Solution Guide: Broken Session and Authentication Management

In this activity, you demonstrated a number of vulnerabilities related to broken authentication and session management, and provided recommendations for prevention and mitigation.

Part 1: Insecure Login Forms

1. In Kali, launch Firefox and avigate to the IP address of the OWASP BWA machine. Complete the following:

- On the landing page, select the **bWAPP** module.
- Log in with the credentials `bee:bug` .
- On the drop down menu in the top right, sroll down and select **Broken Authentication - Insecure Login Forms**.
- Click on **Hack**.
- Make sure the security level is set to Low.
- Login with the following credentials:
  - Username: `hacker`
  - Password: `exploit`

2. After trying to log in, right-click on the page. In the dropdown menu, select **View Page Source**. Scroll down to the `<form action=` section to find the username and password.
3. After inputting the stolen credentials, you should receive a response that says, `“Successful login! You really are Iron Man :)”` .
4. Why does this web vulnerability exists?
   - Inadequate security built into web application software.
5. How would you would mitigate this web vulnerability?
   - Enforce secure software programming practices.

Part 2: Logout Management

1. Choose **Broken Auth – Logout Management** and click **Hack**.
   - Click **Logout**.
2. Execute the logout management exploit by clicking the back button in the browser, which will log the authenticated session back in.
3. Why does this web vulnerability exists?
   - Improper session invalidation or timeouts.
4. How would you would mitigate this web vulnerability?
   - Force the session to terminate upon logout.

Part 3: Administrative Portals

1. Choose **Session Management – Administrative Portals** and click **Hack**.
   - In the URL, change `admin=0` to `admin=1` and press Enter.
   - Why were you able to log into the webpage?
     - `admin=1` is "true," which tells the server that you have admin credentials.
2. Why does this web vulnerability exists?
   - Exposure of session ID in the URL.
3. How would you would mitigate this web vulnerability?

- Prevent the exposure of session information in the URL and enforce session invalidation or timeouts.