

# **DC604 Web Application Security Workshop**

## **Featuring PortSwigger Academy Labs**

### **Broken Access Control Edition**

#### **GUIDE FORMAT NOTES**

In this workshop guide document, excluding complete URL links, input that is meant to be typed into a console or input field inside the web browser will be bolded and colored red (**like this**).

#### **Pre-Workshop Setup**

Please complete these steps before the workshop begins:

1. Download and install Burp Suite Community Edition from the following URL:  
<https://portswigger.net/burp/releases>
2. Create a user account at <https://portswigger.net/>.

## **Part 0**

### **Objective – Burp Suite Startup and Setup Lab Environment**

#### **Step 1 – Open the Burp Suite application**

When the following windows appear, click on the buttons outlined in red in the screenshots:

ⓘ Welcome to Burp Suite Community Edition. Use the options below to create or open a project.

*Note: Disk-based projects are only supported on Burp Suite Professional.*



☒ **Temporary project**

☐ **New project on disk**

Name:

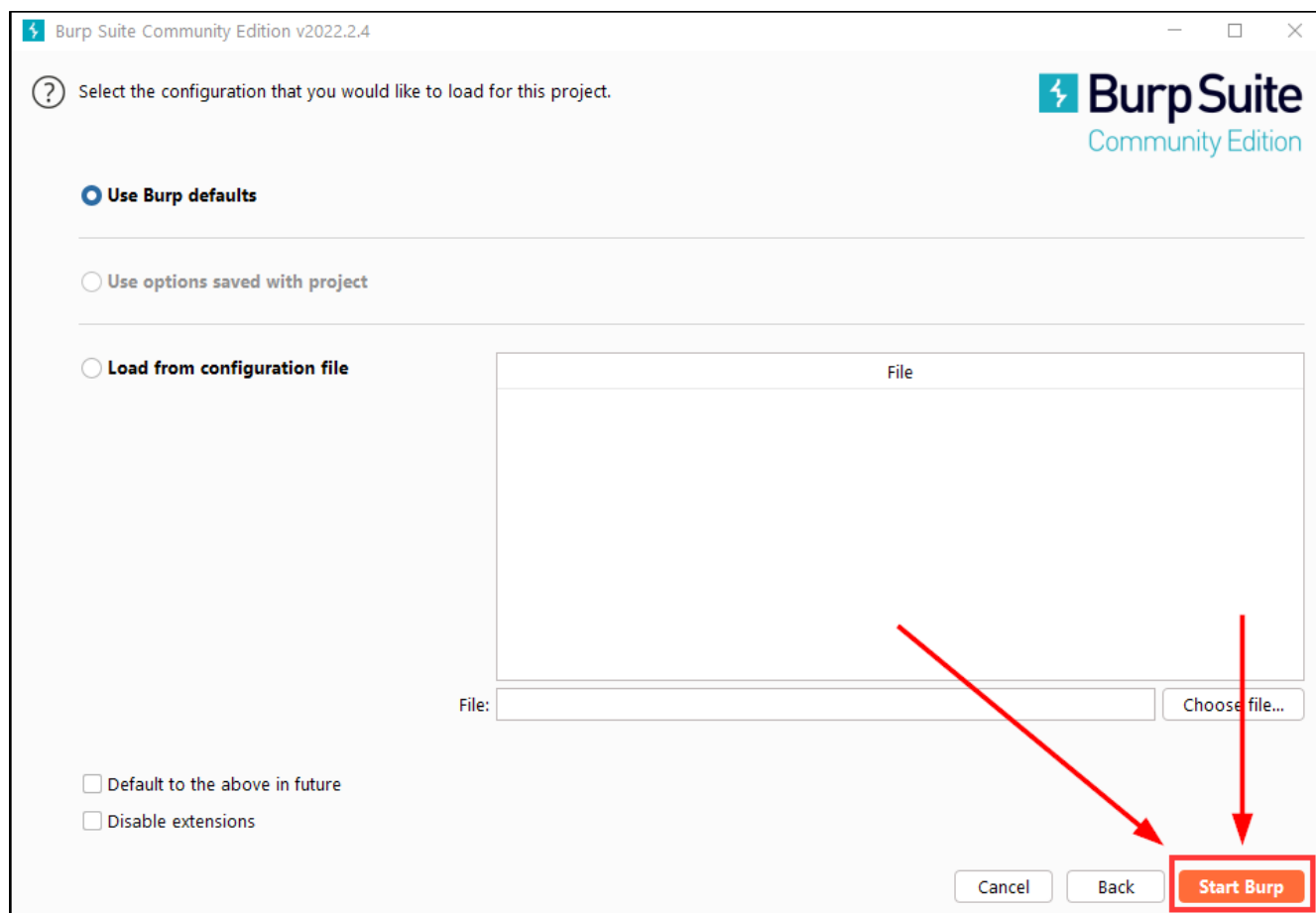
File:

