PR #34892 完整报告

sgl-project/sglang

feat: add safeguards for remote media URLs

合并时间：2026-08-15 18:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34892

执行摘要

- 一句话：新增远程媒体 URL 域名白名单与下载大小限制，防 SSRF
- 推荐动作：值得精读。核心看点： 1) download_remote_media 中先用 requests.Request(...).prepare().url 统一 URL 规范化再校验，避免解析器分歧绕过； 2) 手动逐跳重定向校验 + 流式字节上限 + 全局 deadline 的三重防护设计； 3) 在 dummy-model 边界之前通过 server args 发布进程级策略的接线方式。同时注意仓库正处于 server_args → config bag 迁移期，本 PR 在 base_processor/encode_server 新增的直读点应在后续 step 迁移中纳入 bags。

功能与动机

PR body 明确提出：Allow deployments to restrict client-supplied remote media URLs and bound download size，并强调 The domain allowlist remains opt-in for backward compatibility. Remote downloads default to a 64 MiB limit。SGLang 多模态入口接受客户端直接提供的 HTTP(S) 媒体 URL，由服务端进程下载后解码；此前既无域名限制也无大小上限，攻击者可以诱导服务端访问内网地址或云元数据端点（测试中明确覆盖了 http://169.254.169.254/latest/meta-data），构成 SSRF，也可以提交超大媒体耗尽内存与带宽。该 PR 以默认兼容的方式补齐这两条防线。

实现拆解

1. 新增安全下载器核心（python/sglang/srt/utils/common.py）：新增 _normalize_media_domain（支持 IP 归一化、IDNA 域名编码、去尾点，拒绝端口 / 路径 /URL 片段）、configure_media_url_security（进程级策略发布，返回规范化域名列表）、_assert_media_url_allowed（域名白名单校验）、download_remote_media（手动跟随重定向、逐跳校验目的地、流式读取并强制字节上限与全局 deadline）。
2. 新增 server 参数与启动接线（python/sglang/srt/server_args.py）：新增 allowed_media_domains（List[str]，默认空 = 不限制）与 media_url_max_file_size_mb（int，默认 64，0 表示禁用）；__post_init__ 在 dummy-model 边界之前调用 _handle_media_url_security 完成规范化并发布策略，确保所有 worker 线程共享同一配置。
3. 统一 loader 接入：公共加载器 load_audio、get_image_bytes、_normalize_video_input 从直连 get_mm_http_session() 切换到 download_remote_media；模型特定处理器 mimo_audio.preprocess_audio（保留慢速下载告警）、mimo_v2.has_audio_track（放弃 ffprobe HTTP Range 探测，避免绕过策略）、mimo_v2.fetch_image、

moss_vl._normalize_video_string（从流式写文件改为整块写入临时文件）、inkling._resolve_media_item（urllib 替换为共享下载器）全部接入。

4. 多进程 / 分离部署发布：base_processor.BaseMultimodalProcessor.__init__ 与 disaggregation/encode_server.py 的 MMEncoder 初始化时调用 configure_media_url_security，保证 DP/ 独立编码进程也遵守同一策略。
5. 测试与文档配套：新增 test_media_url_security.py（本地 ThreadingHTTPServer 模拟同域 / 跨域重定向、声明长度与 chunked 超限、重定向循环、非法白名单条目、backslash userinfo 绕过、三个加载器共享策略）；扩展 test_server_args_migration.py（参数归一化如 Media.Example.com. → media.example.com、拒绝非法值并重置全局状态）；test_mm_process_config.py、test_kimi_k25.py、rust/qwen _fixtures.py 补充策略重置；docs 记录新参数语义。

