PR #45510 完整报告

vllm-project/vllm

(security) Enforce audio upload size limit before full file materialization

合并时间：2026-06-15 18:25

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45510

执行摘要

- 一句话：音频上传完全读入内存前提前检查大小限制
- 推荐动作：此 PR 值得所有使用语音 API 的用户关注，建议合并。其设计简洁有效：双重检查（Content-Length + 分块读取）提供了最佳防护；使用已有常量和异常类型保持代码一致性；测试全面覆盖边界和内部行为。对于其他需要上传限制的场景，该模式可直接复用。

功能与动机

语音路由 /v1/audio/transcriptions 和 /v1/audio/translations 在调用 UploadFile.read() 之后才检查文件大小限制，允许超大上传完全读入内存后再拒绝，可能被用于内存耗尽拒绝服务攻击。需要在文件完全读取之前基于 Content-Length 或分块累计进行大小预检。

实现拆解

1. 新增核心函数：在 vllm/entrypoints/speech_to_text/base/utils.py 中实现了 read_upload_with_limit 异步函数。该函数首先检查 file.size（Content-Length 头），若超过限制则立即抛出 VLLMValidationError；然后以 64 KiB 分块循环读取，累计已读字节数，一旦超过限制立即中断并抛出异常，避免完整加载。
2. 修改路由入口：在 vllm/entrypoints/speech_to_text/transcription/api_router.py 和 vllm/entrypoints/speech_to_text/translation/api_router.py 中，将原来的 await request.file.read() 替换为 await read_upload_with_limit(request.file)，并增加对应导入。两处改动完全对称。
3. 补充测试覆盖：新增 tests/entrypoints/speech_to_text/test_upload_size_limit.py，使用 AsyncMock 构造模拟上传文件，测试了 Content-Length 拒绝、分块读拒绝、正常接受、精确边界、超 1 字节拒绝、环境变量默认值以及验证分块读取按预期提前中断等场景，确保逻辑正确。

关键文件：

- tests/entrypoints/speech_to_text/test_upload_size_limit.py（模块 上传限制；类别 test；类型 test-coverage；符号 _make_upload_file, _read, test_rejects_oversized_upload_via_content_length, test_rejects_oversized_upload_via_chunked_read）：新增测试文件，覆盖 Content-Length 拒绝、分块读拒绝、正常接受、边界条件、环境变量默认值以及验证提前中断 read 调用次数，保障核心逻辑正确。

- vllm/entrypoints/speech_to_text/base/utils.py（模块 入口工具；类别 source；类型 dependency-wiring；符号 read_upload_with_limit）：核心实现文件，新增 read_upload_with_limit 函数，实现了双重预检查逻辑（Content-Length 和分块读取），是本次修复的基石。
- vllm/entrypoints/speech_to_text/transcription/api_router.py（模块 转录路由；类别 source；类型 entrypoint）：语音转录路由入口，将原始的 UploadFile.read() 替换为 read_upload_with_limit，引入大小预检查。
- vllm/entrypoints/speech_to_text/translation/api_router.py（模块 翻译路由；类别 source；类型 entrypoint）：语音翻译路由入口，与转录路由完全对称，替换为 read_upload_with_limit。

