	3	1	0		
28	261 (0x105)	PUSH1	0x60	19977352	3	1	0			28	261 (0x105)	PUSH1	0x60	19977352	3	1	0		
29	263 (0x107)	MSTORE8	0x53	19977349	3	1	0			29	263 (0x107)	MSTORE8	0x53	19977349	3	1	0		
30	264 (0x108)	PUSH1	0x60	19977346	3	1	0			30	264 (0x108)	PUSH1	0x60	19977346	3	1	0		
31	266 (0x10a)	PUSH1	0x60	19977343	3	1	0			31	266 (0x10a)	PUSH1	0x60	19977343	3	1	0		
32	268 (0x10c)	MSTORE8	0x53	19977340	3	1	0			32	268 (0x10c)	MSTORE8	0x53	19977340	3	1	0		
33	269 (0x10d)	PUSH1	0x60	19977337	3	1	0			33	269 (0x10d)	PUSH1	0x60	19977337	3	1	0		
34	271 (0x10f)	PUSH1	0x60	19977334	3	1	0			34	271 (0x10f)	PUSH1	0x60	19977334	3	1	0		
35	273 (0x111)	MSTORE8	0x53	19977331	3	1	0			35	273 (0x111)	MSTORE8	0x53	19977331	3	1	0		
36	274 (0x112)	PUSH1	0x60	19977328	3	1	0			36	274 (0x112)	PUSH1	0x60	19977328	3	1	0		
37	276 (0x114)	PUSH1	0x60	19977325	3	1	0			37	276 (0x114)	PUSH1	0x60	19977325	3	1	0		
38	278 (0x116)	MSTORE8	0x53	19977322	3	1	0			38	278 (0x116)	MSTORE8	0x53	19977322	3	1	0		
39	279 (0x117)	PUSH1	0x60	19977319	3	1	0			39	279 (0x117)	PUSH1	0x60	19977319	3	1	0		
40	281 (0x119)	PUSH1	0x60	19977316	3	1	0			40	281 (0x119)	PUSH1	0x60	19977316	3	1	0		
41	283 (0x11b)	MSTORE8	0x53	19977313	3	1	0			41	283 (0x11b)	MSTORE8	0x53	19977313	3	1	0		
42	284 (0x11c)	PUSH1	0x60	19977310	3	1	0			42	284 (0x11c)	PUSH1	0x60	19977310	3	1	0		
43	286 (0x11e)	PUSH1	0x60	19977307	3	1	0			43	286 (0x11e)	PUSH1	0x60	19977307	3	1	0		
44	288 (0x120)	MSTORE8	0x53	19977304	3	1	0			44	288 (0x120)	MSTORE8	0x53	19977304	3	1	0		
45	289 (0x121)	PUSH1	0x60	19977301	3	1	0			45	289 (0x121)	PUSH1	0x60	19977301	3	1	0		
46	291 (0x123)	PUSH1	0x60	19977298	3	1	0			46	291 (0x123)	PUSH1	0x60	19977298	3	1	0		
47	293 (0x125)	PUSH1	0x60	19977295	3	1	0			47	293 (0x125)	PUSH1	0x60	19977295	3	1	0		
48	295 (0x127)	CREATE1	0xf6	19977292 32080						48	295 (0x127)	CREATE1	0xf6	19977292 32080					
49	0 (0x0)	PUSH13	0xc6c	19633647	3	2	0			49	0 (0x0)	PUSH13	0xc6c	19633647	3	2	0		
50	14 (0x0e)	ADDRESS	0x30	19633644	2	2	0			50	14 (0x0e)	ADDRESS	0x30	19633644	2	2	0		
51	15 (0x0f)	CALLDATACOPY	0x37	19633642	2	0	0			51	15 (0x0f)	CALLDATACOPY	0x37	19633642 19633642	2	0	0		
52	296 (0x128)	PUSH1	0x60	311645	3	1	0			52	296 (0x128)	PUSH1	0x60	311645	3	1	0		
53	298 (0x12a)	PUSH1	0x60	311642	3	1	0			53	298 (0x12a)	PUSH1	0x60	311642	3	1	0		
54	300 (0x12c)	PUSH1	0x60	311639	3	1	0			54	300 (0x12c)	PUSH1	0x60	311639	3	1	0		
55	302 (0x12e)	PUSH1	0x60	311636	3	1	0			55	302 (0x12e)	PUSH1	0x60	311636	3	1	0		
56	304 (0x130)	DUPS	0x5a	311633	3	1	0			56	304 (0x130)	DUPS	0x5a	311633	3	1	0		
57	305 (0x131)	GAS	0x5a	311630	2	1	0			57	305 (0x131)	GAS	0x5a	311630	2	1	0		
58	306 (0x132)	PUSH10	0x60	311628	3	1	0			58	306 (0x132)	PUSH10	0x60	311628	3	1	0		
59	317 (0x13d)	CODECOPY	0x30	311625	3	1	0			59	317 (0x13d)	CODECOPY	0x30	311625 311625	3	1	0		
Memory										Memory									
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f [-- ascii --]										00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f [-- ascii --]									
00000000										00000000									

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