关键文件：

- python/sglang/srt/utils/common.py（模块 工具层；类别 source；类型 core-logic；符号 _normalize_media_domain, configure_media_url_security, _assert_media_url_allowed, download_remote_media）：安全策略核心实现：新增域名白名单、重定向校验、流式下载大小限制的统一下载器，所有媒体加载路径的最终汇聚点。
- test/registered/unit/multimodal/test_media_url_security.py（模块 安全测试；类别 test；类型 test-coverage；符号 _MediaHandler, do_GET, log_message, TestMediaURLSecurity）：新增的安全回归测试文件，用本地 HTTP 服务器覆盖白名单、重定向校验、大小限制、绕过攻击等全部核心场景，是验证安全策略的关键配套。
- python/sglang/srt/server_args.py（模块 参数配置；类别 source；类型 core-logic；符号 _handle_media_url_security）：新增两个安全相关 server 参数并在启动早期发布策略，是部署方启用 / 配置安全能力的入口。
- test/registered/unit/test_server_args_migration.py（模块 参数测试；类别 test；类型 test-coverage；符号 test_media_url_security_args, test_media_url_security_args_reject_invalid_values）：验证新参数通过 A[T, Arg] 注解自动生成 CLI 参数、域名归一化（IDNA/ 去尾点 /IP）与非法值拒绝。
- python/sglang/srt/multimodal/processors/mimo_audio.py（模块 音频处理；类别 source；类型 core-logic；符号 preprocess_audio）：模型特定音频下载路径接入共享安全下载器，保留慢速下载告警逻辑。
- python/sglang/srt/multimodal/processors/mimo_v2.py（模块 多模态处理；类别 source；类型 dependency-wiring；符号 has_audio_track, fetch_image）：has_audio_track 放弃 ffprobe HTTP Range 探测（避免绕过策略），fetch_image 改用共享下载器，是消除旁路路径的关键改动。
- python/sglang/srt/multimodal/processors/base_processor.py（模块 处理器基类；类别 source；类型 core-logic）：多模态处理器初始化时发布媒体 URL 安全策略，保证 processor worker 线程共享同一策略。
- python/sglang/srt/disaggregation/encode_server.py（模块 编码服务；类别 source；类型 core-logic）：独立 MMEncoder 进程初始化时发布媒体 URL 安全策略，覆盖 PD 分离部署场景。

- docs/docs/advanced_features/server_arguments.mdx（模块 参数文档；类别 docs；类型 documentation）：记录新参数的语义、默认值与禁用方式，是部署方启用安全能力的文档入口。

关键符号：download_remote_media, configure_media_url_security, _normalize_media_domain, _assert_media_url_allowed, _handle_media_url_security

关键源码片段

python/sglang/srt/utils/common.py

安全策略核心实现：新增域名白名单、重定向校验、流式下载大小限制的统一下载器，所有媒体加载路径的最终汇聚点。

```
def download_remote_media(url: str, timeout: float) -> bytes:
    """按统一媒体 URL 策略下载一个 HTTP(S) 媒体对象。

    重定向被手动跟随，确保每个目标地址在建立连接前都先通过白名单校验；
    响应体以流式读取，在保证总请求截止时间的同时强制执行字节数上限，
    避免先将攻击者可控的响应体整体缓冲进内存。
    """
    if timeout <= 0:
        raise ValueError("media URL timeout must be positive")

    session = get_mm_http_session() # 复用 per-thread 连接池
    deadline = time.monotonic() + timeout
    current_url = url

    for redirect_count in range(_MAX_MEDIA_URL_REDIRECTS + 1):
        # 用 requests 的 prepare 逻辑做与 urllib3 一致的 URL 规范化后再校验，
        # 避免反斜杠、userinfo 分隔符等解析器不一致导致的绕过。
        prepared_url = requests.Request("GET", current_url).prepare().url
        if prepared_url is None:
            raise ValueError(f"Invalid media URL: {current_url!r}")
        _assert_media_url_allowed(prepared_url)

        remaining = deadline - time.monotonic()
        if remaining <= 0:
            raise requests.exceptions.Timeout(
                f"Timed out while downloading media URL: {url}"
            )

        with session.get(
            prepared_url,
            allow_redirects=False, # 关闭自动跟随，逐跳校验
            stream=True,
            timeout=remaining,
        ) as response:
            location = response.headers.get("Location")
            if response.status_code in _MEDIA_URL_REDIRECT_STATUS_CODES and location:
```

```
                if redirect_count == _MAX_MEDIA_URL_REDIRECTS:
                    raise requests.exceptions.TooManyRedirects(
                        f"Media URL exceeded {_MAX_MEDIA_URL_REDIRECTS} redirects: {url}"
                    )
                current_url = urljoin(response.url, location)
                continue

            response.raise_for_status()
            max_bytes = _media_url_max_file_size_bytes
            # 声明长度超限的响应直接拒绝，不必等待正文
            content_length = response.headers.get("Content-Length")
            if max_bytes and content_length is not None:
                try:
                    declared_size = int(content_length)
                except ValueError:
                    declared_size = None
                if declared_size is not None and declared_size > max_bytes:
                    raise ValueError(
                        f"Remote media exceeds the {max_bytes} byte download limit"
                    )

            content = bytearray()
            for chunk in response.iter_content(chunk_size=64 * 1024):
                if not chunk:
                    continue
                if time.monotonic() > deadline:
                    raise requests.exceptions.Timeout(
                        f"Timed out while downloading media URL: {url}"
                    )
                if max_bytes and len(content) + len(chunk) > max_bytes:
                    raise ValueError(
                        f"Remote media exceeds the {max_bytes} byte download limit"
                    )
                content.extend(chunk)
            return bytes(content)

    raise AssertionError("unreachable")
```

