What is access control?
Access control (or authorization) is the application of constraints on who (or what) can perform attempted actions or access resources that they have requested.

Lab1: Unprotected admin functionality

Description: This lab has an unprotected admin panel.

Solve the lab by deleting the user carlos.

Testing procedure and snapshot:

Go to the lab and view robots.txt by appending /robots.txt to the lab URL.

Note that the disallow line identifies the path to the admin panel.

In the URL bar replace /robots.txt with /administrator-panel to load the admin panel.

Delete carlos.

Lab2: Unprotected admin functionality with unpredictable URL

Description: 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.

Testing procedure and snapshot:

Review the lab homepage's source using Burp Suite or your web browser's developer tools.

Observe that it contains some JavaScript that discloses the URL of the admin panel.

Load the admin panel and delete carlos.

Lab3: User role controlled by request parameter

Description: 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 have an account on the application that you can use to help design your attack. The credentials are: wiener:peter.

Testing procedure and snapshot:

Browse to /admin and observe that you can't access the admin panel.

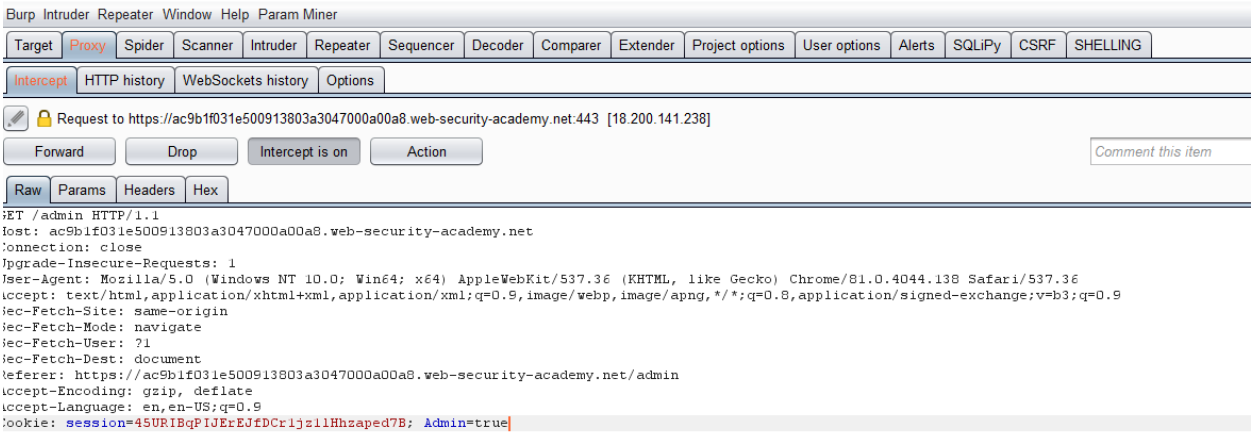
Browse to the login page.

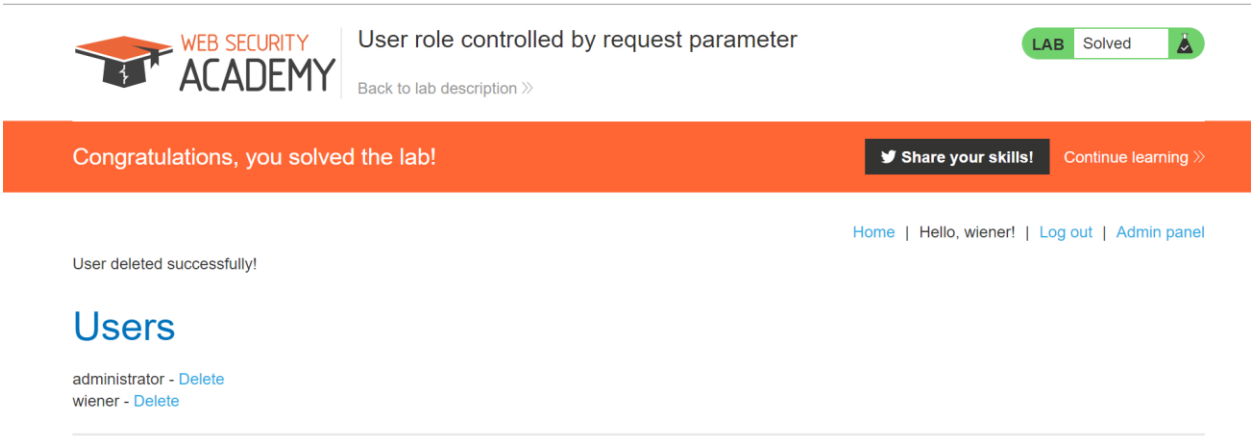
In Burp Proxy, turn interception on and enable response interception.

Complete and submit the login page, and forward the resulting request in Burp.

Observe that the response sets the cookie Admin=false. Change it to Admin=true.

Load the admin panel and delete carlos.





Lab4: User role can be modified in user profile

Description: 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 wiener:peter.

Testing procedure and snapshot:

Log in using the supplied credentials.

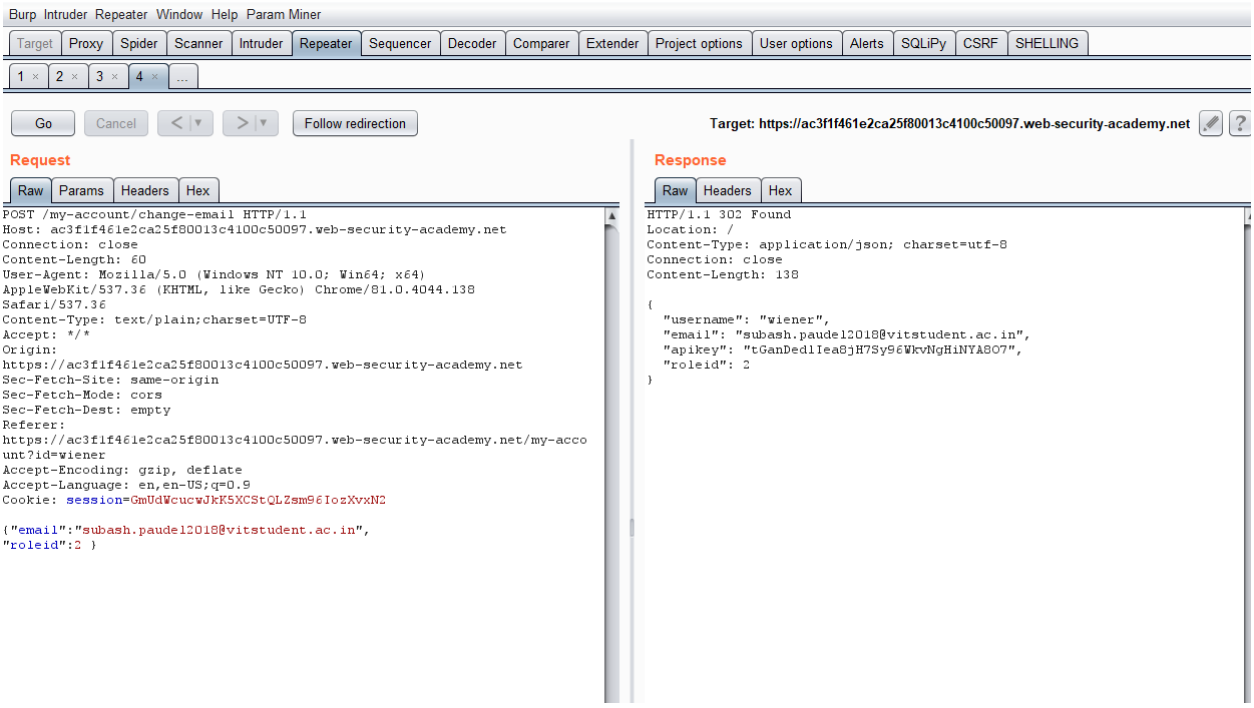
Click on "My Account" and submit a new email address.