☐ **Open existing project**

Name	File

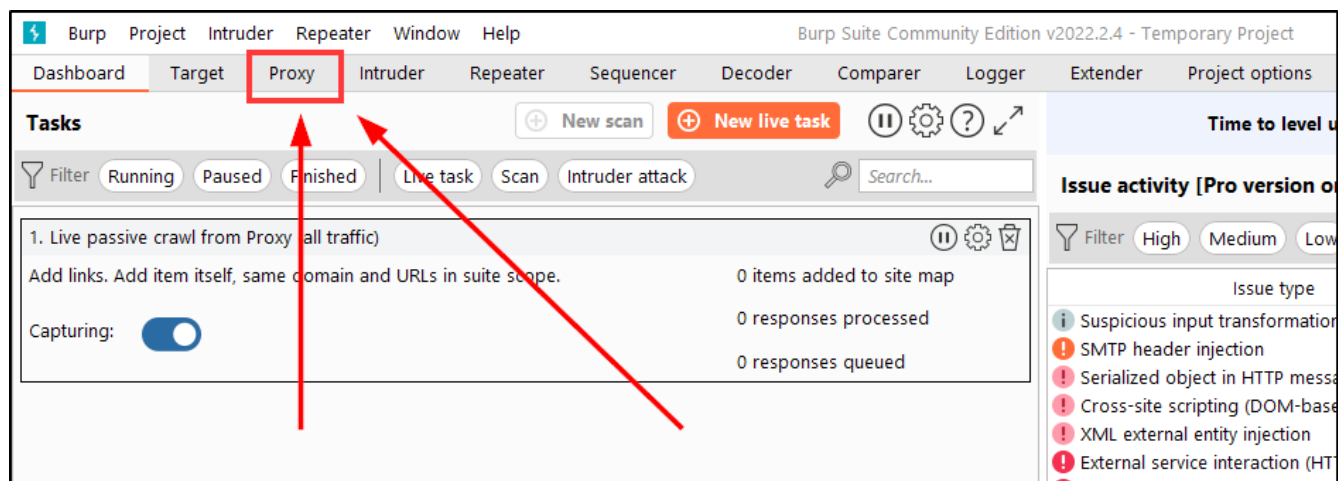
File:

