**SCENARIO**

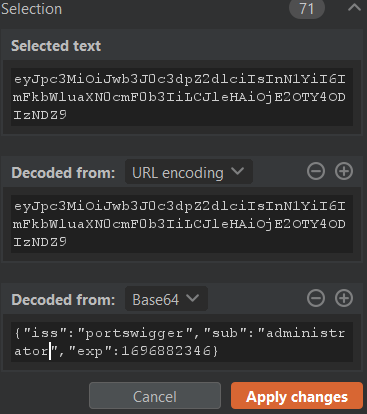
Our web application utilizes JSON Web Tokens (JWT) for session management. Upon a closer look, it seems that the platform fails to validate the JWT signatures appropriately. This weakness might permit unauthorized access to critical areas of the application, such as the administrator panel.

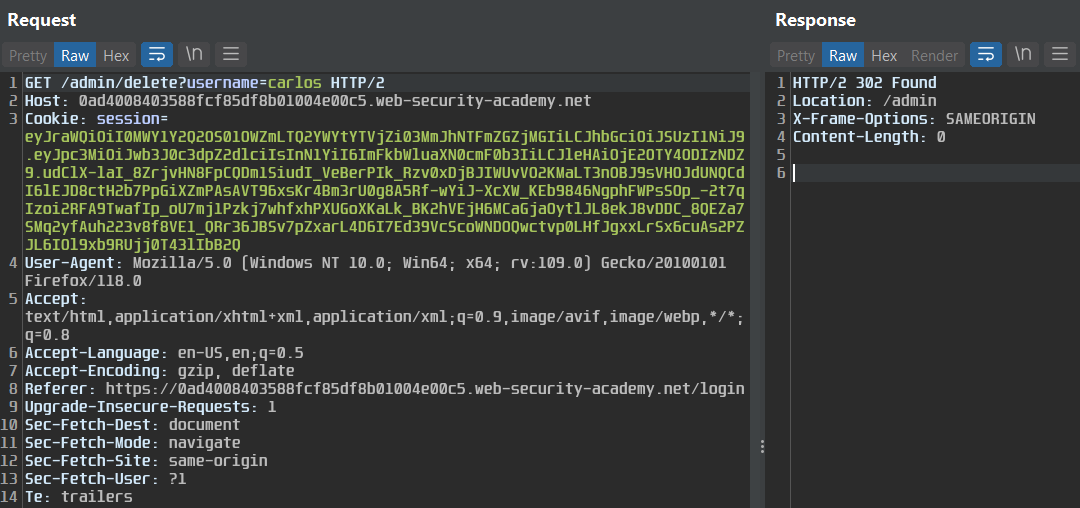
**PROCEDURE**

1. Start by accessing the application and authenticating with our provided credentials: wiener:peter.
2. Post authentication, scrutinize the application's web traffic using Burp Suite, focusing on the Proxy > HTTP history tab.
3. Identify the post-login GET /my-account request and note the JWT session cookie.
4. Isolate and inspect the payload segment of the JWT token. This can be achieved by double-clicking on the payload to use Burp's Inspector panel for decoding the token.
5. Within the decoded payload, note the sub claim that denotes our username. To further investigate the JWT vulnerability, send this request to Burp Repeater.
6. Modify the request path in Burp Repeater to /admin and transmit the request.
7. Understand that the application restricts access to the admin panel unless the user role is that of an administrator.
8. Return to the JWT payload and, using the Inspector panel, alter the sub claim's value from "wiener" to "administrator". Apply the changes.
9. Resend the modified request using Burp Repeater.
10. Upon successful penetration, review the response to spot the link for user deletion. In our case, it's /admin/delete?username=carlos.
11. Make a request to this specific endpoint to delete the user carlos.

**PAYLOAD**

Change in JWT's payload section: "sub": "wiener" to "sub": "administrator"

**PROOF OF CONCEPT**



**REMEDIATION**

1. Always validate the JWT signature server-side before processing the token.
2. Utilize strong, cryptographically secure algorithms when signing JWTs.
3. Consider using a central token store to validate JWTs, thereby allowing for token revocation if needed.
4. Limit JWT data to non-sensitive information and avoid incorporating roles or permissions directly.
5. Employ short expiration times for tokens and require regular token renewal.