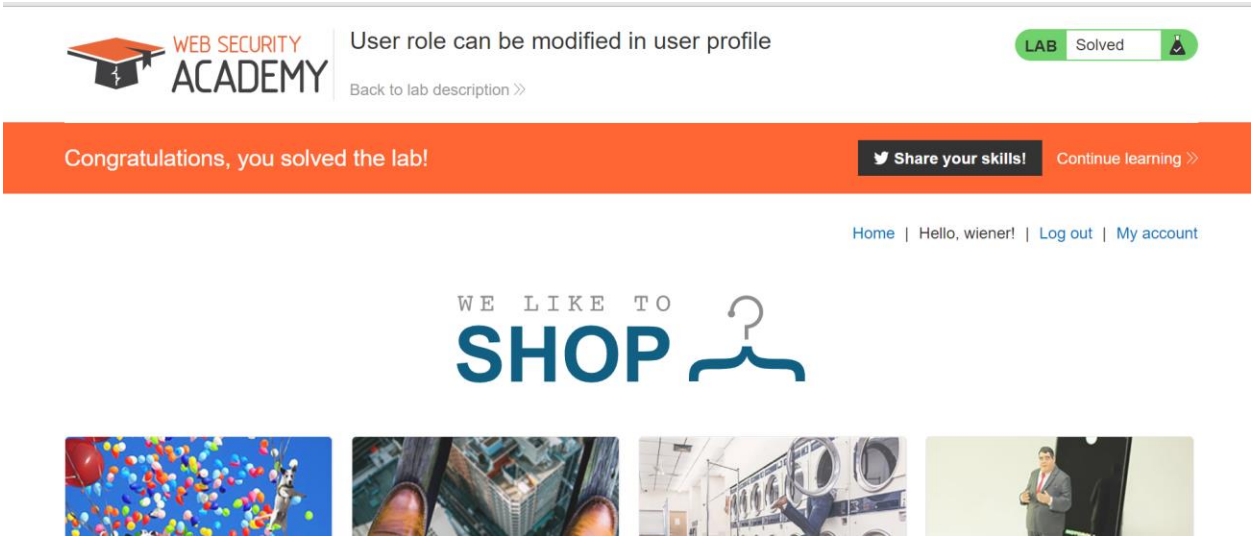
Observe that the response contains your role ID.

Send the email submission request to Burp Repeater, add "roleid":2 into the JSON in the request body, and resend it.

Observe that the response shows your roleid has changed to 2.

Browse to /admin and delete carlos.





Lab5: URL-based access control can be circumvented

Description: 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.

Testing procedure and snapshot:

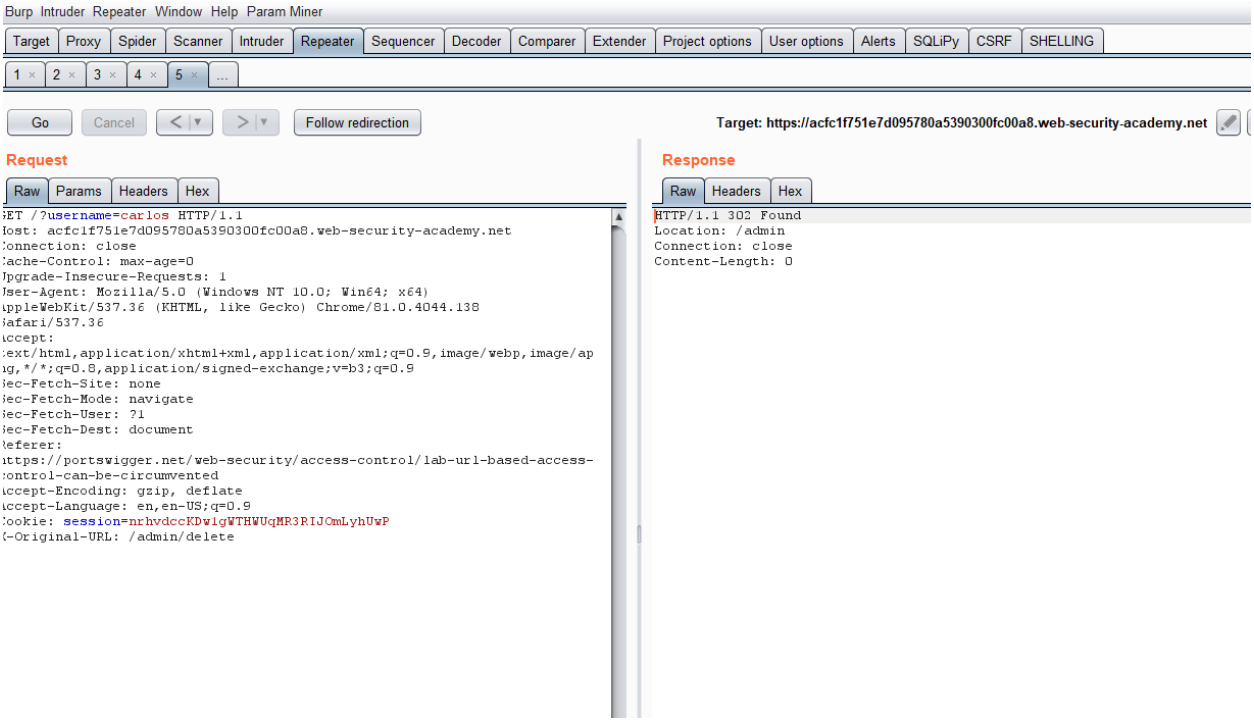
Try to load /admin and observe that you get blocked.