☒ Pause Automated Tasks

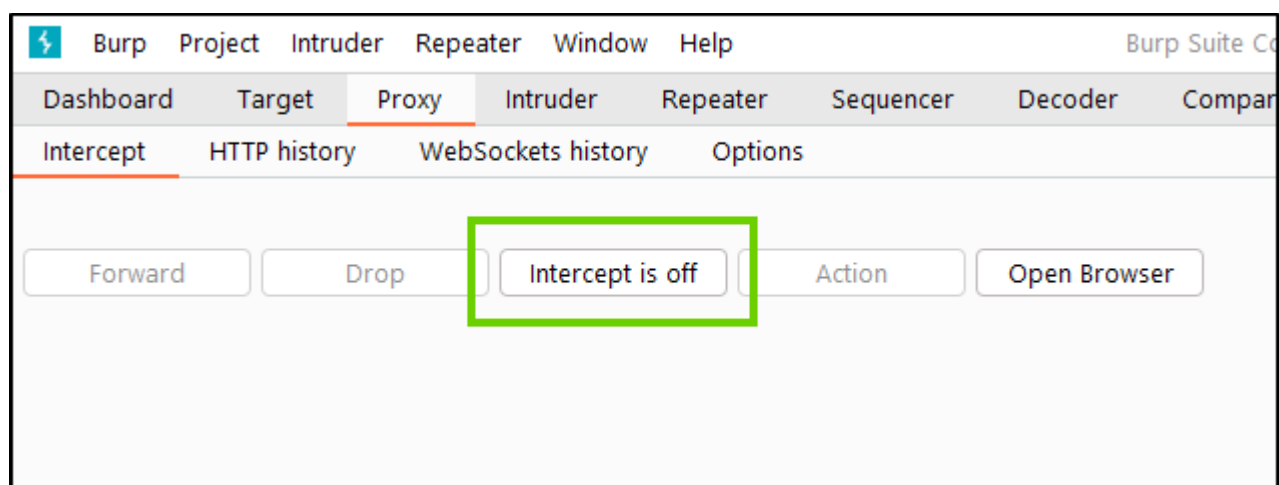
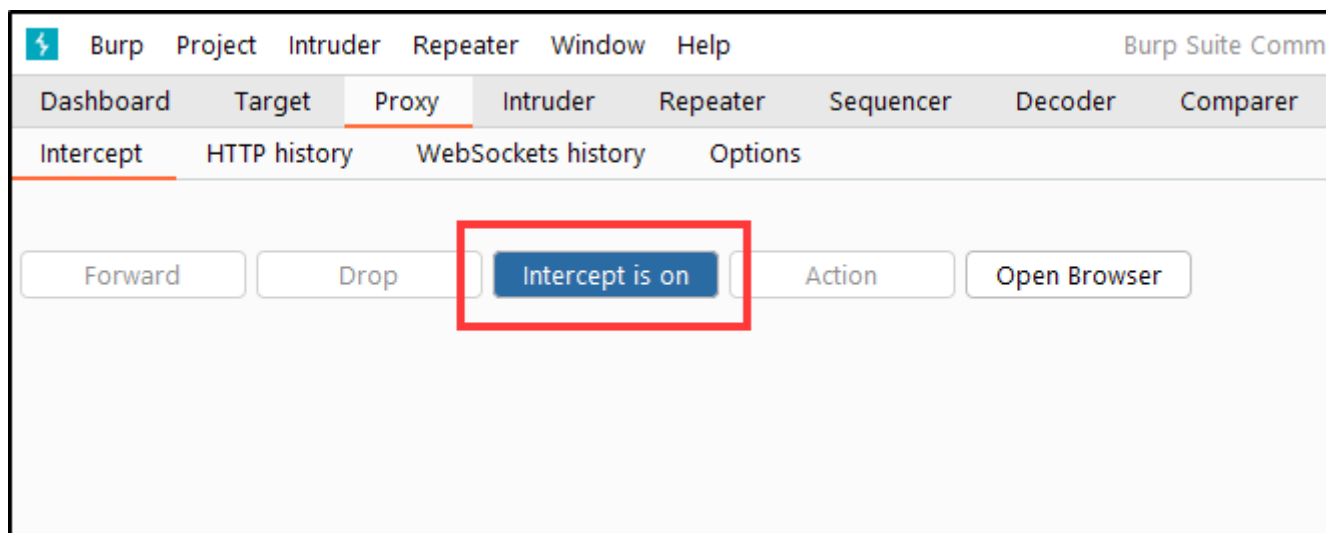


## Step 2 – Turn off Proxy Intercept

Click on the **Proxy** tab located in the top portion of the Burp interface, between the **Target** and **Intruder** tabs (see screenshot below). NOTE: Newer version of BurpSuite start each session with Intercept already turned off.