test/registered/unit/multimodal/test_media_url_security.py

新增的安全回归测试文件，用本地 HTTP 服务器覆盖白名单、重定向校验、大小限制、绕过攻击等全部核心场景，是验证安全策略的关键配套。

```
class _MediaHandler(http.server.BaseHTTPRequestHandler):
    """本地测试用 HTTP 服务，模拟可被安全策略触发的各种响应。"""

    def do_GET(self):
        if self.path == "/other-host-redirect":
            # 302 跳到另一个 host：应被域名白名单拦截，且目标不能被访问
            self.send_response(302)
```

```
            self.send_header(
                "Location",
                f"http://localhost:{self.server.server_port}/redirect-target",
            )
            self.end_headers()
            return

        if self.path == "/chunked-oversized":
            # 不声明 Content-Length，靠流式读取触发大小上限
            self.send_response(200)
            self.end_headers()
            self.wfile.write(b"x" * (1024 * 1024 + 1))
            return

        if self.path == "/redirect-loop":
            self.send_response(302)
            self.send_header("Location", "/redirect-loop")
            self.end_headers()
            return

        self.send_response(404)
        self.end_headers()

    def log_message(self, *_):
        pass # 静默访问日志，避免 CI 输出噪音


class TestMediaURLSecurity(unittest.TestCase):
    def test_redirect_destination_is_checked_before_fetch(self):
        # 白名单只放行 127.0.0.1，跨域重定向应在校验阶段就被拒绝
        configure_media_url_security(["127.0.0.1"], max_file_size_mb=64)
        with self.assertRaisesRegex(ValueError, "not allowed"):
            download_remote_media(self._url("/other-host-redirect"), timeout=5)
        self.assertFalse(self.server.redirect_target_reached)

    def test_streamed_oversized_response_is_rejected(self):
        # 没有声明长度时，流式累计超过上限也必须被拒绝
        configure_media_url_security(["127.0.0.1"], max_file_size_mb=1)
        with self.assertRaisesRegex(ValueError, "download limit"):
            download_remote_media(self._url("/chunked-oversized"), timeout=5)
```

python/sglang/srt/server_args.py

新增两个安全相关 server 参数并在启动早期发布策略，是部署方启用 / 配置安全能力的入口。

```
    allowed_media_domains: A[
        List[str],
        "Restrict client-supplied HTTP(S) image, video, and audio URLs to these "
        "exact hostnames. Redirect destinations are checked against the same "
        "allowlist. When unset, remote media from any domain is allowed.",
```

```
        NS("mm"),
    ] = dataclasses.field(default_factory=list)
    media_url_max_file_size_mb: A[
        int,
        "Maximum size in MiB for one client-supplied remote media download. "
        "The limit is enforced while streaming; set to 0 to disable it.",
        NS("mm"),
    ] = 64

    def _handle_media_url_security(self):
        """Normalize and publish the media URL policy before workers start."""
        # 在 dummy-model 边界之前发布，让所有 worker 共享同一策略；
        # 默认 64 MiB，0 表示禁用大小限制。
        self.allowed_media_domains = configure_media_url_security(
            self.allowed_media_domains,
            self.media_url_max_file_size_mb,
        )
```