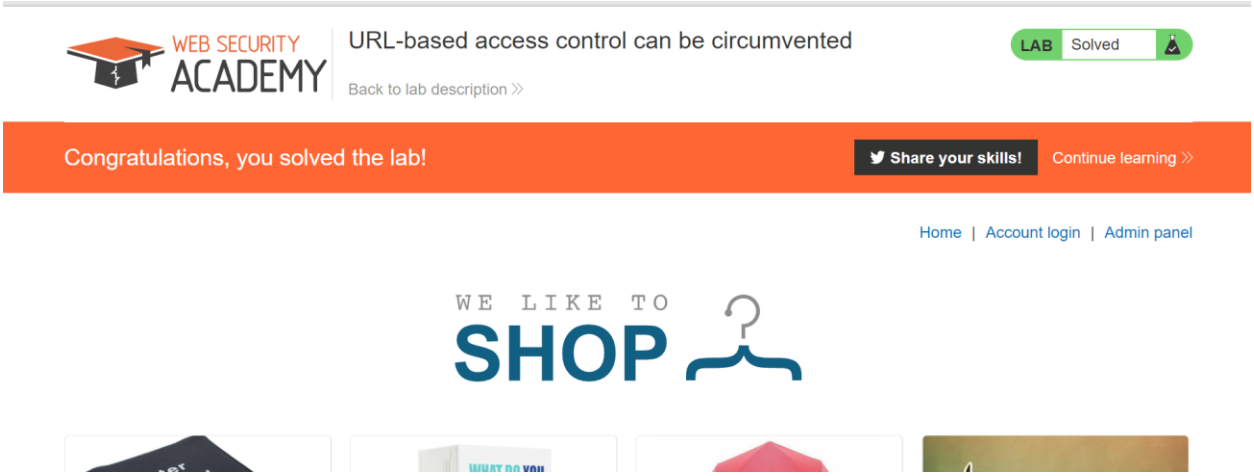
Observe that the response is very plain, suggesting it may originate from a front-end system.

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.

Change the value of the X-Original-URL header to /admin. Observe that you can now access the admin page.

To delete the user carlos, add ?username=carlos to the real query string, and change the X-Original-URL path to /admin/delete.





Lab6: Method-based access control can be circumvented

Description: 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 administrator:admin.

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

Access the lab

Testing procedure and snapshot:

Log in using the admin credentials.

Browse to the admin panel, promote carlos, and send the HTTP request to Burp Repeater.

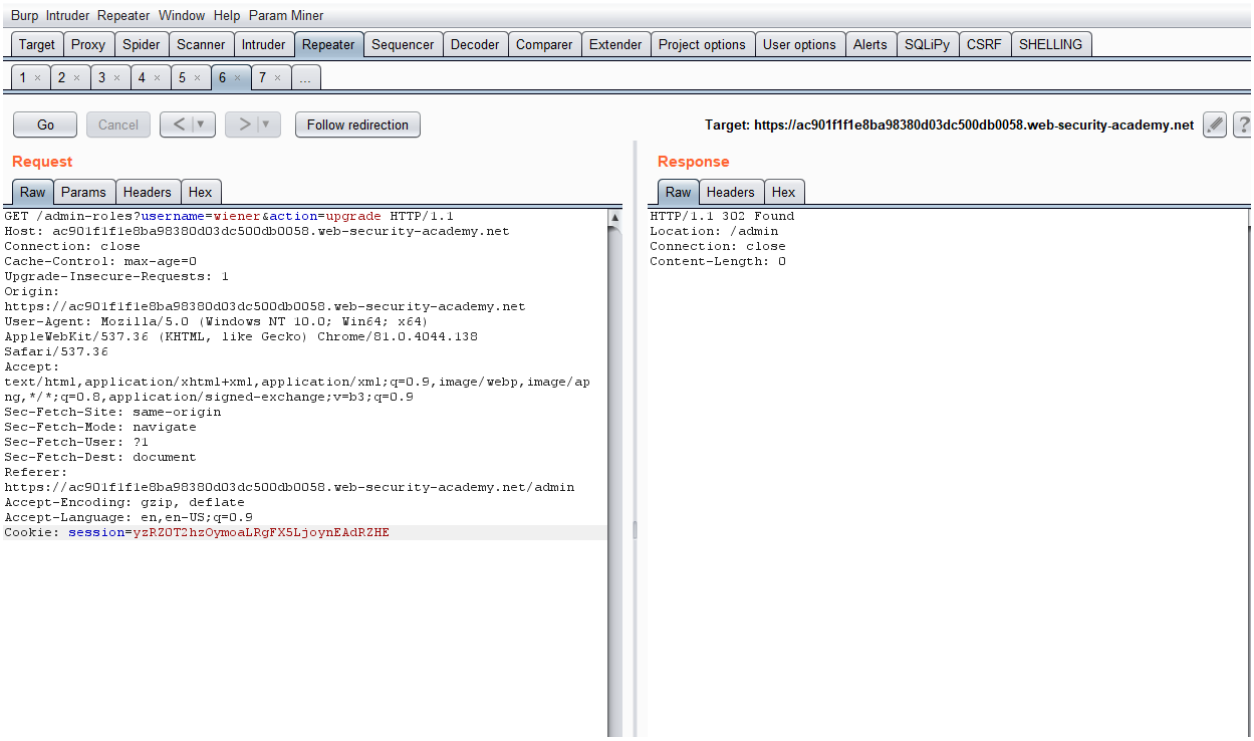
Open a private/incognito browser window, and log in with the non-admin credentials.