In the resulting tab, click on the blue **Intercept is on** button, so that it toggles to a white **Intercept is off** button (see the following two screenshots):

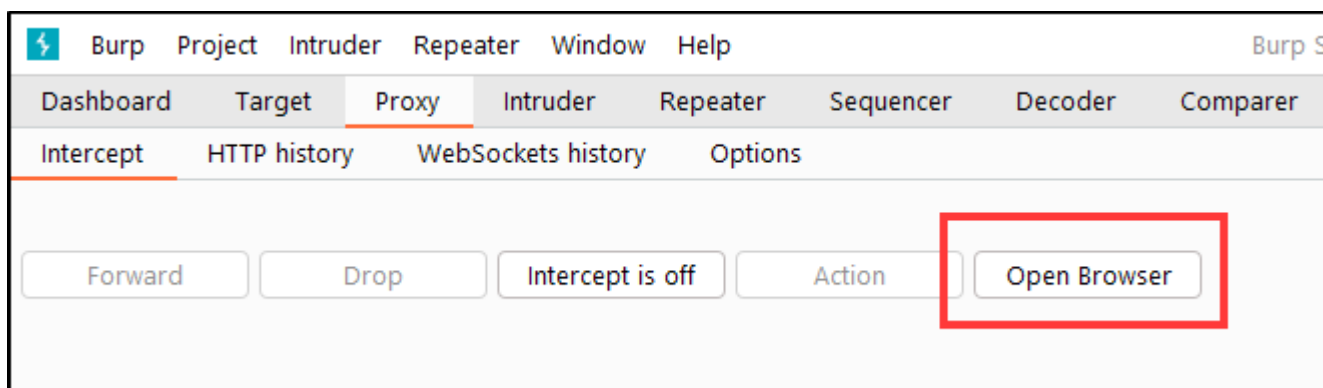


## CONTEXT

If Intercept is turned on in Burp Suite, each web request will need to be manually reviewed by us before being passed from our web browser to the lab website, and vice versa.

### Step 3 – Open the Burp Suite Browser

Staying in the **Proxy** → **Intercept** tab, click on the **Open Browser** button to launch the Burp Suite Chromium web browser (see screenshot below):



**Step 4** – In the Burp Suite Web Browser Login to your Portswigger Account

In the newly-opened Burp Suite web browser, login to your account at the following URL:

<https://portswigger.net/users>

Now it's time to start working through the interactive labs!

## Part 1

**Objective – Solve the Unprotected admin functionality lab**

**Step 1** – While logged into the website, navigate to the following webpage:

<https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality>

**Step 2** - Review the lab instructions:

- Access the lab's unprotected admin panel
- Delete the user **carlos**

**Step 3** – Right-click the green **Access the lab** button to open the lab web app in a new browser tab.

**Step 4** – In the URL address bar for the newly opened web app, add the following **to the end of the address**:

**/robots.txt**

Note that there is another URL endpoint indicated in the output of **robots.txt**. See screenshot below (URL address and exposed admin panel URL outlined in red):

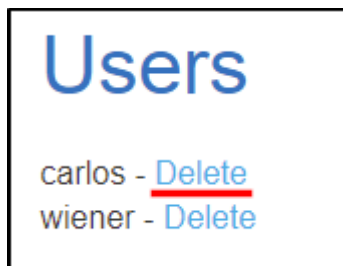


## CONTEXT

The robots.txt file is a way of preventing certain pages or portions of a website from appearing in search engine results. This can be done as an effort to prevent data on sensitive pages from appearing in search results, but if handled the wrong way, it can expose sensitive areas of the web app to potential attackers.

**Step 5** – Navigate to the **/administrator-panel** endpoint in the lab app

Once there, click on the **Delete** link located next to the **carlos** text to complete the lab. See screenshot below (relevant link underlined in red)



## LAB TAKEAWAYS

The vulnerability in the lab was that the admin panel was exposed to normal users, and the exact location of the admin panel was exposed in the robots.txt file. The following are ways we can prevent this kind of vulnerability from being exploited:

- ensure that all admin endpoints are only accessible by user sessions that include cookies with sufficient privileges (admin level) to access them.
- One way to deal with robots.txt potentially exposing sensitive information is to simply configure it to disallow any part of the website from being indexed.