关键符号：read_upload_with_limit, _make_upload_file, _read

关键源码片段

tests/entrypoints/speech_to_text/test_upload_size_limit.py

新增测试文件，覆盖 Content-Length 拒绝、分块读拒绝、正常接受、边界条件、环境变量默认值以及验证提前中断 read 调用次数，保障核心逻辑正确。

```
# 辅助函数：创建一个模拟的 UploadFile，按分块提供数据
# data: 预置的字节内容，size: 模拟 Content-Length（可为 None）
def _make_upload_file(data: bytes, *, size: int | None = None) -> AsyncMock:
    mock = AsyncMock()
    mock.size = size
    offset = 0

    # 内部模拟异步读取，支持指定读取大小
    async def _read(n: int = -1):
        nonlocal offset
        if n <= 0:
            # 读取剩余所有数据
            chunk = data[offset:]
            offset = len(data)
            return chunk
        # 读取指定大小的一个分块
        chunk = data[offset: offset + n]
        offset += len(chunk)
        return chunk

    mock.read = AsyncMock(side_effect=_read)
    return mock


@pytest.mark.asyncio
async def test_rejects_oversized_upload_via_chunked_read():
    """验证分块读取时，超限文件会被中途拒绝，不会完整加载。"""
    max_mb = 1
    max_bytes = max_mb * 1024 * 1024
```

```
    oversized_data = b"\x00" * (max_bytes + 1024)

    # size=None 表示没有 Content-Length，触发分块检查路径
    upload = _make_upload_file(oversized_data, size=None)

    with pytest.raises(VLLMValidationError, match="Maximum file size exceeded"):
        await read_upload_with_limit(upload, max_size_mb=max_mb)
```

vllm/entrypoints/speech_to_text/base/utils.py

核心实现文件，新增 read_upload_with_limit 函数，实现了双重预检查逻辑（Content-Length 和分块读取），是本次修复的基石。

```
from fastapi import UploadFile
import vllm.envs as envs
from vllm.exceptions import VLLMValidationError
from vllm.utils.mem_constants import KiB_bytes, MiB_bytes

# 每次异步读取的分块大小：64 KiB
_READ_CHUNK_SIZE = 64 * KiB_bytes

async def read_upload_with_limit(
    file: UploadFile,
    max_size_mb: float | None = None,
) -> bytes:
    """有大小限制的 UploadFile 读取函数，在完全加载前拒绝超限文件。"""
    # 如果未指定限制，则从环境变量获取默认值
    if max_size_mb is None:
        max_size_mb = envs.VLLM_MAX_AUDIO_CLIP_FILESIZE_MB

    # 将 MB 限制转换为字节数
    max_bytes = int(max_size_mb * MiB_bytes)

    # 第一道检查：利用 Content-Length（file.size）快速拒绝
    if file.size is not None and file.size > max_bytes:
        raise VLLMValidationError(
            "Maximum file size exceeded",
            parameter="audio_filesize_mb",
            value=file.size / MiB_bytes,
        )

    # 第二道检查：分块读取，累计大小，超限立即停止
    chunks: list[bytes] = []
    total = 0
    while True:
        chunk = await file.read(_READ_CHUNK_SIZE)
        if not chunk: # 读取完毕
            break
        total += len(chunk)
        if total > max_bytes: # 累计超过限制，提前拒绝
```

```
                raise VLLMValidationError(
                    "Maximum file size exceeded",
                    parameter="audio_filesize_mb",
                    value=total / MiB_bytes,
                )
            chunks.append(chunk)

    # 正常返回完整内容
    return b"".join(chunks)
```

评论区精华

评审者 @DarkLight1337 建议在 read_upload_with_limit 中使用已定义的 MiB_bytes 常量替代硬编码的 1024*1024，以保持代码一致性。作者接受并修改。未涉及其他重大争议。

- 使用 MiB_bytes 常量替代硬编码 (style): 作者接受并修改。

风险与影响

- 风险：风险极低。改动仅增加预检查逻辑，不影响正常文件处理。分块读取大小固定 64 KiB，对性能影响可忽略。边界条件（如恰好等于限制）已通过测试覆盖。潜在风险：若环境变量 VLLM_MAX_AUDIO_CLIP_FILESIZE_MB 未设置且未传入参数，函数默认读取该环境变量，需确保 envs 模块有合理默认值（测试中 Mock 了该值，实际运行时若未设置可能导致 max_size_mb 为 None，后续计算 int(None * MiB_bytes) 会报错；但该环境变量通常有默认值，需确认）。
- 影响：正面影响：修复了语音 API 的 DoS 漏洞，防止恶意大文件导致内存溢出。用户侧：超限文件立即收到 VLLMValidationError 而非等待上传完成。系统侧：降低内存峰值，提升稳定性。影响范围仅限于 /v1/audio/transcriptions 和 /v1/audio/translations 两个端点，不涉及其他功能。
- 风险标记：安全边界增强，低风险变更，测试覆盖充分

关联脉络

- 暂无明显关联 PR