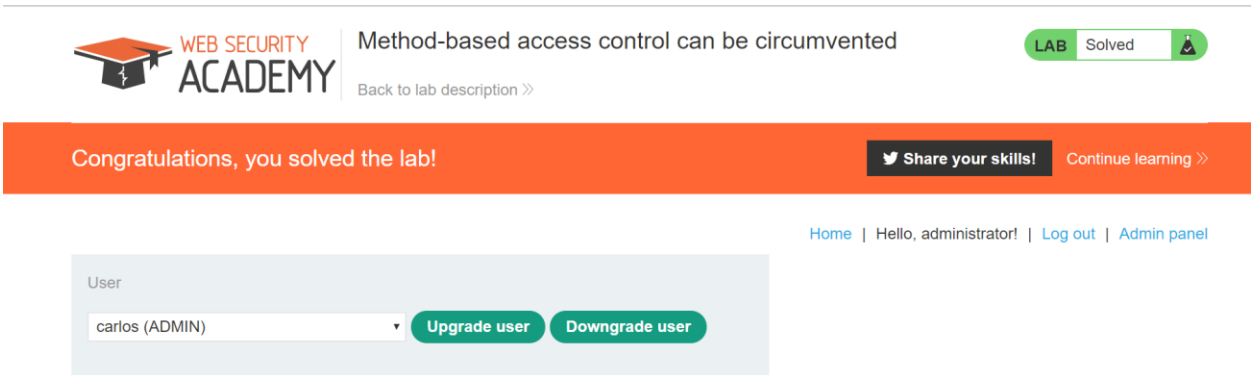
Attempt to re-promote carlos with the non-admin user by copying that user's session cookie into the existing Burp Repeater request, and observe that the response says "Unauthorized".

Change the method from POST to POSTX and observe that the response changes to "missing parameter".

Convert the request to use the GET method by right-clicking and selecting "Change request method".

Change the username parameter to your username and resend the request.





Lab7: User ID controlled by request parameter

Description: This lab has a horizontal privilege escalation vulnerability on the My Account page.

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

You can access your own account using wiener:peter.

Testing procedure and snapshot:

Log in using the supplied credentials and access "Account Details".

Note that the URL contains your username in the "id" parameter.

Send the request to Burp Repeater.

Change the "id" parameter to carlos.

Retrieve and submit the API key for carlos.

Burp Intruder Repeater Window Help Param Miner

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts | SQLiPy | CSRF | SHELLING

1 × ...

Go | Cancel | < | > | Target: https://ac421ff71e1d331380e10822000500af.web-security-academy.net

Request

Raw | Params | Headers | Hex

```
GET /my-account?id=carlos HTTP/1.1
Host: ac421ff71e1d331380e10822000500af.web-security-academy.net
Connection: close
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp
,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://ac421ff71e1d331380e10822000500af.web-security-academy.ne
t/
Accept-Encoding: gzip, deflate
Accept-Language: en,en-US;q=0.9
Cookie: session=ES97jvobVxpHSAYcVnUB86YDS3tM2jk3
```

Response

Raw | Headers | Hex | HTML | Render

```
href="https://portswigger.net/web-security/access-control/lab-user-id-controlled
-by-request-parameter">
                                Back to lab description <svg
version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" viewBox="0 0 28 30"
enable-background="new 0 0 28 30" xml:space="preserve" title="back-arrow">
    <g>
        <polygon points="1.4,0 0,1.2 12.6,15 0,28.8 1.4,30 15.1,15"></polygon>
        <polygon points="14.3,0 12.9,1.2 25.6,15 12.9,28.8 14.3,30
28,15"></polygon>
    </g>
</svg>
                        </a>
                    </div>
                    <div class="widgetcontainer-lab-status is-notsolved">
                        <span>LAB</span>
                        <p>Not solved</p>
                        <span class="lab-status-icon"></span>
                    </div>
                </div>
            </section>
            </div>
            <section class="maincontainer">
                <div class="container is-page">
                    <header class="navigation-header">
                        <section class="top-links">
                            <a href=/>Home</a><p>|</p>
                             Hello, wiener!<p>|</p>
                            <a href="/logout">Log out</a><p>|</p>
                            <a href="/my-account?id=wiener">My
account</a><p>|</p>
                        </section>
                    </header>
                    <h1>My Account</h1>
                    <div>Your API Key is:
hgM6ZKGjS4lM9t2YnF5XHS94fUKF10dA</div><br/>
                    <form class="login-form" action="/my-account/change-email"
method="POST">
                        <label>Email</label>
                        <input required type="email" name="email" value="">
                        <input required type="hidden" name="csrf"
value="krJT9bcYPOXqcOYYdVRaohj1SzfdmtE2">
                        <button class='button' type='submit'> Update email
```

WEB SECURITY ACADEMY | User ID controlled by request parameter | LAB Solved

Back to lab description »

Congratulations, you solved the lab! | Share your skills! | Continue learning »

Home | Hello, wiener! | Log out | My account

My Account

Your API Key is: inb9CjelHMMzTy6zYUua9SW5m8YzQMff

Email

Lab8: User ID controlled by request parameter, with unpredictable user IDs

Description: This lab has a horizontal privilege escalation vulnerability on the My 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 access you own account using wiener:peter.

Testing procedure and snapshot:

Find a blog post by carlos.

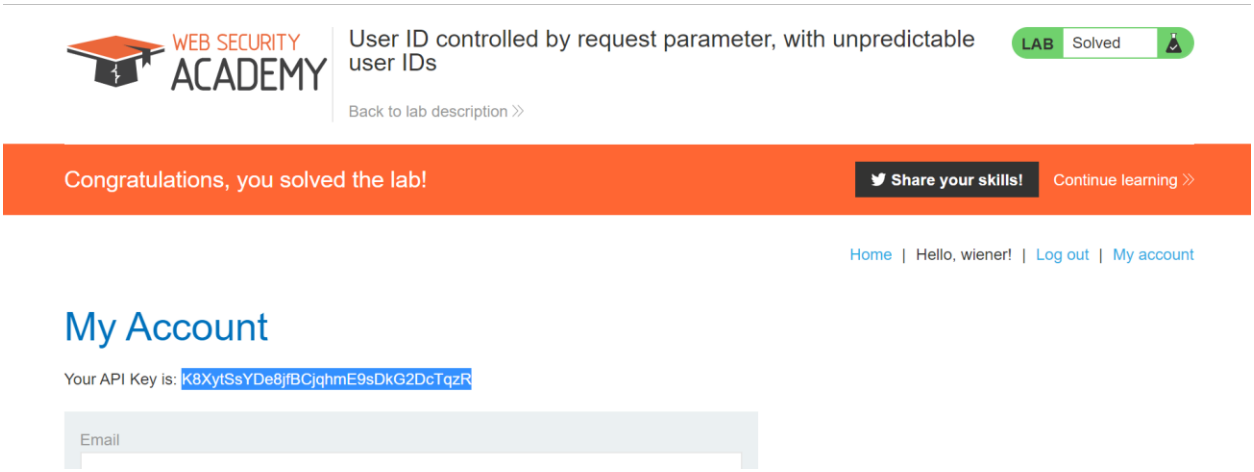
Click on carlos and observe that the URL contains his user ID.

