# PortSwigger Access Control Lab Notes

Access control is about enforcing who can do what in a web application. It ensures that users can only access resources and perform actions allowed for their role. PortSwigger categorizes the main issues as:

* Vertical privilege escalation – A low-privileged user (e.g., normal user) can perform actions reserved for higher-privileged users (e.g., admin).
* Horizontal privilege escalation – A user can access another user’s data or resources at the same privilege level (e.g., viewing someone else’s account by changing an ID).
* Context-dependent access flaws – Restrictions missing in specific workflows (e.g., bypassing a “step 1” check by directly calling the “step 2” endpoint).

Common mistakes in access control:

* Relying on client-side checks (hidden fields, UI restrictions).
* Inconsistent enforcement across endpoints.
* Predictable or manipulable identifiers.
* Assuming obscurity (e.g., “secret” URLs) is enough.

Key takeaway: Always enforce access control server-side, consistently, and based on the user’s session/role—not on user input or hidden parameters.

1. Unprotected admin functionality

This lab has an unprotected admin panel.

Solve the lab by deleting the user carlos.

1. Solution
2. Go to the lab and view robots.txt by appending /robots.txt to the lab URL. Notice that the Disallow line discloses the path to the admin panel.
3. In the URL bar, replace /robots.txt with /administrator-panel to load the admin panel.
4. Delete carlos.
5. Unprotected admin functionality with unpredictable URL

This lab has an unprotected admin panel. It's located at an unpredictable location, but the location is disclosed somewhere in the application.

Solve the lab by accessing the admin panel, and using it to delete the user carlos.

 Solution

1. Review the lab home page's source using Burp Suite or your web browser's developer tools.
2. Observe that it contains some JavaScript that discloses the URL of the admin panel.
3. Load the admin panel and delete carlos.
4. User role controlled by request parameter

This lab has an admin panel at /admin, which identifies administrators using a forgeable cookie.

Solve the lab by accessing the admin panel and using it to delete the user carlos.

You can log in to your own account using the following credentials: wiener:peter

 Solution

1. Browse to /admin and observe that you can't access the admin panel.
2. Browse to the login page.
3. In Burp Proxy, turn interception on and enable response interception.
4. Complete and submit the login page, and forward the resulting request in Burp.
5. Observe that the response sets the cookie Admin=false. Change it to Admin=true.
6. Load the admin panel and delete carlos.
7. User role can be modified in user profile

This lab has an admin panel at /admin. It's only accessible to logged-in users with a roleid of 2.

Solve the lab by accessing the admin panel and using it to delete the user carlos.

You can log in to your own account using the following credentials: wiener:peter

 Solution

1. Log in using the supplied credentials and access your account page.
2. Use the provided feature to update the email address associated with your account.
3. Observe that the response contains your role ID.
4. Send the email submission request to Burp Repeater, add "roleid":2 into the JSON in the request body, and resend it.
5. Observe that the response shows your roleid has changed to 2.
6. Browse to /admin and delete carlos.
7. User ID controlled by request parameter

This lab has a horizontal privilege escalation vulnerability on the user account page.

To solve the lab, obtain the API key for the user carlos and submit it as the solution.

You can log in to your own account using the following credentials: wiener:peter

 Solution

1. Log in using the supplied credentials and go to your account page.
2. Note that the URL contains your username in the "id" parameter.
3. Send the request to Burp Repeater.
4. Change the "id" parameter to carlos.
5. Retrieve and submit the API key for carlos.
6. User ID controlled by request parameter, with unpredictable user IDs

This lab has a horizontal privilege escalation vulnerability on the user account page, but identifies users with GUIDs.

To solve the lab, find the GUID for carlos, then submit his API key as the solution.

You can log in to your own account using the following credentials: wiener:peter

Solution

1. Find a blog post by carlos.
2. Click on carlos and observe that the URL contains his user ID. Make a note of this ID.
3. Log in using the supplied credentials and access your account page.
4. Change the "id" parameter to the saved user ID.
5. Retrieve and submit the API key.
6. User ID controlled by request parameter with data leakage in redirect

This lab contains an access control vulnerability where sensitive information is leaked in the body of a redirect response.

To solve the lab, obtain the API key for the user carlos and submit it as the solution.

