

PR #56662 完整报告

vllm-project/vllm

[Bugfix] Redact credentials from benchmark logs

合并时间: 2026-09-15 06:58

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56662>

执行摘要

- 一句话: Bench 日志与 PyTorch 元数据脱敏, 防止凭据外泄
- 推荐动作: 值得精读, 改动虽小但展示了日志脱敏的通用套路: 共享纯函数 `redact_sensitive_namespace` 加调用点替换, 并刻意把脱敏副本与请求数据路径分离, 避免在打印与写入 artifact 的过程中泄露凭据。值得关注的决策: 一是用浅拷贝而不是原地修改, 保证原参数零副作用; 二是 `save_to_pytorch_benchmark_format` 也传入脱敏副本, 防止凭据从 stdout 转移到结果文件。建议后续在 `tests/benchmarks/test_serve_cli.py` 中补充正式断言, 把 stdout 不含明文 header / api_key、结果文件不含明文、原始 args 不变这三点固化下来, 并考虑基于参数命名约定自动识别敏感字段。

功能与动机

PR body 明确指出: `vllm bench serve` printed its full argument namespace, including values passed through `--header`, while the multi-turn benchmark logged both `--api-key` and `--header`. This could expose credentials in local or CI logs。也就是说, 用户通过 `--header` 传入的 `Authorization: Bearer ...` 或通过 `--api-key` 传入的密钥, 会随启动诊断日志出现在终端、日志文件与 CI 产物中。修复目标是只对日志与元数据输出做脱敏, 同时保留原始参数用于真实 HTTP 请求。

实现拆解

1. 新增共享脱敏工具: 在 `vllm/benchmarks/lib/utils.py` 中新增 `redact_sensitive_namespace(args, fields)`。它通过 `argparse.Namespace(**vars(args))` 浅拷贝整个参数 Namespace, 再对 `fields` 中值为非 `None` 的字段统一赋值为 `'***'`, 最后返回副本。这样任何调用方都可以在打印或序列化前获得一份脱敏数据, 而原始 `args` 完全不变, 请求发送路径不受影响。
2. `serve.py` 双路径接入: 在 `vllm/benchmarks/serve.py` 中引入该工具并定义 `_SENSITIVE_ARG_FIELDS = ('header',)`。`main_async` 启动时的 `print(args)` 改为 `print(redact_sensitive_namespace(args, _SENSITIVE_ARG_FIELDS))`; 同时 `save_to_pytorch_benchmark_format` 调用 `convert_to_pytorch_benchmark_format` 时, `args` 参数也换成脱敏副本, 避免 `--header` 中的 `Authorization` 头以元数据形式写入 `pytorch.json` 结果文件。
3. 多轮 benchmark 接入: 在 `benchmarks/multi_turn/benchmark_serving_multi_turn.py` 中同样导入该工具, 定义 `_SENSITIVE_ARG_FIELDS = ('api_key', 'header')`, 并将

logger.info(args) 改为打印脱敏副本。这样 --api-key 与 --header 不再出现在启动日志里。

4. 测试与配套情况：本次没有新增或修改测试文件；PR body 中描述了本地验证方式——用 python -c 断言 stdout、PyTorch 元数据、多轮日志均脱敏且原始 Namespace 值保持不变，并执行 python -m pytest tests/benchmarks/test_serve_cli.py 与 pre-commit 检查。这意味着脱敏行为目前缺乏仓库内的自动化回归测试。

关键文件：

- vllm/benchmarks/lib/utils.py（模块 脱敏工具；类别 source；类型 core-logic；符号 redact_sensitive_namespace）：新增核心脱敏工具 redact_sensitive_namespace，serve 与 multi_turn 两个入口均复用它，是整个修复的公共支点。
- vllm/benchmarks/serve.py（模块 基准脚本；类别 source；类型 dependency-wiring；符号 main_async, save_to_pytorch_benchmark_format）：serve 启动日志与 PyTorch 元数据写入路径都使用脱敏副本，覆盖 --header 中的 Authorization 头，是泄露面最大的调用点。
- benchmarks/multi_turn/benchmark_serving_multi_turn.py（模块 多轮脚本；类别 source；类型 dependency-wiring；符号 main）：多轮 benchmark 的启动日志同时覆盖 --api-key 与 --header 两个敏感参数，是第二处泄露入口。

