Security Advisory: SP1 V6 Recursion Circuit Row-Count Binding Gap

High **tamirhemo** published **GHSA-63x8-x938-vx33** last week

Package

sp1_prover (Rust)

Affected versions	Patched versions
<6.0.2	6.1.0

sp1_recursion_circuit (Rust)

<6.0.2	6.1.0

sp1_sdk (Rust)

<6.0.2	6.1.0

Description

Summary

A soundness vulnerability in the SP1 V6 recursive shard verifier allows a malicious prover to construct a recursive proof from a shard proof that the native verifier would reject.

- **Affected versions:** `>= 6.0.0, <= 6.0.2`
- **Not affected:** SP1 V5 (all versions)
- **Severity:** High

Details

Background

The recursive shard verifier circuit verifies shard proofs inside a recursive proof. Each shard proof includes a jagged PCS opening, which binds trace-shape metadata into a modified commitment and uses that same shape to evaluate the committed polynomials. These two operations must agree on the committed table heights.

The Bug

In the V6 recursion circuit's jagged verifier, the two checks above are served by separate witnesses: a vector of row counts hashed into the modified commitment (commitment side), and a separate witness of prefix sums derived from row and column counts that drives the jagged polynomial evaluator (evaluation side). The prefix sums are observed within the shard verifier.

The consistency check between these two witnesses was missing in the recursion sub-circuit describing the jagged PCS verifier. A malicious prover can therefore supply one trace shape for commitment binding and a different shape for polynomial evaluation.

Potential Impact

The vulnerability applies to both main trace and preprocessed trace metadata. Because preprocessed traces encode circuit structure (selectors, fixed columns, permutation layout), the potential impact extends beyond data forgery to misrepresentation of the circuit itself.

While we have not demonstrated a full exploit proving arbitrary statements — since modifying one table's layout incidentally constrains changes to related tables — this barrier is not by design and should not be relied upon. We consider this a soundness violation that is unacceptable regardless of current exploitability.

Why the Native Verifier Is Not Affected

The native shard verifier uses a single jagged PCS verifier object where row counts and evaluation layout are derived from the same data, so the split-witness divergence cannot occur. The recursion circuit's shard-level checks (prefix-sum and total-area assertions) only constrain the evaluation-side parameters, not the commitment-side row counts, so they do not catch the gap.

Mitigation

The fix adds a post-evaluation consistency constraint in the recursive jagged verifier. After the jagged evaluation returns the prefix-sum values derived from the evaluation layout, the circuit reconstructs expected prefix sums from the commitment-side row counts (repeating each row count by its corresponding column count and accumulating). It then asserts element-wise equality between the reconstructed and returned prefix sums, and verifies that the final accumulated area matches the total area from the evaluation parameters.

This forces both witnesses to describe the same trace geometry. Any divergence is now a constraint failure.

Credit

This vulnerability was identified through the SP1 bug bounty program on Code4rena.

Severity

High

CVE ID

CVE-2026-40323

Weaknesses

No CWEs