**SCENARIO**

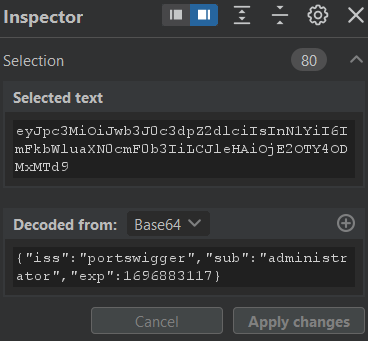
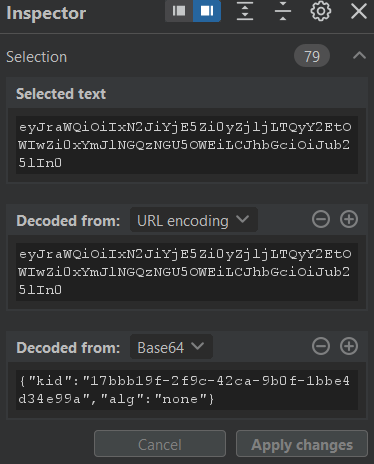
The lab environment operates on a JWT-based session handling mechanism. Unfortunately, the server is misconfigured and accepts JWTs without requiring a signature. This vulnerability might allow unauthorized users to impersonate other accounts, including administrative ones.

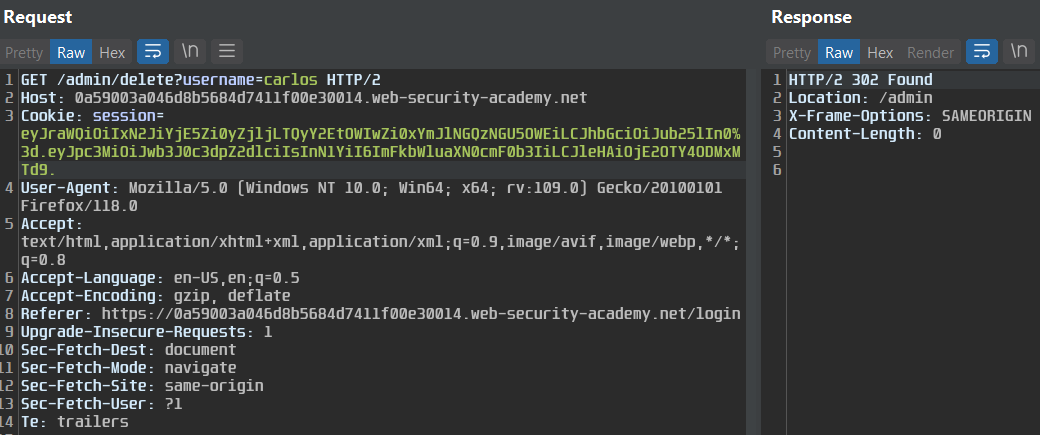
**PROCEDURE**

1. Initially, we log into the provided account using the credentials: wiener:peter.
2. Once authenticated, we navigate to Burp Suite and select the Proxy > HTTP history tab. Here, we spot the post-login GET /my-account request, indicating our session is JWT-based.
3. By double-clicking on the JWT's payload section, the decoded JSON representation becomes visible in the Inspector panel. We note that the sub claim encapsulates our username. This request is promptly forwarded to Burp Repeater for further analysis.
4. Using Burp Repeater, we adjust the request path to /admin and reissue it. The resulting response highlights that only the "administrator" account can access the admin panel.
5. We refocus on the JWT's payload and, through the Inspector, modify the sub claim's value to "administrator". Afterward, we apply the changes.
6. Moving to the JWT's header section, we switch the alg parameter's value to "none" using the Inspector and again apply these alterations.
7. Back in the message editor of Burp Repeater, we carefully remove the JWT's signature. It's crucial to retain the trailing period after the payload section.
8. We send the modified request and, to our success, gain access to the admin panel.
9. Upon examining the response, we identify the URL endpoint responsible for user deletion: /admin/delete?username=carlos. A subsequent request to this endpoint ensures the user "carlos" is deleted.

**PAYLOAD**

Modified JWT with sub claim adjusted to "administrator", alg parameter set to "none", and removal of the JWT's signature (leaving the trailing dot).

******PROOF OF CONCEPT**



**REMEDIATION**

1. Always verify JWT signatures and reject JWTs with an alg set to "none" unless explicitly allowed for specific cases.
2. Store and manage secret keys (for HMAC) or private keys (for RSA and ECDSA) in secure environments and rotate them periodically.
3. Implement strict JWT expiration times and consider using a revocation list for extra security.
4. Educate developers about the risks associated with JWT and best practices for its secure implementation.
5. Use libraries and frameworks that are well-maintained and have been reviewed for security, reducing the chances of implementation mistakes.