PR #46463 完整报告

vllm-project/vllm

fix(security): prevent infinite loop in split_audio with NaN audio sa···

合并时间：2026-06-23 18:24

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46463

执行摘要

- 一句话：修复 NaN 音频导致 split_audio 无限循环的安全漏洞
- 推荐动作：建议精读，此 PR 展示了典型的数值安全性 bug 修复模式：IEEE 754 NaN 比较陷阱 + 循环进度保护。值得作为安全编码范例。

功能与动机

修复安全漏洞：全 NaN 音频输入导致 split_audio 永久挂起工作线程，可能被用于拒绝服务攻击。PR body 明确说明了根本原因和修复策略。

实现拆解

1. find_split_point 默认值修复（vllm/multimodal/audio.py）：将 quietest_idx 初始值从 0 改为 start_idx，确保当所有窗口能量为 NaN 时返回合理的起始索引而非 0。
2. NaN 能量跳过（vllm/multimodal/audio.py）：在能量比较条件中增加 not math.isnan(energy) 检查，避免 NaN 污染 min_energy 并导致错误比较。
3. split_audio 进度保护（vllm/multimodal/audio.py）：在调用 find_split_point 后加入守卫逻辑，若返回的 split_point 未前进（<= i），则强制回退到硬分块边界 i + chunk_size，确保循环必然推进。
4. 测试覆盖（tests/multimodal/test_audio.py）：新增 test_find_split_point_nan_input 验证全 NaN 输入 find_split_point 返回 start_idx；新增 test_split_audio_nan_input_terminates 验证全 NaN 音频 split_audio 能正常完成分块且不丢样本。

关键文件：

- vllm/multimodal/audio.py（模块 音频处理；类别 source；类型 core-logic；符号 find_split_point, split_audio）：源代码主修改文件：修复了 find_split_point 的默认值、NaN 跳过和 split_audio 的进度保护逻辑，共 3 处改动，直接修复无限循环 bug。
- tests/multimodal/test_audio.py（模块 音频测试；类别 test；类型 test-coverage；符号 test_find_split_point_nan_input, test_split_audio_nan_input_terminates）：新增测试覆盖：两个新测试验证 NaN 输入下的正确行为，确保修复有效且不会退化。

关键符号：find_split_point, split_audio

关键源码片段

vllm/multimodal/audio.py

源代码主修改文件：修复了 find_split_point 的默认值、NaN 跳过和 split_audio 的进度保护逻辑，共 3 处改动，直接修复无限循环 bug。

```
# vllm/multimodal/audio.py ( 关键变更片段 )

def split_audio(audio_data, sample_rate, max_clip_duration_s,
                overlap_duration_s, min_energy_window_size):
    # ... 前置计算 ...
    while i < audio_data.shape[-1]:
        # ... 最后一块处理 ...
        search_start = i + chunk_size - overlap_size
        search_end = min(i + chunk_size, audio_data.shape[-1])
        split_point = find_split_point(
            audio_data, search_start, search_end, min_energy_window_size
        )

        # 安全保护：确保 split_point 严格大于当前 i，防止无限循环
        # 当音频全为 NaN 时，find_split_point 可能返回 start_idx（即 i），
        # 导致循环不前进。此时回退到硬分块边界。
        if split_point <= i:
            split_point = min(i + chunk_size, audio_data.shape[-1])

        chunks.append(audio_data[..., i:split_point])
        i = split_point
    return chunks


def find_split_point(wav, start_idx, end_idx, min_energy_window):
    segment = wav[start_idx:end_idx]
    min_energy = math.inf
    # 将 quietest_idx 初始化为 start_idx 而非 0，确保全 NaN 时返回安全值
    quietest_idx = start_idx

    for i in range(0, len(segment) - min_energy_window, min_energy_window):
        window = segment[i : i + min_energy_window]
        energy = (window**2).mean() ** 0.5
        # 跳过 NaN 能量值，避免 NaN 比较恒为 False 导致 min_energy 永不更新
        if not math.isnan(energy) and energy < min_energy:
            quietest_idx = i + start_idx
            min_energy = energy

    return quietest_idx
```

tests/multimodal/test_audio.py

新增测试覆盖：两个新测试验证 NaN 输入下的正确行为，确保修复有效且不会退化。

```
# tests/multimodal/test_audio.py ( 新增测试片段 )
```

```
def test_find_split_point_nan_input(self):
    """find_split_point must not return 0 for all-NaN input."""
    from vllm.multimodal.audio import find_split_point

    nan_audio = np.full(32000, float("nan"), dtype=np.float32)
    start_idx = 16000
    end_idx = 32000

    split_idx = find_split_point(
        wav=nan_audio,
        start_idx=start_idx,
        end_idx=end_idx,
        min_energy_window=1600,
    )

    # 必须返回 start_idx（安全回退值），而不是 0
    assert split_idx == start_idx


def test_split_audio_nan_input_terminates(self):
    """split_audio must terminate on all-NaN audio (no infinite loop)."""
    # 31 秒 NaN 音频，16kHz 采样率，超过 max_clip_duration_s=30
    nan_audio = np.full(16000 * 31, float("nan"), dtype=np.float32)

    chunks = split_audio(
        audio_data=nan_audio,
        sample_rate=16000,
        max_clip_duration_s=30.0,
        overlap_duration_s=1.0,
        min_energy_window_size=1600,
    )

    # 必须产生至少 2 个块，且覆盖所有样本
    assert len(chunks) >= 2
    total_samples = sum(c.shape[-1] for c in chunks)
    assert total_samples == nan_audio.shape[-1]
```

评论区精华

PR 无 review 评论，但有审核通过记录。未出现公开争议或设计权衡讨论。

- 暂无高价值评论线程

风险与影响

- 风险：低风险：修复逻辑明确且局部，仅影响全 NaN 或部分 NaN 音频路径。原本的无限循环将导致服务不可用，修复后降级为按硬边界分割（可能产生次优分块点）。由于正常音频能量非零且非 NaN，修复代码不会影响正常路径的性能或行为。

- 影响：影响范围小，仅影响音频预处理模块中两个函数。正常用户无感知；恶意用户通过构造全 NaN 音频发起的 DoS 攻击被消除。对系统稳定性有正面影响。
- 风险标记：数值边界情况，安全漏洞修复

关联脉络

- 暂无明显关联 PR