Make a note of the user ID.

Log in using the supplied credentials and access "My Account".

Change the "id" parameter to the saved user ID.

Retrieve and submit the API key.



Lab9: User ID controlled by request parameter with data leakage in redirect

Description: 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 access you own account using wiener:peter.

Testing procedure and snapshot:

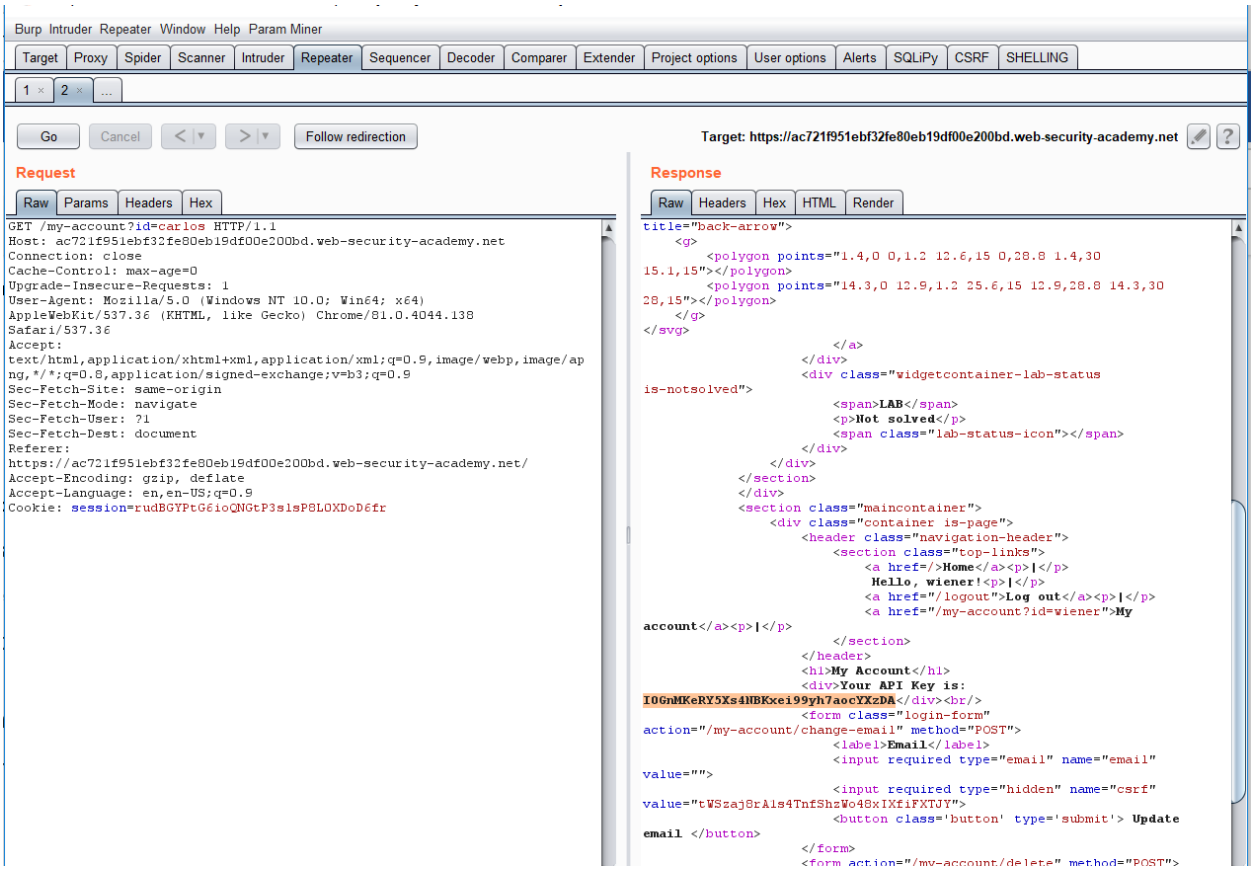
Log in using the supplied credentials and access "My Account".

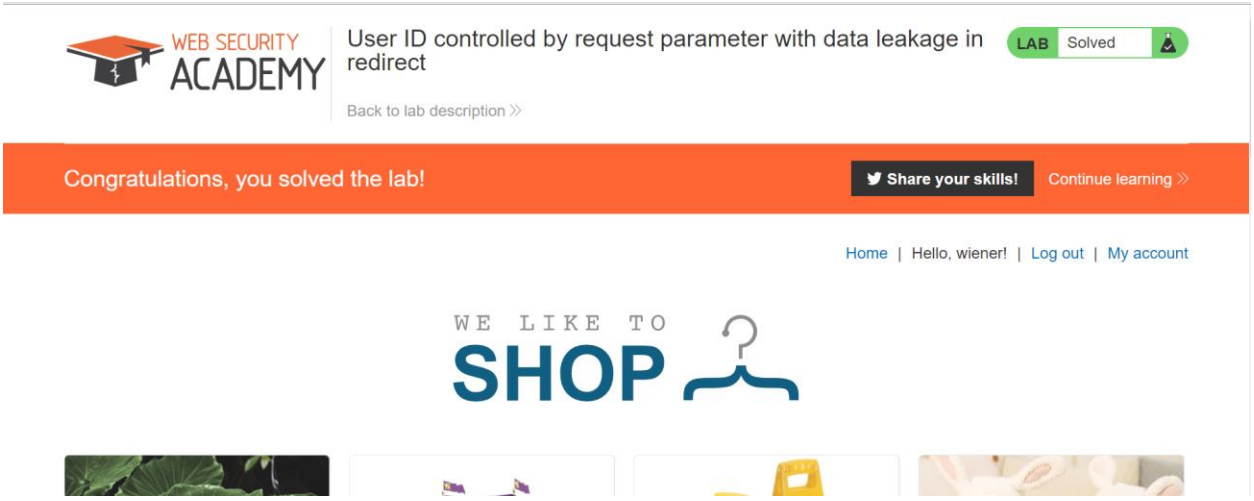
Send the request to Burp Repeater.

Change the "id" parameter to carlos.

Observe that although the response is now redirecting you to the homepage, it has a body containing the API key belonging to carlos.

Submit the API key.





Lab10: User ID controlled by request parameter with password disclosure

Description: This lab has an "Account Details" page for users that contains their existing password prefilled in a masked input.

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

You can access your own account using wiener:peter.

