HTMLHeaderTextSplitter.split_text_from_url SSRF Redirect Bypass

Moderate **ccurme** published **GHSA-fv5p-p927-qmxr** last week

Package

langchain-text-splitters

Affected versions

<1.1.2

Patched versions

>=1.1.2

Description

Summary

`HTMLHeaderTextSplitter.split_text_from_url()` validated the initial URL using `validate_safe_url()` but then performed the fetch with `requests.get()` with redirects enabled (the default). Because redirect targets were not revalidated, a URL pointing to an attacker-controlled server could redirect to internal, localhost, or cloud metadata endpoints, bypassing SSRF protections.

The response body is parsed and returned as `Document` objects to the calling application code. Whether this constitutes a data exfiltration path depends on the application: if it exposes Document contents (or derivatives) back to the requester who supplied the URL, sensitive data from internal endpoints could be leaked. Applications that store or process Documents internally without returning raw content to the requester are not directly exposed to data exfiltration through this issue.

Affected versions

- `langchain-text-splitters` < 1.1.2

Patched versions

- `langchain-text-splitters` >= 1.1.2 (requires `langchain-core` >= 1.2.31)

Affected code

File: `libs/text-splitters/langchain_text_splitters/html.py` — `split_text_from_url()`

The vulnerable pattern validated the URL once then fetched with redirects enabled:

```
validate_safe_url(url, allow_private=False, allow_http=True)
response = requests.get(url, timeout=timeout, **kwargs)
```

Attack scenario

1. A developer passes external URLs to `split_text_from_url()`, relying on its built-in `validate_safe_url()` check to block requests to internal networks.
2. An attacker supplies a URL pointing to a public host they control. The URL passes `validate_safe_url()` (public hostname, public IP).
3. The attacker's server responds with a `302` redirect to an internal endpoint (e.g., an unauthenticated internal admin API, or a cloud instance metadata service that does not require request headers — such as AWS IMDSv1).
4. `requests.get()` follows the redirect automatically. The redirect target is **not** revalidated.
5. The response body is parsed and returned as `Document` objects to the application.

Notes:

- The core issue is a bypass of an explicitly provided SSRF protection. `split_text_from_url()` included `validate_safe_url()` specifically to be safe with untrusted URLs — the redirect loophole defeated that guarantee.
- Cloud metadata endpoints that require special headers (AWS IMDSv2, GCP, Azure) are not reachable through this bug because the attacker does not control request headers. AWS IMDSv1, which requires no headers, is reachable.
- Data exfiltration requires the application to return Document contents to the party that supplied the URL. The SSRF itself — forcing the server to issue a request to an internal endpoint — does not require this.

Fix

The fix replaces `requests.get()` with an SSRF-safe httpx transport (`SSRFSafeSyncTransport` from `langchain-core`) that validates DNS results and pins connections to validated IPs on every request, including redirect targets, eliminating redirect-based bypasses.

Additionally, `split_text_from_url()` has been deprecated. Users should fetch HTML content themselves and pass it to `split_text()` directly.

Severity

Moderate 6.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

CVE ID

CVE-2026-41481

Weaknesses

▶ CWE-918

Credits

 **Aeg1sx** Reporter