SCENARIO

The application is vulnerable to web cache poisoning that is only exploitable in the user comment on post functionality and the script gets executed every time someone loads the page. We will try to poison the cache with a response that executes alert(document.cookie) in the visitor's browser but we also need to make sure that the response is served to the specific subset of users to which the intended victim belongs.

**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-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/js/tracking.js**.
4. Go to the exploit server and replace the name with the Payload 1 and the body with the Payload 2 and store the exploit now.
5. Get back to BurpSuite’s Repeater and put the value of the **X-Host** parameter as our exploit server URL.
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. Notice that the **Vary header** is used to specify that the **User-Agent** is part of the cache key and we need to find the victim’s **User-Agent** value.
9. Go to the exploit server and open the "Access log". Refresh the page every few seconds until you see requests made by a different user, copy their User-Agent from the logs.
10. Send the malicious request after removing the cache buster parameter paste the victim's User-Agent into the corresponding header. Remove the cache buster from the URL 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/js/tracking.js
2. alert(document.cookie)
3. <img src="https://YOUR-EXPLOIT-SERVER-ID.exploit-server.net/foo" />

**REMEDIATION**

1. **Ignore Headers from Untrusted Sources:** As seen with the X-Host header, your application shouldn't trust or process arbitrary headers, especially when they can influence critical components like URL or resource generation.
2. **Restrict Dynamic Script Insertion:** Avoid inserting scripts or other potentially dangerous elements into pages based on user input or headers. If dynamic content insertion is necessary, ensure strict validation and escaping mechanisms are in place.
3. **Input and Output Encoding:** Ensure that all inputs are validated and encoded correctly, and that outputs are escaped properly to prevent any malicious content from being executed.
4. **Strict Cache Controls:** Use strict cache controls to prevent certain dynamic pages, especially those based on user input or unusual headers, from being cached.
5. **Consistent Behaviour:** Ensure that your application and its caching solutions behave consistently. If one layer of the application recognizes a custom header, every layer (including caching mechanisms) should be aware of its implications.
6. **Use Application Firewalls:** Web Application Firewalls (WAF) can be configured to block or alert on unusual headers or combinations of headers. They can also block suspicious input in user comments or other input fields.
7. **Limit Cached Content Based on Custom Headers:** If your application uses custom headers like X-Host to modify its behavior, you should not cache the resultant content. Alternatively, make sure the cache key is constructed using all parameters and headers that influence the content.
8. **Isolate Cache for Different User Groups:** Use the Vary header properly. If certain content varies based on the user-agent or other headers, either do not cache it or ensure that the caching mechanism can differentiate between different user groups.