Let's move on to the next lab!

## Part 2

**Objective – Solve the Unprotected admin functionality with unpredictable URL lab**

**Step 1** – Navigate to the following URL:

<https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality-with-unpredictable-url>

**Step 2** – Review the Lab Instructions:

- There is an unprotected admin panel in the lab application
- The location of the admin panel is disclosed somewhere in the application
- Delete the user **carlos** to complete the lab

**Step 3** – Click the green **Access the lab** button

**Step 4** – In the lab app that appears in the new browser tab, right-click and choose **View page source** (alternately, we can use the **CTRL + U** keyboard shortcut)

**Step 5** – In the page source window, note that on line 46, there is a reference to an admin panel endpoint. See screenshot below (admin panel endpoint underlined in red).

```
42 var isAdmin = false;
43 if (isAdmin) {
44     var topLinksTag = document.getElementsByClassName("top-links")[0];
45     var adminPanelTag = document.createElement('a');
46     adminPanelTag.setAttribute('href', '/admin-XXXXXXXXXX');
47     adminPanelTag.innerText = 'Admin panel';
48     topLinksTag.append(adminPanelTag);
```

### CONTEXT

This lab web app includes code that randomly assigns the endpoint that leads to the admin panel, but the exact location is exposed in the page source, which is easily accessible.

**Step 6** – Access the discovered admin panel endpoint, and solve the lab

Once at the admin panel page, click on the **Delete** link located next to the **carlos** text, like in the previous lab.

## LAB TAKEAWAYS

This lab's vulnerability is once again the presence of an unprotected admin panel. The exact endpoint of the panel is slightly randomized, which adds a layer of security, but the implementation of randomizing the admin panel endpoint exposes its location in the source code. The following remediations could be applied to the application to resolve these vulnerabilities:

- Admin panels or other admin functions should never be accessible by regular users, so any pages with admin functions should not be accessible unless the user-session (cookie) is that of an admin user. The app could be modified to include access control lists which restrict access to certain endpoints depending on the type of user-session associated with the user's session token.
- If there is JavaScript that creates sensitive information (such as the location of an admin panel), that code should not be on a page or in a file that is accessible by regular users.

On to the next lab!

## Part 3

### Objective – Complete the User role controlled by request parameter lab

#### Step 1 – Navigate to the lab description webpage

<https://portswigger.net/web-security/access-control/lab-user-role-controlled-by-request-parameter>

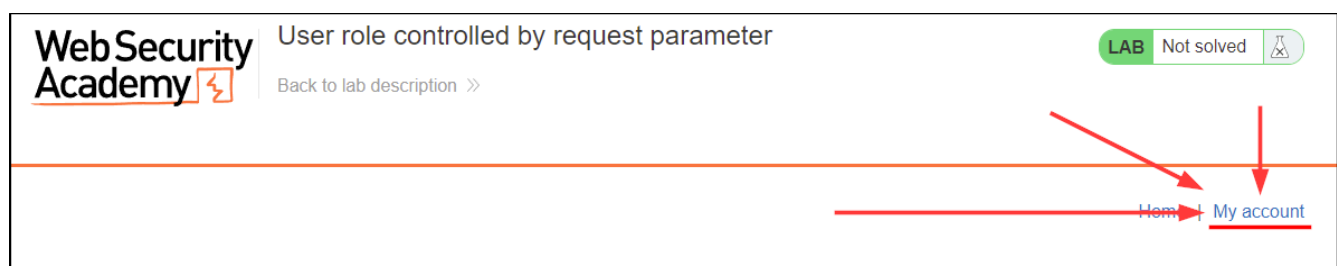
#### Step 2 – Review the lab instructions

- The lab app has an admin panel at the **/admin** endpoint
- We must delete the user **carlos** to complete the lab
- We are provided a working set of credentials: **wiener:peter** (username **wiener**, password **peter**)

#### Step 3 – Click on the green **Access the lab** button

#### Step 4 – Click on the **My account** link located at the top-right portion of the webpage

See screenshot below (relevant portion underlined in red)





**Step 5** – Use the credentials **wiener** (username) and **peter** (password) to login

**Step 6** – Attempt to access the **/admin** endpoint

Notice that the app won't let us access the page.

