PR #47007 完整报告

vllm-project/vllm

fix(security): bound tokenizer work when explicit truncation_side is set

合并时间：2026-06-30 23:08

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47007

执行摘要

- 一句话：修复显式 truncation_side 时 tokenizer 无界 token 化安全漏洞
- 推荐动作：建议阅读，了解 token 化安全设计模式和防御性编程技术。该 PR 的设计权衡（在 tokenizer 层和字符层双重防护）具有参考价值。

功能与动机

当用户指定 truncation_side 和 truncate_prompt_tokens 时， get_encode_kwargs() 返回 truncation=False 且不设置 max_length ，导致 tokenizer 在处理超长输入时不受限制，产生大量无用 token，构成安全风险。详细描述见 PR body。

实现拆解

1. 重构 get_encode_kwargs() 中的注释和逻辑：保留原有流程，但更新注释明确安全防御转移到 _text_len_check。
2. 增强 _text_len_check() 的字符级预截断：当 truncate_prompt_tokens 和 truncation_side 同时设置时，在 token 化之前根据 max_input_chars 对文本进行字符级裁剪，确保 token 化输入始终有界。
3. 简化 _text_len_check() 控制流：合并重复的 max_input_chars 计算，使用 if-elif 结构清晰区分两种情况（无显式截断时报错；有显式截断时静默裁剪）。
4. 更新测试文件，新增四个测试用例覆盖边界场景：无界 token 化被阻止、左侧截断正确性、右侧截断正确性、字符级预裁减生效。

关键文件：

- vllm/renderers/params.py（模块 渲染器配置；类别 source；类型 core-logic；符号 get_encode_kwargs, _text_len_check）：核心逻辑变更，修复安全漏洞，涉及 get_encode_kwargs 和 _text_len_check 方法。
- tests/renderers/test_completions.py（模块 渲染器测试；类别 test；类型 test-coverage；符号 test_explicit_side_tokenizer_unbounded, test_explicit_side_left_text, test_explicit_side_right_text, test_explicit_side_text_pretokenization_guard）：新增 4 个测试用例覆盖安全边界及截断正确性。

关键符号：get_encode_kwargs, _text_len_check

关键源码片段

vllm/renderers/params.py

核心逻辑变更，修复安全漏洞，涉及 get_encode_kwargs 和 _text_len_check 方法。

```
def _text_len_check(self, tokenizer: TokenizerLike | None, text: str) -> str:
    max_input_tokens = self.max_input_tokens
    # 如果未设置最大输入 token 数或 tokenizer 为空，直接返回
    if max_input_tokens is None or tokenizer is None:
        return text

    max_input_chars = max_input_tokens * tokenizer.max_chars_per_token

    if self.truncate_prompt_tokens is None:
        # 当没有显式设置截断 token 数时，如果文本超长则直接拒绝
        if len(text) > max_input_chars:
            raise VLLMValidationError(
                '...' # 详细错误信息省略
            )
    elif self.truncation_side is not None and len(text) > max_input_chars:
        # 深度防御：显式截断场景下，在 token 化之前按字符级裁剪文本
        if self.truncation_side == 'left':
            text = text[-max_input_chars:]
        else:
            text = text[:max_input_chars]

    return text
```

tests/renderers/test_completions.py

新增 4 个测试用例覆盖安全边界及截断正确性。

```
def test_explicit_side_tokenizer_unbounded(self):
    renderer = _build_renderer(MockModelConfig())
    # 使用 500 个字符的输入
    prompts = renderer.render_prompts(
        _preprocess_prompt(renderer.model_config, 'x' * 500)
    )
    results = renderer.tokenize_prompts(
        prompts,
        TokenizeParams(
            max_total_tokens=100,
            truncate_prompt_tokens=4,
            truncation_side='left',
        ),
    )
    # 输出应仅为 4 个 token，而非 500
    assert len(results) == 1
    assert len(results[0]['prompt_token_ids']) == 4
    # tokenizer 必须收到 truncation=False（保留完整序列用于后续切片）
    kwargs = renderer.tokenizer._captured_encode_kwargs
    assert kwargs['truncation'] is False
```

评论区精华

- DarkLight1337 建议使用局部变量简化逻辑，作者采纳。
- DarkLight1337 指出 _text_len_check 中存在重复检查，建议改为嵌套 if-else，作者修改。
- depthfirst-app[bot] 指出 get_encode_kwargs 中存在死代码分支（max_length is None 在 truncate_prompt_tokens 非 None 时不可达），但该问题不在本 PR 范围内，未作修改。
- 使用局部变量简化逻辑 (design): 已采纳，代码得到简化。
- 消除重复检查 (design): 已采纳，代码结构更清晰。
- 死代码分支分析 (correctness): 讨论提及但未修改，可能留待后续改进。

风险与影响

- 风险：字符级预截断可能改变截断语义（严格按字符数而非 token 数），但仅在 truncation_side 被显式设置时生效，且与后续 token 化后截断协同，风险可控。_text_len_check 新增的 tokenizer is None 提前返回，可能使某些路径绕过字符检查，但 tokenizer 为 None 时通常不需要 token 化，影响有限。测试覆盖了主要场景，但未涵盖 truncate_prompt_tokens 为负值的情况（当为负值时走不同分支）。
- 影响：对用户：修复了潜在的安全问题，透明；对于正常使用，行为不变，因为 explicit-side 分支本就需要完整序列，预截断仅影响超长输入。对系统：减少了因恶意超长输入导致的 CPU/memory 消耗，提升稳定性。对团队：逻辑清晰化，易于维护。
- 风险标记：核心路径变更，字符级截断，新增测试覆盖

关联脉络

- 暂无明显关联 PR