PR #42594 完整报告

vllm-project/vllm

fix: add API key authorization to /v2 endpoints

合并时间：2026-05-16 09:29

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42594

执行摘要

- 一句话：修复 /v2 端点 API key 认证绕过漏洞
- 推荐动作：值得立即合并。作为一个安全修复，变更简洁且测试完备。设计上采用元组常量管理受保护前缀的做法值得推广。建议后续跟进路径规范化以消除评论中提出的边缘情况。

功能与动机

Issue #42591 报告了 /v2/embed 端点可绕过 --api-key 认证的安全漏洞。AuthenticationMiddleware 仅拦截 /v1 路径，导致 /v2 端点未受保护。此变更旨在弥补这一缺失，防止未授权用户调用敏感端点。

实现拆解

1. 在 vllm/entrypoints/openai/server_utils.py 中，于 logger 初始化后新增模块级常量 GUARDED_PREFIX = ("/v1", "/v2", "/inference")，集中管理需要认证的路径前缀。
2. 修改 AuthenticationMiddleware.__call__ 中的条件判断，将 url_path.startswith("/v1") 替换为 url_path.startswith(GUARDED_PREFIX)，确保 /v2 和 /inference 路径也触发 token 验证。
3. 同步更新类的文档注释，将“跳过条件”从“不以 /v1 开头”改为“不以 GUARDED_PREFIX 开头”，保持文档准确。
4. 在 tests/entrypoints/serve/instrumentator/test_optional_middleware.py 中，将原有测试 test_not_v1_api_token 重命名为 test_not_v1_or_v2_path_skips_auth，以清晰表述新行为；并新增两个测试用例：test_v2_endpoint_rejects_missing_api_token 和 test_v2_endpoint_accepts_valid_api_token，分别验证 /v2/embed 在无 token 时返回 401、有有效 token 时返回 200。
5. 在 docs/usage/security.md 中更新受保护端点列表：将 /inference/v1/generate 从“不受保护”移至“受保护”列表，并新增 /v2/embed 和 /v2/rerank；同时修改概述段落，明确提及 /v2 和 /inference 前缀也受保护。

关键文件：

- vllm/entrypoints/openai/server_utils.py（模块 入口服务；类别 source；类型 core-logic；符号 GUARDED_PREFIX, AuthenticationMiddleware.call）：核心修复文件：定义了 GUARDED_PREFIX 常量，修改了 AuthenticationMiddleware.__call__ 中的路径前缀检查逻辑，是安全漏洞的直接修复点。

- tests/entrypoints/serve/instrumentator/test_optional_middleware.py（模块 中间件测试；类别 test；类型 test-coverage；符号 test_not_v1_or_v2_path_skips_auth, test_v2_endpoint_rejects_missing_api_token, test_v2_endpoint_accepts_valid_api_token）：测试覆盖文件：验证新行为，包括重命名现有测试和新增两个针对 /v2/embed 端点的认证测试用例，确保修复正确且无回归。
- docs/usage/security.md（模块 文档；类别 docs；类型 documentation）：文档更新：同步反映认证覆盖范围的变化，将 /v2 和 /inference 端点从不受保护列表移至受保护列表，避免用户误解。

关键符号：AuthenticationMiddleware.call, verify_token

关键源码片段

vllm/entrypoints/openai/server_utils.py

核心修复文件：定义了 GUARDED_PREFIX 常量，修改了 AuthenticationMiddleware.__call__ 中的路径前缀检查逻辑，是安全漏洞的直接修复点。

```
# vllm/entrypoints/openai/server_utils.py (partial)

logger = init_logger("vllm.entrypoints.openai.server_utils")

# 集中管理需要 API Key 认证的路径前缀
# 所有以此元组中任一字符串开头的 HTTP 请求都将通过 verify_token 检查
GUARDED_PREFIX = ("/v1", "/v2", "/inference")


class AuthenticationMiddleware:
    """
    Pure ASGI middleware 对每个请求进行 Bearer token 认证。

    两种跳过认证的情况：
        1. HTTP method 为 OPTIONS。
        2. 请求路径不以 GUARDED_PREFIX 中的任何一项开头（如 /health）。
    """

    def __init__(self, app: ASGIApp, tokens: list[str]) -> None:
        self.app = app
        # 存储 SHA-256 哈希后的 token，避免明文存储
        self.api_tokens = [hashlib.sha256(t.encode("utf-8")).digest() for t in tokens]

    def verify_token(self, headers: Headers) -> bool:
        """验证 Authorization header 中的 Bearer token 是否匹配。"""
        authorization_header_value = headers.get("Authorization")
        if not authorization_header_value:
            return False
        scheme, _, param = authorization_header_value.partition(" ")
        if scheme.lower() != "bearer":
            return False
```

```
            param_hash = hashlib.sha256(param.encode("utf-8")).digest()
            # 使用 secrets.compare_digest 进行常量时间比较，防止时序攻击
            token_match = False
            for token_hash in self.api_tokens:
                token_match |= secrets.compare_digest(param_hash, token_hash)
            return token_match

        def __call__(self, scope: Scope, receive: Receive, send: Send) -> Awaitable[None]:
            if (
                scope["type"] not in ("http", "websocket")
                or scope.get("method") == "OPTIONS"
            ):
                return self.app(scope, receive, send)
            root_path = scope.get("root_path", "")
            url_path = URL(scope=scope).path.removeprefix(root_path)
            headers = Headers(scope=scope)
            # 如果路径以 GUARDED_PREFIX 开头且 token 验证失败，返回 401
            if url_path.startswith(GUARDED_PREFIX) and not self.verify_token(headers):
                response = JSONResponse(content={"error": "Unauthorized"}, status_code=401)
                return response(scope, receive, send)
            return self.app(scope, receive, send)
```