You can log in to your own account using the following credentials: wiener:peter

 Solution

1. Log in using the supplied credentials and access your account page.
2. Send the request to Burp Repeater.
3. Change the "id" parameter to carlos.
4. Observe that although the response is now redirecting you to the home page, it has a body containing the API key belonging to carlos.
5. Submit the API key.
6. User ID controlled by request parameter with password disclosure

lab has user account page that contains the current user's existing password, prefilled in a masked input.

To solve the lab, retrieve the administrator's password, then use it to delete the user carlos.

You can log in to your own account using the following credentials: wiener:peter

 Solution

1. Log in using the supplied credentials and access the user account page.
2. Change the "id" parameter in the URL to administrator.
3. View the response in Burp and observe that it contains the administrator's password.
4. Log in to the administrator account and delete carlos.
5. Insecure direct object references

This lab stores user chat logs directly on the server's file system, and retrieves them using static URLs.

Solve the lab by finding the password for the user carlos, and logging into their account.

 Solution

1. Select the **Live chat** tab.
2. Send a message and then select **View transcript**.
3. Review the URL and observe that the transcripts are text files assigned a filename containing an incrementing number.
4. Change the filename to 1.txt and review the text. Notice a password within the chat transcript.
5. Return to the main lab page and log in using the stolen credentials.
6. URL-based access control can be circumvented

This website has an unauthenticated admin panel at /admin, but a front-end system has been configured to block external access to that path. However, the back-end application is built on a framework that supports the X-Original-URL header.

To solve the lab, access the admin panel and delete the user carlos.

 Solution

1. Try to load /admin and observe that you get blocked. Notice that the response is very plain, suggesting it may originate from a front-end system.
2. Send the request to Burp Repeater. Change the URL in the request line to / and add the HTTP header ***X-Original-URL: /invalid***. Observe that the application returns a ***"not found"*** response. This indicates that the back-end system is processing the URL from the X-Original-URL header.
3. Change the value of the X-Original-URL header to /admin. Observe that you can now access the admin page.
4. To delete carlos, ***add ?username=carlos*** to the real query string, and change the***X-Original-URL path to /admin/delete.***
5. Method-based access control can be circumvented

This lab implements access controls based partly on the HTTP method of requests. You can familiarize yourself with the admin panel by logging in using the credentials administrator:admin.

To solve the lab, log in using the credentials wiener:peter and exploit the flawed access controls to promote yourself to become an administrator.

Solution

1. Log in using the admin credentials.
2. Browse to the admin panel, promote carlos, and send the HTTP request to Burp Repeater.
3. Open a private/incognito browser window, and log in with the non-admin credentials.
4. Attempt to re-promote carlos with the non-admin user by copying that user's session cookie into the existing Burp Repeater request, and observe that the response says "Unauthorized".
5. Change the method from POST to POSTX and observe that the response changes to "missing parameter".
6. Convert the request to use the GET method by right-clicking and selecting "Change request method".
7. Change the username parameter to your username and resend the request.
8. Multi-step process with no access control on one step

This lab has an admin panel with a flawed multi-step process for changing a user's role. You can familiarize yourself with the admin panel by logging in using the credentials administrator:admin.

To solve the lab, log in using the credentials wiener:peter and exploit the flawed access controls to promote yourself to become an administrator.

 Solution

1. Log in using the admin credentials.
2. Browse to the admin panel, promote carlos, and send the confirmation HTTP request to Burp Repeater.
3. Open a private/incognito browser window, and log in with the non-admin credentials.
4. Copy the non-admin user's session cookie into the existing Repeater request, change the username to yours, and replay it.

13.Referer-based access control

This lab controls access to certain admin functionality based on the Referer header. You can familiarize yourself with the admin panel by logging in using the credentials administrator:admin.

To solve the lab, log in using the credentials wiener:peter and exploit the flawed access controls to promote yourself to become an administrator.

 Solution

1. Log in using the admin credentials.
2. Browse to the admin panel, promote carlos, and send the HTTP request to Burp Repeater.
3. Open a private/incognito browser window, and log in with the non-admin credentials.
4. Browse ***to /admin-roles?username=carlos&action=upgrade*** and observe that the request is treated as unauthorized due to the absent Referer header.
5. Copy the non-admin user's session cookie into the existing Burp Repeater request, change the username to yours, and replay it.