PR #44131 完整报告

vllm-project/vllm

[CI] Stabilize OpenAI schema fuzzing for malformed structural tags

合并时间：2026-06-02 10:56

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44131

执行摘要

- 一句话：在 API 层提前校验 structural_tag 格式
- 推荐动作：推荐阅读，特别是 validate_structural_tag_response_format 的实现，展示了一种将深层引擎错误转化为 API 层校验错误的模式，有助于保持 API 的错误分类清晰。

功能与动机

根据 PR 描述，Schemathesis 生成的请求 {"response_format": {"type": "structural_tag", "format": null}} 导致服务器返回 500，而应当拒绝返回 400。此举也消除了 CI fuzzing 测试的不稳定性。

实现拆解

1. 在 vllm/entrypoints/openai/engine/protocol.py 中新增 validate_structural_tag_response_format、validate_structural_tag_payload、validate_structured_outputs_structural_tag 三个函数。其中 validate_structural_tag_response_format 负责将 dict 类型的 response_format 转换为 Pydantic 模型，然后调用 validate_structural_tag_payload 进行 xgrammar 检查。validate_structured_outputs_structural_tag 解构 structured_outputs 中的 structural_tag 字段，调用相同 payload 验证。
2. 在 ChatCompletionRequest 的 validate_response_format 校验器中插入代码：当 rf_type == 'structural_tag' 时调用 validate_structural_tag_response_format。同时在其 check_structured_outputs_count 和 BatchChatCompletionRequest.check_batch_mode 中增加对应的 validate_structured_outputs_structural_tag 调用。
3. 在 CompletionRequest 的对应校验器中进行完全相同的修改，确保 completion 端点也受保护。
4. 为以上三个路径编写针对性测试：新增 test_structural_tag_response_format_invalid 和 test_batch_structural_tag_response_format_invalid 覆盖 chat completion 和 batch，test_structured_outputs_structural_tag_invalid 覆盖直接 structured_outputs 发送的方式。completion 端点测试同理。
5. 调整 tests/tool_parsers/test_mistral_tool_parser.py 中的测试 fixture，将其原先使用的部分有效（但存在歧义）的 structural_tag 格式改为明确的合法结构，以免被新验证拦截。

关键文件：

- vllm/entrypoints/openai/engine/protocol.py（模块 协议层；类别 source；类型 dependency-wiring；符号 validate_structural_tag_response_format, validate_structural_tag_payload, validate_structured_outputs_structural_tag）：核心验证函数所在文件，新增三个验证函数，是整个 PR 的基石
- vllm/entrypoints/openai/chat_completion/protocol.py（模块 协议层；类别 source；类型 core-logic）：ChatCompletionRequest 调用验证函数，覆盖主要入口
- tests/entrypoints/openai/chat_completion/test_chat_error.py（模块 聊天错误测试；类别 test；类型 test-coverage；符号 test_structural_tag_response_format_invalid, test_batch_structural_tag_response_format_invalid, test_structured_outputs_structural_tag_invalid）：为聊天完成请求添加三种 structural_tag 拒绝测试
- vllm/entrypoints/openai/completion/protocol.py（模块 协议层；类别 source；类型 core-logic）：CompletionRequest 调用验证函数，覆盖完成请求路径
- tests/entrypoints/openai/completion/test_completion_error.py（模块 完成错误测试；类别 test；类型 test-coverage；符号 test_structural_tag_response_format_invalid, test_structured_outputs_structural_tag_invalid）：为完成请求添加 structural_tag 拒绝测试
- tests/tool_parsers/test_mistral_tool_parser.py（模块 工具解析；类别 test；类型 test-coverage）：调整测试数据以通过新的验证

关键符号：validate_structural_tag_response_format, validate_structural_tag_payload, validate_structured_outputs_structural_tag

