SCENARIO

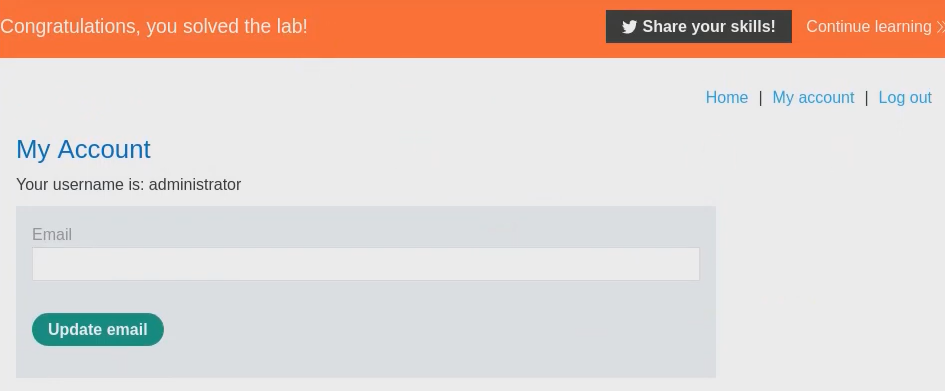
Our e-commerce site appears to have an exploitable SQL injection flaw. It's embedded within the stock check function. The results from this query are openly returned in the app's response.

**PROCEDURE**

1. Navigate through the shop's features and observe the productId and storeId conveyed in XML format via the stock check feature.
2. Intercept the POST /product/stock request and forward it to Burp Repeater.
3. In Repeater, append the storeId to determine if it undergoes evaluation.
4. Use UNION SELECT with the storeId to determine the number of columns returned.
5. If blocked by a WAF, employ XML entity obfuscation techniques such as those provided by Hackvertor.
6. Once bypassed, exploit the SQL injection vulnerability to fetch the administrator user's credentials.
7. Log in using these credentials.

**PAYLOAD**

<storeId>1 UNION SELECT NULL</storeId>

**PROOF OF CONCEPT**

**REMEDIATION**

1. Adopt parameterized queries or use ORM frameworks.
2. Always sanitize and validate user inputs.
3. Establish limitations on what SQL user accounts can do.
4. Use web application firewalls (WAFs) to detect malicious activities.
5. Regularly update and patch systems against known vulnerabilities.