**Step 7** – Observe how the login POST request was processed

In Burp Suite, click on the **Proxy** → **HTTP History** tab, then click on the **Method** column, then scroll to the bottom of the list and click on the POST request to the **/login** URL. See screenshot below (relevant fields underlined in red):

Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer
Intercept	HTTP history	WebSockets history	Options				
Filter: Hiding CSS, image and general binary content							
#	Host	Meth... ^	URL	Params	Edited		
146	https://play.google.com	POST	/log?format=json&hasfast=true&auth...	✓			
148	https://play.google.com	POST	/log?format=json&hasfast=true&auth...	✓			
150	https://play.google.com	POST	/log?format=json&hasfast=true&auth...	✓			
155	https://play.google.com	POST	/log?format=json&hasfast=true&auth...	✓			
156	https://play.google.com	POST	/log?format=json&hasfast=true&auth...	✓			
158	https://play.google.com	POST	/log?format=json&hasfast=true&auth...	✓			
159	https://www.youtube.com	POST	/youtubei/v1/log_event?alt=json&key=...	✓			
179	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=450766242&...	✓			
181	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=450766242&...	✓			
201	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=1694163777...	✓			
202	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=1694163777...	✓			
279	https://0ac000ed04d7a3c6c0bd...	POST	/login	✓			

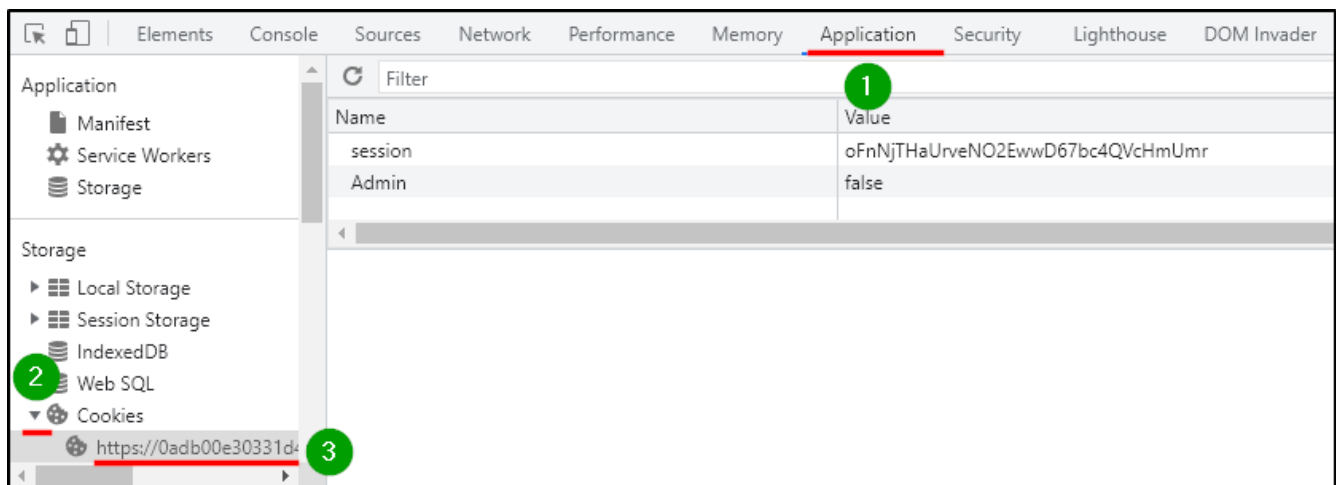
In the Response window, we see that the server assigns us a cookie named **Admin** with a value of **false** (see screenshot below).

