What is HTTP request smuggling?

HTTP request smuggling abuses a disagreement between two servers in a chain, typically a front-end proxy or load balancer and a back-end server, about how to determine the length of an HTTP request. By crafting a request with conflicting Content-Length and Transfer-Encoding headers, an attacker can make one server see one request while the other sees two, smuggling a hidden request that gets prepended to the next user's traffic. Consequences include request hijacking, web cache poisoning, and bypassing front-end security controls.

HOW IT WORKS

01 The abuse and impact

Testers craft a request that the two servers parse differently. Shown for defensive context:

- CL.TE: the front-end honors `Content-Length`, the back-end honors `Transfer-Encoding: chunked`. The attacker hides a request after a zero-length chunk terminator.
- TE.CL: the reverse, the front-end honors chunked encoding and the back-end honors `Content-Length`.
- TE.TE: both support chunked, but one can be tricked into ignoring it by obfuscating the header, for example `Transfer-Encoding: xchunked`.

Impact is high: capturing other users' requests and session tokens, poisoning a shared cache so every visitor is served attacker content, and bypassing front-end access controls to reach restricted back-end paths. These chained outcomes make it a priority target in web app pentest engagements against proxied architectures.

HOW TO DEFEND

- Prefer HTTP/2 end to end and do not downgrade to HTTP/1.1 at the back-end, since HTTP/2 carries length unambiguously.
- Reject ambiguous requests: the front-end should refuse any request that contains both `Content-Length` and `Transfer-Encoding`, and normalize headers before forwarding.
- Use the same server software and configuration for front-end and back-end where possible so parsing matches.
- Disable connection reuse to the back-end if the risk cannot otherwise be removed, so a smuggled fragment cannot attach to another user's request.

SOURCES

[1] PortSwigger: HTTP request smuggling

[2] OWASP: HTTP Request Smuggling

[3] MITRE ATT&CK: Exploit Public-Facing Application (T1190)

Find the parsing gap before an attacker does.

securelayer7.net/learn/web-exploitation/what-is-http-request-smuggling

Open online