SCENARIO

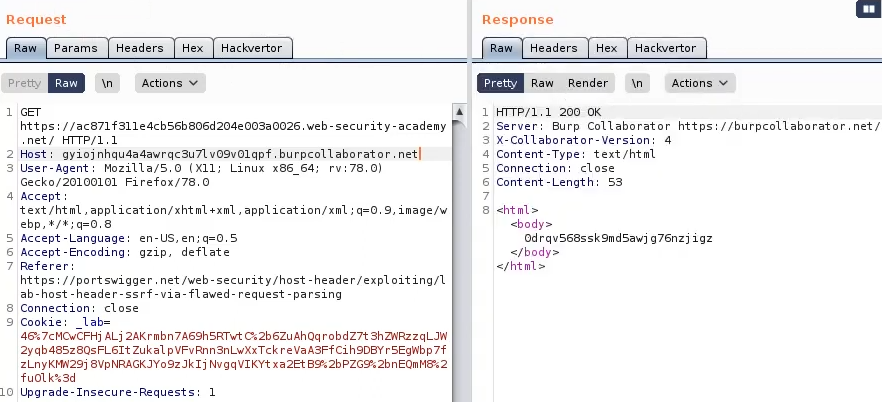
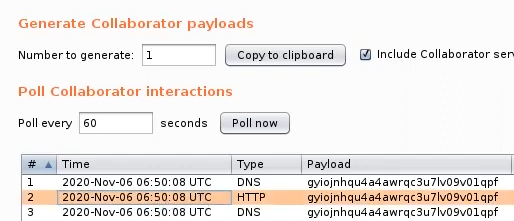
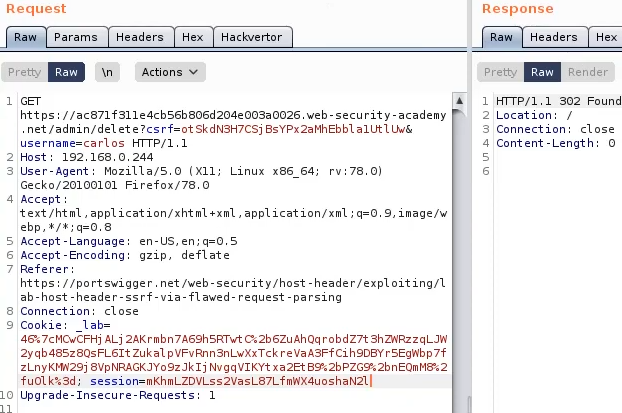
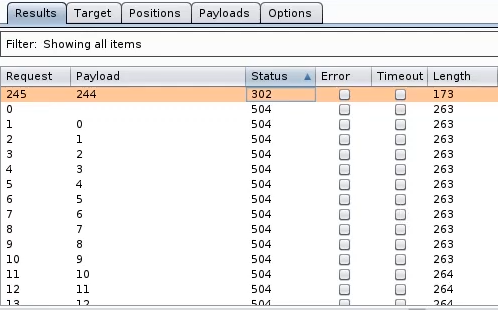
This application is vulnerable to routing-based SSRF due to its flawed parsing of the request's intended host. We can exploit this to access an insecure intranet admin panel located on an internal IP address. We will try to get access to the admin panel.

**PROCEDURE**

1. Go to the web application and using BurpSuite send the request for homepage to the Repeater.
2. We see that if we change the Host header value, we can no longer access the page but we can enter the entire URL of homepage in place of just **/** in order to access the page**.**
3. Go to the Collaborator Client and copy the URL from there and paste in place of the Host Header in the request then send the request, we see that after polling we successfully made some requests to an arbitrary server.
4. Send the GET request to BurpSuite’s Intruder and **untick the Update host header to match target option.**
5. Replace the Host header value with the provided IP Address which is 192.168.0.0 and add the last 0 as payload then brute force the address in order to look for a 302 response.
6. We see that we got a 302 response and now we were redirected to the admin panel but we can not delete the user because of the presence of CSRF token so we will inject the Payload in place of the URL to delete something.
7. Now send the request to access the admin page and we get access to it.

**PAYLOAD**

https://0abd00ac0372464584e205f70039006e.web-security-academy.net/admin/delete?csrf=D3MstUtNSyaZ70IbQ5PTZQFzpqsIZnxT&username=carlos

**PROOF OF CONCEPT**

**REMEDIATION**

1. **Restrict Outgoing Requests:** By default, your server should not be able to make arbitrary outgoing requests. Limit requests to only trusted and necessary domains.
2. **Whitelist Expected Hosts:** Create a whitelist of allowed hosts and only serve requests if the host header matches an entry in this whitelist.
3. **Use Secure Parsers:** When parsing user input or headers, ensure that the parsing is done securely. If using third-party libraries or frameworks, ensure they are up to date and securely handle edge cases.
4. **Reject Suspicious Requests:** If a request looks suspicious, such as containing certain patterns or sequences, reject it outright.
5. **Disable Redirects:** Do not allow URL redirection based solely on user-provided input. If necessary, use a mapping system where the user provides an identifier, and the server determines the correct URL based on that identifier.