It can be used to escalate to more malicious attacks such as Cross-site Scripting (XSS), page injection, web cache poisoning, cache-based defacement, and more

A CRLF injection vulnerability exists if an attacker can inject the CRLF characters into a web application, for example using a user input form or an HTTP request.

**How to test**

**Add a fake HTTP response header: Content-Length: 0. This causes the web browser to treat this as a terminated response and begin parsing a new response.**

**Add a fake HTTP response: HTTP/1.1 200 OK. This begins the new response.**

**Add another fake HTTP response header: Content-Type: text/html. This is needed for the web browser to properly parse the content.**

**Add yet another fake HTTP response header: Content-Length: 25. This causes the web browser to only parse the next 25 bytes.**

**Add page content with an XSS: <script>alert(1)</script>. This content has exactly 25 bytes.**

**Because of the Content-Length header, the web browser ignores the original content that comes from the web server**

example : http://www.example.com/somepage.php?page=%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2025%0d%0a%0d%0a%3Cscript%3Ealert(1)%3C/script%3E

I tried in alotromutual.com - worked for me