评论区精华

本 PR 的 review_comments_count 为 0，PR 评论区没有实质设计争论，主要由作者发起的 3 轮 /rerun-test 指令与机器人执行结果构成（覆盖 test_media_url_security.py、test_server_args_migration.py、test_base_processor_bad_input.py、test_mm_process_config.py、test_kimi_k25.py、rust/qwen e2e parity 等），最终全绿后无异议合并。3 次 commit 的演进顺序（功能实现 → 直接读取 server args → 更新测试 fixtures）说明作者是先完成核心逻辑，再简化参数访问方式，最后补齐测试配套。

- CI 回归验证 (other): 三轮 rerun 全部通过，PR 无 review 评论即合并；说明变更未破坏既有多模态链路。

风险与影响

- 风险：
  1. DNS 重绑定（TOCTOU）未被覆盖：_assert_media_url_allowed 校验的是主机名字符串而非解析后的 IP，域名解析与连接建立之间若 DNS 变化仍可绕过；窗口极小且测试未覆盖，属于已知盲区。
  2. 默认行为变化：media_url_max_file_size_mb 默认 64 MiB，此前远程下载无大小限制，超大媒体（如 > 64 MiB 视频）的部署会开始失败，属于静默行为变更；mimo_v2.has_audio_track 从 ffprobe Range 探测改为整体下载，任何超过限制的视频 URL 的音频探测都会被拒绝。
  3. 进程级全局状态：configure_media_url_security 修改模块级全局变量，ServerArgs.__post_init__、base_processor.py、encode_server.py 多处调用；同一进程内多 engine 配置并存时会互相覆盖，与 #34376 修复的 per-runner 配置污染模式类似，测试均通过 try/finally 重置全局状态来缓解。
  4. 下载路径回归面：moss_vl.py 从流式写文件改为整块内容写临时文件，inkling.py 从 urllib.request 改为 requests 下载（UA、连接复用等行为有细微差异）；受 64 MiB 上

限约束，内存风险可控，但涉及 4 个模型特定处理器的行为变化。

5. 未开启白名单时的防护缺口：白名单默认关闭时，SSRF 防护实际不生效，只剩大小限制；文档需明确提醒部署方主动开启。 - 影响：对部署方：新增 --allowed-media-domains 与 --media-url-max-file-size-mb 两个可选参数，白名单 opt-in、默认兼容旧行为，但 64 MiB 默认上限会约束此前无限制的超大媒体下载。对系统：公共 loader 与 4 个模型特定 processor、encode server、base processor 统一走安全下载器，消除了 requests/urllib/ffprobe 多套 HTTP 路径，安全策略可集中演进。对团队：建立了可复用的媒体 URL 安全基线与本地 HTTP 测试基础设施（可扩展模拟更多攻击场景）；影响范围覆盖所有接收远程多媒体 URL 的多模态服务部署，主要在数据处理链路而非推理路径。 - 风险标记：默认 opt-in 白名单，全局配置状态，下载路径行为变更，DNS 重绑定未覆盖，64 MiB 默认限制

关联脉络

- PR #34913 [CI] Move the static ratchets back to CPU unit tests: 与本 PR 共享 test_server_args_migration.py 及 server_args 静态 ratchet 机制；本 PR 在 server_args.py 新增参数后需继续满足 global_config_read / mutation ratchet 约束。
- PR #34819 config: the post-publish consumers of the supplied-instance surface read the bags: 仓库正推进 server_args 直读迁移到 config bags；本 PR 在 base_processor.py / encode_server.py 直接读取 server_args 并发布策略，属于后续迁移需覆盖的新消费点。
- PR #34376 [Fix] Make the linear-attn kernel choice per-runner, and pin draft/target loader-hook parity: configure_media_url_security 是进程级全局状态，与 #34376 修复的 per-runner 配置污染模式相似，可借鉴其隔离与测试思路。