关键源码片段

vllm/entrypoints/openai/engine/protocol.py

核心验证函数所在文件，新增三个验证函数，是整个 PR 的基石

```
def validate_structural_tag_response_format(
    response_format: AnyStructuralTagResponseFormat | dict[str, Any],
) -> None:
    """Validate structural tags before they are sent to the engine.

    Engine-side validation reports malformed structural tags as generation
    failures. OpenAI request parsing should classify them as bad requests.
    """
    import json

    from pydantic import TypeAdapter, ValidationError

    # 如果 response_format 是 dict，先转换为 Pydantic 模型以获得字段校验
    if isinstance(response_format, dict):
        try:
            response_format = TypeAdapter(
                AnyStructuralTagResponseFormat
            ).validate_python(response_format)
```

```
        except ValidationError as exc:
            raise VLLMValidationError(
                "Invalid response_format structural_tag specification.",
                parameter="response_format",
            ) from exc

    try:
        # 序列化后交给下层 payload 验证
        payload = json.dumps(response_format.model_dump(by_alias=True))
        validate_structural_tag_payload(payload, parameter="response_format")
    except (TypeError, ValueError) as exc:
        raise VLLMValidationError(
            "Invalid response_format structural_tag specification.",
            parameter="response_format",
        ) from exc


def validate_structural_tag_payload(payload: Any, *, parameter: str) -> None:
    from vllm.sampling_params import SamplingParams, StructuredOutputsParams
    from vllm.v1.structured_output.backend_xgrammar import validate_xgrammar_grammar

    # 空字符串直接拒绝，避免下层引擎当作缺失处理
    if isinstance(payload, str) and not payload:
        raise VLLMValidationError(
            f"Invalid {parameter} structural_tag specification.",
            parameter=parameter,
        )

    try:
        # 复用 xgrammar 的 grammar 验证逻辑
        validate_xgrammar_grammar(
            SamplingParams(
                structured_outputs=StructuredOutputsParams(structural_tag=payload)
            )
        )
    except (TypeError, ValueError) as exc:
        raise VLLMValidationError(
            f"Invalid {parameter} structural_tag specification.",
            parameter=parameter,
        ) from exc
```

tests/entrypoints/openai/chat_completion/test_chat_error.py

为聊天完成请求添加三种 structural_tag 拒绝测试

```
@pytest.mark.parametrize("format_value", [None, {}])
def test_structural_tag_response_format_invalid(format_value):
    # 验证当 response_format 的 format 字段为 None 或空 dict 时，
    # 请求构建会抛出 ValidationError，匹配提示信息
    with pytest.raises(
```

```
        ValidationError,
        match="Invalid response_format structural_tag",
    ):
        ChatCompletionRequest(
            model=MODEL_NAME,
            messages=[{"role": "user", "content": "hello"}],
            response_format={"type": "structural_tag", "format": format_value},
        )
```

评论区精华

review 中仅有一轮讨论：DarkLight1337 提问是否应该使用 VLLMValidationError 进行测试，作者回应直接用 ValidationError 更直接，因为测试触发的是 Pydantic 校验器而非手动调用函数。最终 reviewer 认可并通过。

- 测试中应使用 ValidationError 还是 VLLMValidationError (design): 决定保留 ValidationError，reviewer 认可并批准。

风险与影响

- 风险：新加的 validate_structural_tag_payload 内部调用 validate_xgrammar_grammar，如果 xgrammar 版本更迭导致验证规则变化，可能影响有效请求。但由于该函数本就在引擎内部使用，风险保持一致。另外，空字符串被提前拒绝，可能影响一些边缘用例，但空 structural_tag 本无意义，所以影响很小。
- 影响：对用户：无效 structural_tag 请求得到明确的 400 错误响应而非 500；对系统：无额外性能开销；对团队：新增的验证函数可被其他需要校验 structural_tag 的路径复用，降低重复代码。
- 风险标记：依赖 xgrammar 验证行为，新增验证路径可能误拒有效请求，涉及三个 API 路径（chat、completion、batch）

关联脉络

- 暂无明显关联 PR