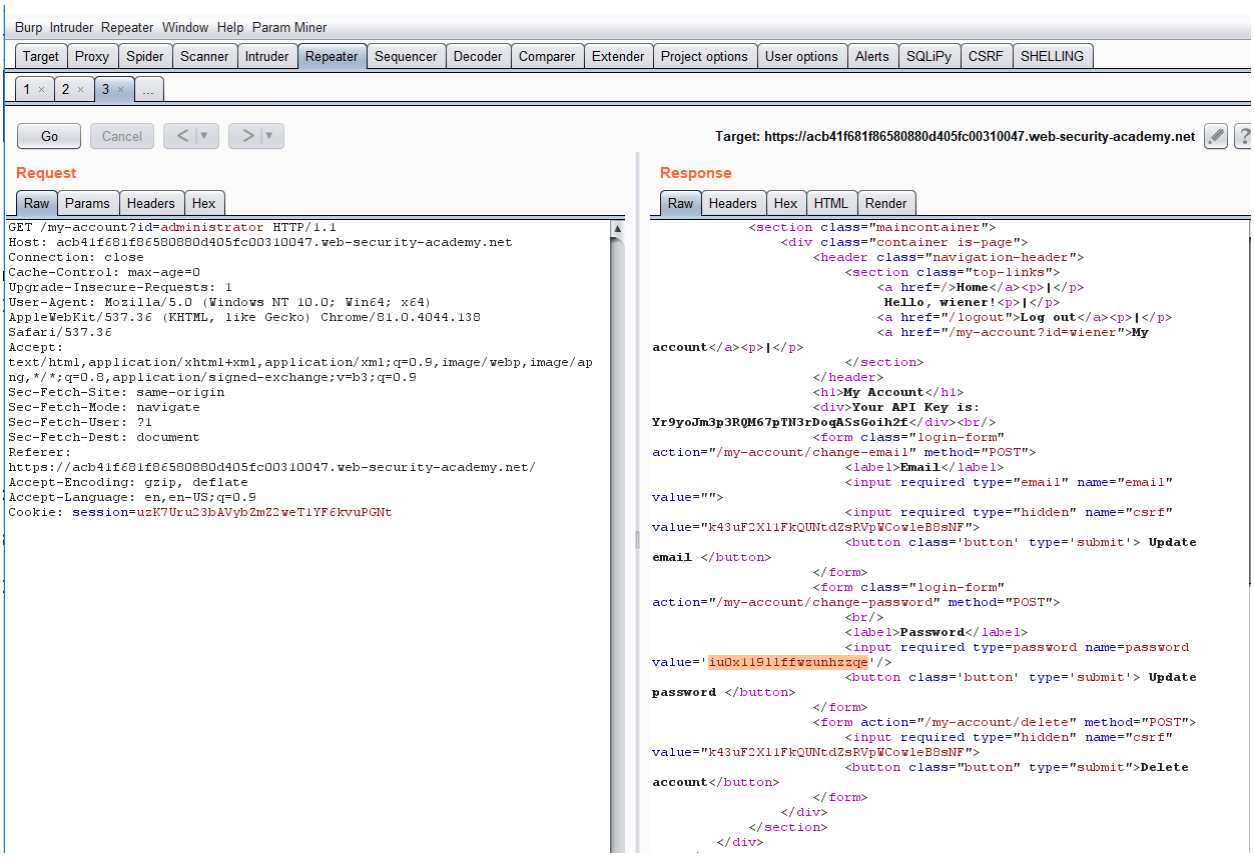
Testing procedure and snapshot:

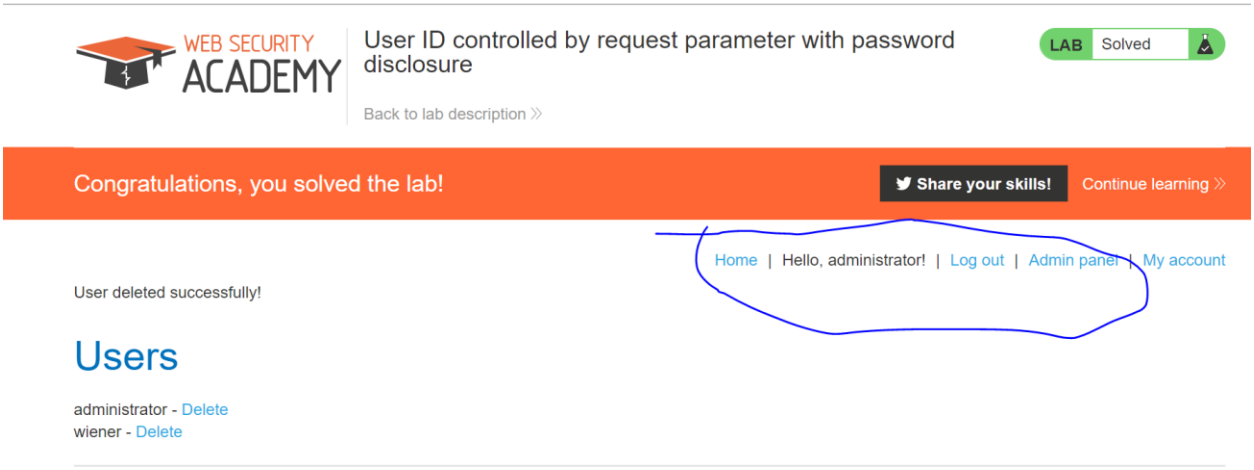
Log in using the supplied credentials and access My Account.

Change the "id" parameter in the URL to "administrator".

View the response in Burp and observe that it contains the administrator's password.

Log in to the administrator account and delete carlos.





Lab11: Insecure direct object references

Description: 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.

Testing procedure and snapshot:

Select the "Live chat" tab.