tests/entrypoints/serve/instrumentator/test_optional_middleware.py

测试覆盖文件：验证新行为，包括重命名现有测试和新增两个针对 /v2/embed 端点的认证测试用例，确保修复正确且无回归。

```
# tests/entrypoints/serve/instrumentator/test_optional_middleware.py (partial)

# 已有测试用例：test_missing_api_token, test_passed_api_token...

# -----------------------------------------------------------------------
# /v2 path authentication tests
# -----------------------------------------------------------------------


@pytest.mark.parametrize(
    "server",
    [["--api-key", "test"]],
    indirect=True,
)
@pytest.mark.asyncio
async def test_v2_endpoint_rejects_missing_api_token(server: RemoteOpenAIServer):
    # 不带 token 请求 /v2/embed 应返回 401
    body = {
        "model": MODEL_NAME,
        "texts": ["hello"],
        "embedding_types": ["float"],
    }
    response = requests.post(server.url_for("/v2/embed"), json=body)
```

```
    assert response.status_code == HTTPStatus.UNAUTHORIZED


@pytest.mark.parametrize(
    "server",
    [["--api-key", "test"]],
    indirect=True,
)
@pytest.mark.asyncio
async def test_v2_endpoint_accepts_valid_api_token(server: RemoteOpenAIServer):
    # 携带有效 Bearer token 请求 /v2/embed 应返回 200
    body = {
        "model": MODEL_NAME,
        "texts": ["hello"],
        "embedding_types": ["float"],
    }
    response = requests.post(
        server.url_for("/v2/embed"),
        json=body,
        headers={"Authorization": "Bearer test"},
    )
    assert response.status_code == HTTPStatus.OK


# 原来的 test_not_v1_api_token 已重命名为 test_not_v1_or_v2_path_skips_auth
# 并更新注释，确保 /health 等非受保护路径仍然跳过认证
@pytest.mark.parametrize(
    "server",
    [["--api-key", "test"]],
    indirect=True,
)
@pytest.mark.asyncio
async def test_not_v1_or_v2_path_skips_auth(server: RemoteOpenAIServer):
    # 不以 /v1, /v2, /inference 开头的路径应跳过认证
    response = requests.get(server.url_for("health"))
    assert response.status_code == HTTPStatus.OK
```

评论区精华

只有一次来自 gemini-code-assist[bot] 的评论，指出了路径规范化风险：如果 root_path 尾随斜杠或路径包含多个前导斜杠（如 //v1/chat），removeprefix 可能产生不带前导斜线的路径，从而绕过 startswith 检查。评论建议使用 lstrip("/") 增强健壮性。该建议在最终提交中未被采纳，但现有代码也不处理此情况，因此风险非本 PR 引入。

- 路径规范化风险：root_path 与多重前导斜杠可能绕过认证 (security)：未被采纳为变更；现有代码也存在同样问题，不在本 PR 修复范围内。不影响当前安全修复的有效性。

风险与影响

- 风险：低风险。核心变更仅 7 行，逻辑清晰，测试覆盖了正反场景。但存在路径规范化隐患（如 //v2/embed 或 root_path 配置导致路径变形），可能绕过认证——此问题在修复前同样存在，并非本 PR 引入。另外 /inference 端点从无认证变更为需认证，可能影响依赖无令牌访问的已有脚本或集成，但这是安全修复的预期行为。
- 影响：对用户：任何使用 --api-key 启动服务器并暴露 /v2/embed、/v2/rerank、/inference/v1/generate 端点的用户，现在都必须提供有效 API key 才能访问这些端点。这是一个 breaking change（对依赖无认证访问的客户端），但属于安全修复的必要措施。对系统：减少了未授权调用风险。对团队：代码维护性略有提升（通过 GUARDED_PREFIX 常量集中管理受保护路径）。
- 风险标记：缺少路径规范化，影响已配置 --api-key 的用户，breaking change（依赖无认证访问的用户）

关联脉络

- PR #42591 [Bug]: Security: /v2/embed endpoint bypasses API Key authentication in v0.20.2: 直接关联的问题报告，描述了 /v2/embed 绕过认证的漏洞，是本 PR 的动机来源。
- PR #29922：无需提供 title；评论中 noooop 提到 'more background details:
  - 29922'，该 PR 可能提供了更早的认证相关上下文，但未提供标题。