SCENARIO

The application is vulnerable to web cache poisoning because it accepts GET requests that have a body, but does not include the body in the cache key. There is also inconsistent parameter parsing between the cache and the back-end. We will try to poison the cache with a response that executes alert(1) in the visitor's browser.

**PROCEDURE**

1. Open the web application and send the request for **/js/geolocate.js?callback=setCountryCookie** to BurpSuite’s Repeater and start testing it by adding Cache Buster to the URL.
2. Observe that every page imports the script **/js/geolocate.js**, executing the callback function **setCountryCookie()**.
3. Notice that we can control the name of the function that is called in the response by passing in a duplicate callback parameter via the request body. Also notice that the cache key is still derived from the original callback parameter in the request line.
4. According to the way our injected query parameter we will craft an exploit string which will break out of that tag and trigger our alert by executing arbitrary JavaScript.
5. Append the crafted exploit as shown in the Payload in the request body.
6. 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**

callback=alert(1)

**REMEDIATION**