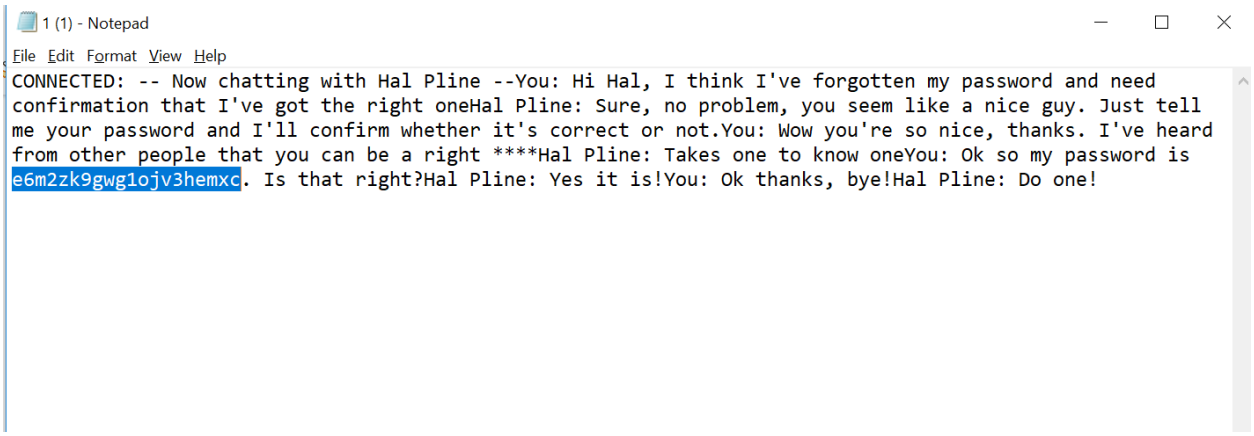
Send a message and then select "View transcript".

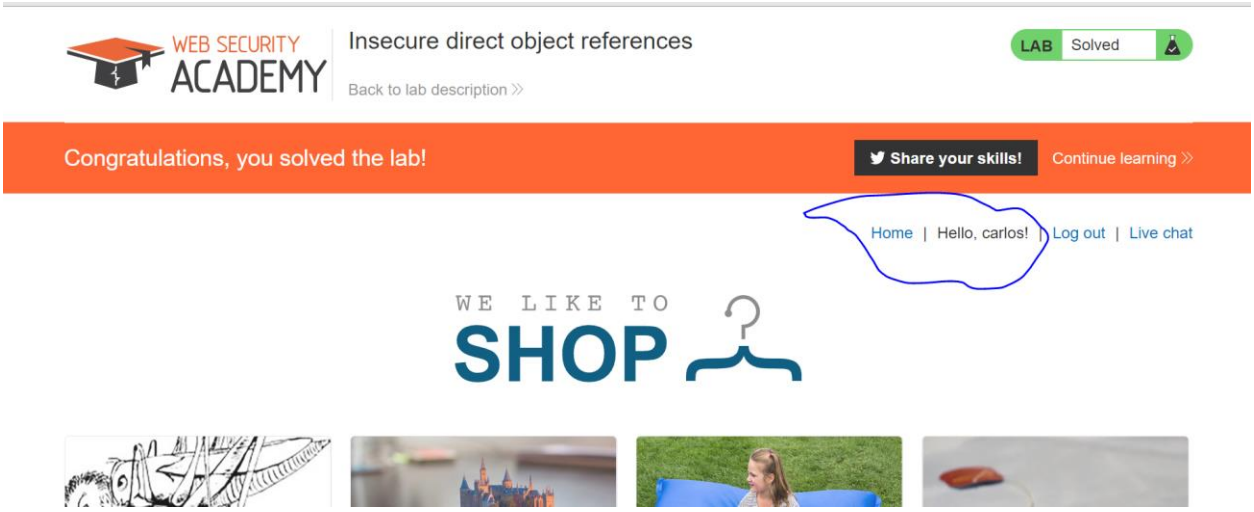
Review the URL and observe that the transcripts are text files assigned a filename containing an incrementing number.

Change the filename to 1.txt and review the text.

You will notice a password within the chat transcript.

Return to the main lab page and log in using the stolen credentials.





Lab12: Multi-step process with no access control on one step

Description: 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 administrator:admin.

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

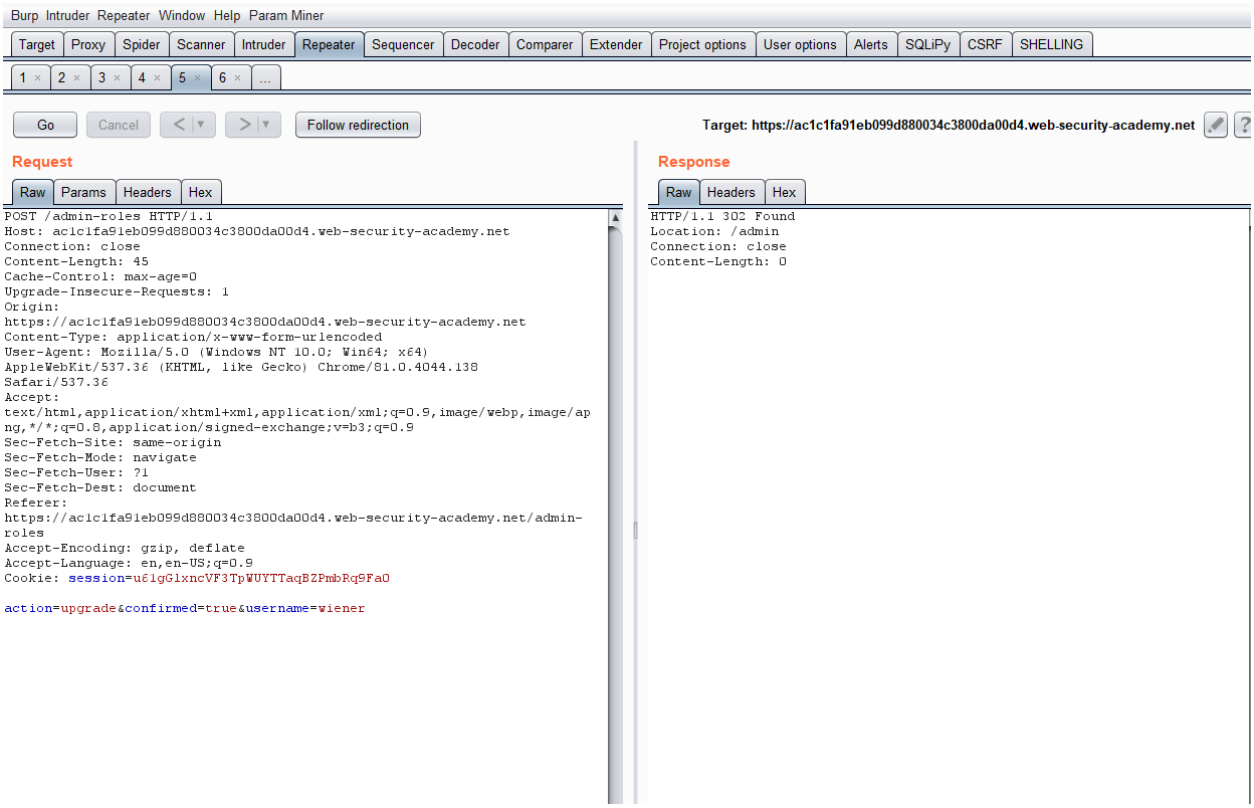
Testing procedure and snapshot:

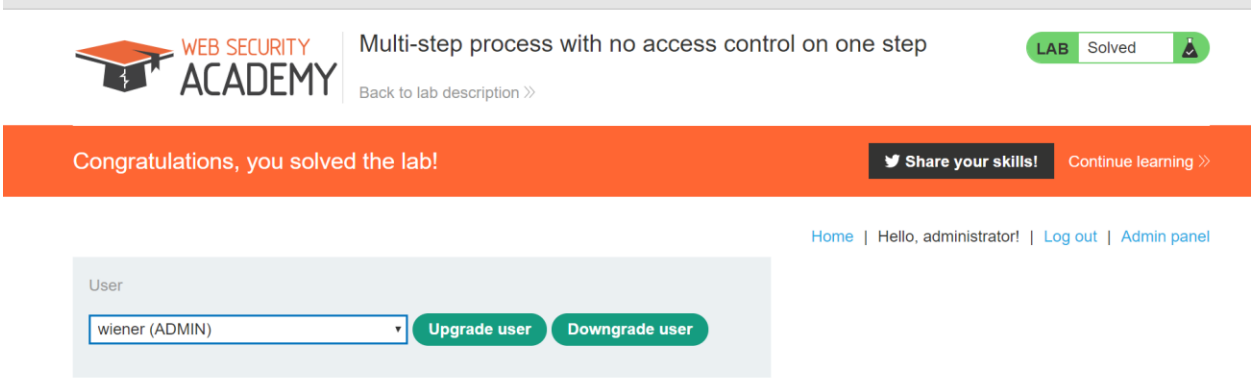
Log in using the admin credentials.

Browse to the admin panel, promote carlos, and send the confirmation HTTP request to Burp Repeater.

Open a private/incognito browser window, and log in with the non-admin credentials.

Copy the non-admin user's session cookie into the existing Repeater request, change the username to yours, and replay it.





Lab13: Referer-based access control

Description: 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 administrator:admin.

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

Testing procedure and snapshot:

Log in using the admin credentials.

Browse to the admin panel, promote carlos, and send the HTTP request to Burp Repeater.

Open a private/incognito browser window, and log in with the non-admin credentials.

Browse to /admin-roles?username=carlos&action=upgrade and observe that the request is treated as unauthorized due to the absent Referer header.

Copy the non-admin user's session cookie into the existing Burp Repeater request, change the username to yours, and replay it.

Burp Intruder Repeater Window Help Param Miner

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts | SQLiPy | CSRF | SHELLING

1 × 2 × 3 × 4 × 5 × 6 × 7 × 8 × ...

Go | Cancel | < | > | Follow redirection

Target: https://ac5b1faf1e81948e80644c660044002f.web-security-academy.net

Request

Raw | Params | Headers | Hex

```
GET /admin-roles?username=wiener&action=upgrade HTTP/1.1
Host: ac5b1faf1e81948e80644c660044002f.web-security-academy.net
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://ac5b1faf1e81948e80644c660044002f.web-security-academy.net/admin
Accept-Encoding: gzip, deflate
Accept-Language: en,en-US;q=0.9
Cookie: session=y1IQJo8FMMG6Y6IeeUHAWq7G1tC9qHaC
```

Response

Raw | Headers | Hex

```
HTTP/1.1 302 Found
Location: /admin
Connection: close
Content-Length: 0
```

Burp Intruder Repeater Window Help Param Miner

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts | SQLiPy | CSRF | SHELLING

1 × 2 × 3 × 4 × 5 × 6 × 7 × 8 × ...

Go | Cancel | < | > | Follow redirection

Target: https://ac5b1faf1e81948e80644c660044002f.web-security-academy.net

Request

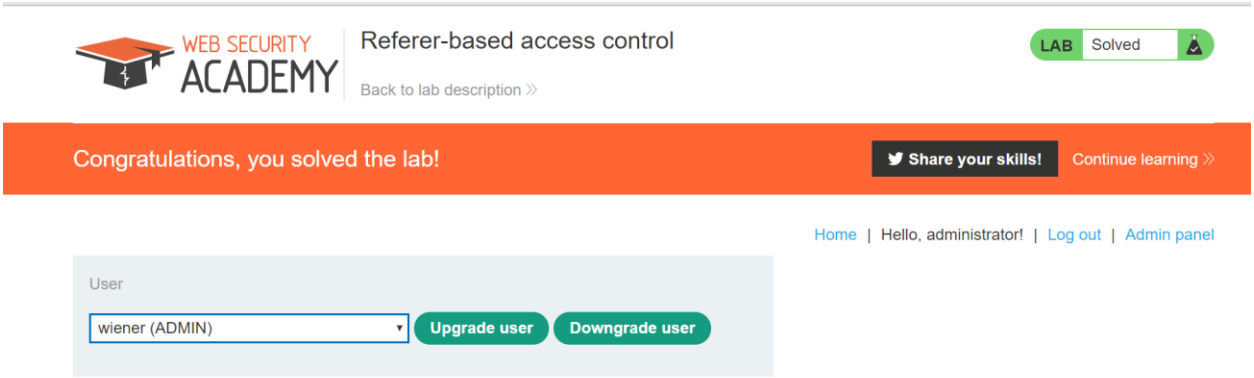
Raw | Params | Headers | Hex

```
GET /admin-roles?username=wiener&action=upgrade HTTP/1.1
Host: ac5b1faf1e81948e80644c660044002f.web-security-academy.net
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://ac5b1faf1e81948e80644c660044002f.web-security-academy.net/admin
Accept-Encoding: gzip, deflate
Accept-Language: en,en-US;q=0.9
Cookie: session=y1IQJo8FMMG6Y6IeeUHAWq7G1tC9qHaC
```

Response

Raw | Headers | Hex

```
HTTP/1.1 302 Found
Location: /admin
Connection: close
Content-Length: 0
```



How to prevent access control vulnerabilities?

Access control vulnerabilities can generally be prevented by taking a defense-in-depth approach and applying the following principles:

- Never rely on obfuscation alone for access control.
- Unless a resource is intended to be publicly accessible, deny access by default.
- Wherever possible, use a single application-wide mechanism for enforcing access controls.
- At the code level, make it mandatory for developers to declare the access that is allowed for each resource, and deny access by default.
- Thoroughly audit and test access controls to ensure they are working as designed.