Byte decomposition of pc in AUIPC chip can overflow

High **jonathanpwang** published **GHSA-jf2r-x3j4-23m7** yesterday

Package	Affected versions	Patched versions
openvm (Rust)	1.0.0	None

Severity

High

CVE ID

CVE-2025-46723

Weaknesses

No CWEs

Description

The fix to https://cantina.xyz/code/c486d600-bed0-4fc6-aed1-de759fd29fa2/findings/21 has a typo that still results in the highest limb of `pc` being range checked to 8-bits instead of 6-bits.

In the AIR, we do

openvm/extensions/rv32im/circuit/src/auipc/core.rs
Line 135 in `0f94c8a`

```
135     for (i, limb) in pc_limbs.iter().skip(1).enumerate() {
```

```
for (i, limb) in pc_limbs.iter().skip(1).enumerate() {
    if i == pc_limbs.len() - 1 {
```

It should be

```
for (i, limb) in pc_limbs.iter().enumerate().skip(1) {
```

Right now the if statement is never triggered because the enumeration gives `i=0,1,2` when we instead want `i=1,2,3`. What this means is that `pc_limbs[3]` is range checked to 8-bits instead of 6-bits.

This leads to a vulnerability where the `pc_limbs` decomposition differs from the true `pc`, which means a malicious prover can make the destination register take a different value than the AUIPC instruction dictates, by making the decomposition overflow the BabyBear field.