SCENARIO

The application is vulnerable to web cache poisoning because it handles input from an unkeyed header in an unsafe way. 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 request to BurpSuite’s Repeater and study the GET request to the home URL.
2. Append the Payload 1 into the URL in BurpSuite’s Repeater tab and we see that for every unique value of cache buster, the request is made to the server and then stored in the cache.
3. We can notice that the response is successfully stored in the cache when there comes **X-Cache: HIT/MISS.**
4. Now add Payload 2 in the request and we can notice in the response that the URL has been used to dynamically generate an absolute URL for importing a JavaScript file stored at **/resources/js/tracking.js**
5. Replay the request and observe that the response contains the header **X-Cache: hit** which tells us that the response came from the cache.
6. Go to the exploit server and change the filename to Payload 3 and the body to Payload 4.
7. Now open the GET request in BurpSuite’s Repeater and remove the cache buster and replace the value of **X-Forwarded-Host** to your exploit server id.
8. Send the malicious request 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. ?cb=1234
2. X-Forwarded-Host: example.com
3. /resources/js/tracking.js

**REMEDIATION**