PR #44744 完整报告

vllm-project/vllm

[Security] Fix remote DoS via invalid recovered token reinjection

合并时间：2026-06-10 17:31

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44744

执行摘要

- 一句话：修复远程 DoS：Triton 采样器越界 token 掩码为 -inf
- 推荐动作：值得精读。展示了 Triton kernel 边界条件下安全加固的典型模式，且测试用例设计技巧（零概率尾部、无草案概率路径覆盖）有参考价值。安全团队和推测解码功能开发者应重点关注。

功能与动机

此修复针对安全公告 GHSA-8wr5-jm2h-8r4f。当 vocab_size 不是 BLOCK_SIZE 的倍数时，最后一个 Triton tile 中的无效位置可能因为平局而在 tl.max 中被选中，产生越界 token ID。这会导致远程拒绝服务，因为无效 token 被注入后续处理。

实现拆解

修复分为三步：

1. Kernel 掩码：在 vllm/v1/sample/rejection_sampler.py 的 sample_received_tokens_kernel 中，score 计算后插入 score = tl.where(vocab_mask, score, float("-inf"))，确保越界位置 score 为负无穷，无法赢得 argmax。
2. Clamping：在写入前添加 recovered_id = tl.minimum(recovered_id, vocab_size - 1) 作为防御性钳制，防止任何逻辑遗漏。
3. 回归测试：在 tests/v1/sample/test_rejection_sampler.py 中新增 test_sample_recovered_tokens_vocab_boundary，参数化覆盖 vocab_size=100、8193、10000、151936，强制尾部 tile 有效概率为零，验证所有恢复的 token ID 均合法。

关键文件：

- vllm/v1/sample/rejection_sampler.py（模块 采样器；类别 source；类型 core-logic）：核心 kernel 修复，添加越界掩码和钳制逻辑
- tests/v1/sample/test_rejection_sampler.py（模块 测试；类别 test；类型 test-coverage；符号 test_sample_recovered_tokens_vocab_boundary）：添加回归测试覆盖多种 vocab_size 边界

关键符号：sample_recovered_tokens_kernel, test_sample_recovered_tokens_vocab_boundary

关键源码片段

vllm/v1/sample/rejection_sampler.py

核心 kernel 修复，添加越界掩码和钳制逻辑

```
    for v in range(0, vocab_size, BLOCK_SIZE):
        vocab_offset = v + tl.arange(0, BLOCK_SIZE)
        vocab_mask = vocab_offset < vocab_size

        # ... load prob and inv_q ...

        score = prob * inv_q
        # Mask out-of-vocabulary entries to -inf so they never win argmax
        # Prevents recovered_id >= vocab_size when all valid entries
        # in the last tail tile have zero probability.
        score = tl.where(vocab_mask, score, float("-inf"))
        local_max, local_id = tl.max(score, axis=0, return_indices=True)

        if local_max > max_val:
            max_val = local_max
            recovered_id = v + local_id

    # Belt-and-suspenders clamp before store
    recovered_id = tl.minimum(recovered_id, vocab_size - 1)
    tl.store(output_token_ids_ptr + token_idx, recovered_id)
```

tests/v1/sample/test_rejection_sampler.py

添加回归测试覆盖多种 vocab_size 边界

```
@pytest.mark.parametrize("no_draft_probs", [True, False])
@pytest.mark.parametrize(
    "vocab_size",
    [
        100, # below BLOCK_SIZE: single partial tile with many padding entries
        8193, # BLOCK_SIZE + 1: only 1 valid entry in the last tile
        10000, # non-aligned, moderate tail
        151936, # real-world Qwen3 vocab size from the CVE report
    ],
)
def test_sample_recovered_tokens_vocab_boundary(vocab_size: int, no_draft_probs: bool):
    """Regression test for GHSA-8wr5-jm2h-8r4f."""
    BLOCK_SIZE = 8192
    batch_size = 2
    max_spec_len = 3
    num_tokens = batch_size * max_spec_len

    last_tile_start = (vocab_size // BLOCK_SIZE) * BLOCK_SIZE

    target_probs = torch.rand(num_tokens, vocab_size, dtype=torch.float32, device=DEVICE_
```

```
TYPE)
if last_tile_start > 0:
    # Zero out valid entries in the last partial tile
    target_probs[:, last_tile_start:] = 0.0
else:
    # For single tile, put all mass at entry 0
    target_probs = torch.zeros_like(target_probs)
    target_probs[:, 0] = 1.0
target_probs = target_probs / target_probs.sum(dim=-1, keepdim=True)

draft_probs = torch.rand(num_tokens, vocab_size, dtype=torch.float32, device=DEVICE_
TYPE)
draft_probs = torch.nn.functional.softmax(draft_probs, dim=-1)

if last_tile_start == 0:
    draft_token_ids = torch.zeros(num_tokens, 1, dtype=torch.int32, device=DEVICE_TYPE)
else:
    draft_token_ids = torch.randint(0, vocab_size, (num_tokens, 1), dtype=torch.int32, device=
    DEVICE_TYPE)

sampling_metadata = create_sampling_metadata(all_greedy=False, temperature=torch.
ones(batch_size, device=DEVICE_TYPE))
spec_decode_metadata = create_spec_decode_metadata(
    draft_token_ids.reshape(batch_size, max_spec_len).tolist(),
    torch.rand(num_tokens, vocab_size, device=DEVICE_TYPE),
)

recovered = sample_recovered_tokens(
    max_spec_len,
    spec_decode_metadata.num_draft_tokens,
    spec_decode_metadata.cu_num_draft_tokens,
    draft_token_ids.squeeze(-1),
    None if no_draft_probs else draft_probs,
    target_probs,
    sampling_metadata,
    device=DEVICE_TYPE,
)

assert (recovered >= 0).all(), f"Negative IDs found: {recovered[recovered < 0].tolist()}"
assert (recovered < vocab_size).all(), f"IDs >= {vocab_size}: {recovered[recovered >= vocab_
size].tolist()}"
```

评论区精华

核心讨论：MatthewBonanni 询问测试是否在未修补内核上失败，以及当 vocab_size < BLOCK_SIZE 时是否同样受影响。作者确认测试会失败，并补充了小 vocab_size 测试用例（100）覆盖单 tile 场景。该讨论已解决，无未解决疑虑。

- 测试在未修补内核上的预期失败 (correctness): 作者确认测试会失败，并已添加 vocab_size=100 用例覆盖单 tile 场景。

风险与影响

- 风险：风险很低。修复仅多了一行 tl.where 和一行 tl.minimum，计算开销可忽略，不影响正常路径。测试覆盖了多种边界组合，包括无草案概率路径。唯一潜在风险是 tl.where 在全词汇表场景下多余判断，但无实际影响。钳制是防御性编程，不会掩盖其他 bug。
- 影响：影响范围：所有使用推测解码的部署，特别是大词汇模型（如 Qwen3、DeepSeek）。由于是远程 DoS 漏洞修复，严重性高，建议尽快合并。用户端无 API 变更，无需配置调整。
- 风险标记：安全修复，核心 kernel 变更，触发条件数学边界

关联脉络

- 暂无明显关联 PR