# CWE Detail – CWE-444

## Description

The product acts as an intermediary HTTP agent  
 (such as a proxy or firewall) in the data flow between two  
 entities such as a client and server, but it does not  
 interpret malformed HTTP requests or responses in ways that  
 are consistent with how the messages will be processed by  
 those entities that are at the ultimate destination.

## Extended Description

HTTP requests or responses ("messages") can be  
 malformed or unexpected in ways that cause web servers or  
 clients to interpret the messages in different ways than  
 intermediary HTTP agents such as load balancers, reverse  
 proxies, web caching proxies, application firewalls,  
 etc. For example, an adversary may be able to add duplicate  
 or different header fields that a client or server might  
 interpret as one set of messages, whereas the intermediary  
 might interpret the same sequence of bytes as a different  
 set of messages. For example, discrepancies can arise in  
 how to handle duplicate headers like two Transfer-encoding  
 (TE) or two Content-length (CL), or the malicious HTTP  
 message will have different headers for TE and  
 CL. The inconsistent parsing and interpretation of messages  
 can allow the adversary to "smuggle" a message to the  
 client/server without the intermediary being aware of it. This weakness is usually the result of the usage  
 of outdated or incompatible HTTP protocol versions in the  
 HTTP agents.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2022-24766: SSL/TLS-capable proxy allows HTTP smuggling when used in tandem with HTTP/1.0 services, due to inconsistent interpretation and input sanitization of HTTP messages within the body of another message

**•** CVE-2021-37147: Chain: caching proxy server has improper input validation (CWE-20) of headers, allowing HTTP response smuggling (CWE-444) using an "LF line ending"

**•** CVE-2020-8287: Node.js platform allows request smuggling via two Transfer-Encoding headers

**•** CVE-2006-6276: Web servers allow request smuggling via inconsistent HTTP headers.

**•** CVE-2005-2088: HTTP server allows request smuggling with both a "Transfer-Encoding: chunked" header and a Content-Length header

**•** CVE-2005-2089: HTTP server allows request smuggling with both a "Transfer-Encoding: chunked" header and a Content-Length header

## Related Attack Patterns (CAPEC)

* CAPEC-273
* CAPEC-33

## Modes of Introduction

**•** Implementation: N/A

## Common Consequences

**•** Impact: Unexpected State, Hide Activities, Bypass Protection Mechanism — Notes: An attacker could create HTTP messages to exploit a number of weaknesses including 1) the message can trick the web server to associate a URL with another URL's webpage and caching the contents of the webpage (web cache poisoning attack), 2) the message can be structured to bypass the firewall protection mechanisms and gain unauthorized access to a web application, and 3) the message can invoke a script or a page that returns client credentials (similar to a Cross Site Scripting attack).

## Potential Mitigations

**•** Implementation: Use a web server that employs a strict HTTP parsing procedure, such as Apache [REF-433]. (Effectiveness: N/A)

**•** Implementation: Use only SSL communication. (Effectiveness: N/A)

**•** Implementation: Terminate the client session after each request. (Effectiveness: N/A)

**•** System Configuration: Turn all pages to non-cacheable. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** When this request is sent to the proxy server, the proxy server parses the first four lines of the POST request and encounters the two "Content-Length" headers. The proxy server ignores the first header, so it assumes the request has a body of length 54 bytes. Therefore, it treats the data in the next three lines that contain exactly 54 bytes as the first request's body:

**•** When this request is sent to the web server, the first POST request has a content-length of 49,223 bytes, and the firewall treats the line with 49,152 copies of "z" and the lines with an additional lines with 71 bytes as its body (49,152+71=49,223). The firewall then continues to parse what it thinks is the second request starting with the line with the third POST request.

**•** N/A

## Notes

**•** Theoretical: Request smuggling can be performed due to a multiple interpretation error, where the target is an intermediary or monitor, via a consistency manipulation (Transfer-Encoding and Content-Length headers).