What is HTTP request smuggling?

HTTP request smuggling is a vulnerability where a front-end (proxy, load balancer, CDN) and a back-end server disagree about where one HTTP request ends, usually due to conflicting `Content-Length` and `Transfer-Encoding` headers. The attacker exploits this desync to smuggle a partial request that gets prepended to the next user's request, enabling response poisoning, credential capture, control bypass, and cache poisoning. The fix is making the whole chain parse requests identically and prefer HTTP/2 end-to-end.

HOW IT WORKS

01 How it works and example

The attacker sends a request with ambiguous length headers (classic CL.TE or TE.CL desync):

- One server honours Content-Length, the other honours Transfer-Encoding, so a chunk of the attacker's body is interpreted by the back-end as the start of the next request.
- That smuggled prefix can capture another user's request (stealing their cookies/credentials), bypass front-end security controls, or poison the cache so other users receive attacker content.
- Modern variants exploit HTTP/2-to-HTTP/1 downgrade desyncs.

Examples shown for defensive context.

> AGREE ON THE BOUNDARY
>
> *Smuggling is a parsing disagreement. Ensure the whole chain treats request boundaries identically, reject ambiguous* `Content-Length` *plus* `Transfer-Encoding`*, and use HTTP/2 end-to-end where possible.*

HOW TO DEFEND

- Normalise and reject ambiguity: drop requests that contain both Content-Length and Transfer-Encoding, or malformed chunked encoding.
- Use HTTP/2 end-to-end (and avoid downgrading to HTTP/1 to the back-end), which removes most desync surface.
- Keep proxies, CDNs, and servers patched and consistently configured so they parse identically.
- Disable connection reuse to the back-end if the chain cannot be made consistent.
- Test the full proxy chain for desync, not just the application.

SOURCES

[1] OWASP Web Security Testing Guide

[2] MITRE CWE-444: Inconsistent Interpretation of HTTP Requests (HTTP Request Smuggling)

[3] OWASP Top 10

Test your application for request smuggling and 30+ other classes.

`securelayer7.net/learn/application-security/http-request-smuggling`

Open online