关键符号：redact_sensitive_namespace

关键源码片段

vllm/benchmarks/lib/utils.py

新增核心脱敏工具 redact_sensitive_namespace，serve 与 multi_turn 两个入口均复用它，是整个修复的公共支点。

```
# vllm/benchmarks/lib/utils.py
# 核心改动：共享的脱敏工具，供 serve 与 multi_turn 两个入口复用。
```

```
import argparse
```

```
def redact_sensitive_namespace(
    args: argparse.Namespace, fields: tuple[str, ...]
) -> argparse.Namespace:
    # 用 vars(args) 浅拷贝一份全新的 Namespace，原始对象保持不变
    redacted_args = argparse.Namespace(**vars(args))
    for field in fields:
        # 只替换非 None 值，避免把未启用的可选参数也打印成 '****'
        if getattr(redacted_args, field, None) is not None:
            setattr(redacted_args, field, '****')
    return redacted_args
```

vllm/benchmarks/serve.py

serve 启动日志与 PyTorch 元数据写入路径都使用脱敏副本，覆盖 --header 中的 Authorization 头，是泄露面最大的调用点。

```

# vllm/benchmarks/serve.py
# 敏感字段清单只包含 header, Authorization 头主要由它携带。
_SENSITIVE_ARG_FIELDS = ('header',)

def save_to_pytorch_benchmark_format(
    args: argparse.Namespace, results: dict[str, Any], file_name: str
) -> None:
    # 输出到 PyTorch 元数据的指标集合
    metrics = [
        'median_ttft_ms', 'mean_ttft_ms', 'std_ttft_ms', 'p99_ttft_ms',
        'mean_tpot_ms', 'median_tpot_ms', 'std_tpot_ms', 'p99_tpot_ms',
        'median_itl_ms', 'mean_itl_ms', 'std_itl_ms', 'p99_itl_ms',
    ]
    # 原始输出序列体积较大, 暂不进元数据
    ignored_metrics = ['ttfts', 'itls', 'generated_texts', 'errors']
    # 关键变更: 传入脱敏副本, 避免 header 中的凭据写入 .pytorch.json
    pt_records = convert_to_pytorch_benchmark_format(
        args=redact_sensitive_namespace(args, _SENSITIVE_ARG_FIELDS),
        metrics={k: [results[k]] for k in metrics if k in results},
        extra_info={
            k: results[k]
            for k in results
            if k not in metrics and k not in ignored_metrics
        },
    )
    if pt_records:
        # 故意不使用 json 后缀, 防止 CI 误采集该文件
        pt_file = f'{os.path.splitext(file_name)[0]}.pytorch.json'
        write_to_json(pt_file, pt_records)

```

benchmarks/multi_turn/benchmark_serving_multi_turn.py

多轮 benchmark 的启动日志同时覆盖 --api-key 与 --header 两个敏感参数, 是第二处泄露入口。

```

# benchmarks/multi_turn/benchmark_serving_multi_turn.py
# 多轮 benchmark 的敏感字段同时覆盖 api_key 与 header。
_SENSITIVE_ARG_FIELDS = ('api_key', 'header')

async def main() -> None:
    args = parser.parse_args()
    # 原代码是 logger.info(args), 会把裸的 API Key 打印到日志中
    logger.info(redact_sensitive_namespace(args, _SENSITIVE_ARG_FIELDS))
    # 后续读取 url、model 等字段仍使用原始 args, 不受脱敏影响
    logger.info(f'url={args.url}')
    logger.info(f'model={args.model}')

```

评论区精华

本 PR 没有产生实质性 review 讨论：GitHub 上没有 inline review 评论；claude[bot] 自动提示 fork PR 默认关闭自动 review，维护者 yewentao256 直接批准并给出 LGTM, thanks for the work!。可留意的点在于 PR body 用复现命令与 main/branch 输出对比证明了脱敏效果，但这些断言没有被固化为仓库测试。

- 暂无高价值评论线程

风险与影响

- 风险：redact_sensitive_namespace 依赖调用方手工维护敏感字段清单：serve.py 只覆盖 header，多轮脚本覆盖 api_key 与 header。未来新增 --token、--password、带明文凭据的 URL 等参数时，若忘记扩展 _SENSITIVE_ARG_FIELDS，日志仍会泄露凭据。本次没有新增测试文件，脱敏行为只存在于 PR body 的本地断言中，仓库内的回归保护不足。另外该函数是浅拷贝，如果敏感值类型变成嵌套对象或可变容器，副本与原始对象可能共享内部状态，需要重新评估。多轮脚本新增了对 vllm.benchmarks.lib.utils 的导入，脚本被单独拷贝到无 vLLM 包的环境时会出现 ImportError，但仓库内运行不受影响。
- 影响：对用户：运行 vllm bench serve 与多轮 benchmark 的用户在 stdout / 日志中不再看到 --api-key 与 --header 的明文值；--save-result 等生成的 .pytorch.json 元数据文件同样脱敏。对系统：请求路径使用的仍是原始 args，吞吐、时延与请求内容没有任何变化。对团队：CI 日志安全等级提升；但由于敏感字段清单是硬编码的，后续新增相似入口时需要同步接入，否则会出现覆盖盲区。
- 风险标记：缺少测试文件变更，敏感字段清单需人工维护，浅拷贝边界需留意

关联脉络

- PR #51104 [Rust][Benchmark] Support HF ShareGPT datasets in multi-turn mode: 同属 benchmark 工具链的演进，但位于 Rust 侧，与本 PR 没有共享文件，关联较间接。
- PR #55650 [Bugfix] Make KV cache and MFU log lines backend-neutral: 同属日志输出治理类改动，但作用于 metrics 日志路径，与本 PR 不在同一模块。