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Location: /my-account
3 Set-Cookie: Admin=false; Secure; HttpOnly
4 Set-Cookie: session=
  37IKf2yd7rckTyx9tprDzBHx9qpjnn5h; Secure; HttpOnly;
  SameSite=None
5 Connection: close
6 Content-Length: 0
```

This is good for us, since we can easily edit the values of any cookies in our web browser.

### Step 8 – Edit the value of the **Admin** cookie

In the web browser, use the keyboard shortcut **Ctrl + Shift + I** to open the Developer Tools window, then click on the **Application** tab, then on the left-hand side of the window, click on the arrow beside the **Cookies** entry, then click on the https URL under that entry. See screenshot below (relevant portions outlined in red and numbered)



Note that name and value of the cookies are modifiable in this window. Double click on the cell with the **false** value and replace it with the value **true**.

### Step 9 – Navigate to the **/admin** endpoint

### Step 10 – Delete the **carlos** user in the manner identical to the previous labs

## LAB TAKEAWAYS

This lab allowed access an admin page based on settings contained within a cookie assigned to the user upon login. This by itself isn't a issue. However, since the value of the cookie that allows access to the admin page (Admin:true) is easily determined and cookie values are user controlled, this leads to exploitation and broken access control. The following remediations could be implemented to strengthen the security of the app:

- The cookie values assigned to regular user sessions should be both different from those assigned to admin user sessions, and also random enough so that they cannot be easily determined through casual observation or random guessing.

## Part 4

### Objective – Complete the User role can be modified in user profile lab

**Step 1** – Navigate to the following URL:

<https://portswigger.net/web-security/access-control/lab-user-role-can-be-modified-in-user-profile>

**Step 2** – Review the lab instructions

- There is an admin panel at the /admin endpoint
- The panel can only be accessed by user sessions that include a **roleid** value of **2**
- We are provided with credentials for the web app: **wiener:peter**
- The goal of the lab is to access the admin panel and delete the carlos user

**Step 3** – Click on the green Access the lab button to start the lab environment

**Step 4** – In the lab web app, login to the application by following the **My Account** link located in the same location as the last lab, then providing the lab-provided credentials, **wiener** (username) and **peter** (password).

**Step 5** – In the resulting **/my-account** endpoint, change the email account by entering **test@test.com** in the **Email** field, then click on the green **Update email** button. We will know that it processed when the line under **Your username is: wiener** reads **Your email is: test@test.com**.

**Step 6** – In Burp Suite, click on the **Proxy** → **HTTP history** tab, then sort the results by **Method**, then scroll down to the bottom of the list of POST requests and click on the request to the **/my-account/change-email** URL.

**Step 7** – In the resulting Request and Response windows, we observe two things:

First, in the Request window, we see that we send the information to the lab server is sent in JSON format. See screenshot below (JSON data outlined in red):

```
Request
Pretty Raw Hex
1 POST /my-account/change-email HTTP/1.1
2 Host: 0a880028033feed4c0f1149d000e0050.web-security-academy.net
3 Cookie: session=qHEFB7VFgbp1YwR9vfpXzsOn40nL7b4p
4 Content-Length: 25
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A) Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.53 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: text/plain; charset=UTF-8
10 Accept: */*
11 Origin: https://0a880028033feed4c0f1149d000e0050.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0a880028033feed4c0f1149d000e0050.web-security-academy.net/my-account
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 {
  "email": "test@test.com"
}
```

Second, in the Response window, we see that a JSON entry is returned as a profile with a number of different values. We also see that our email has been updated to reflect the input we sent to the server, and that the parameter that we were told will let us access the admin panel (**roleid**) is included in the data. See screenshot below (JSON data outlined in red, email and roleid values underlined in red).

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Location: /my-account
3 Content-Type: application/json; charset=utf-8
4 Connection: close
5 Content-Length: 117
6
7 {
8   "username": "wiener",
9   "email": "test@test.com",
10  "apikey": "2x7eVQ8HuaAWRDgMqMw96ZZcihtJAgmT",
11  "roleid": 1
12 }
```

We know that solving the lab involves setting our roleid to the value 2, so let's try modifying that as well.

**Step 8** – Resend the POST request to the **/my-account/change-email** endpoint with extra data

In Burp Suite, right-click inside the Request window and select **Send to Repeater**, then click on the Repeater tab (at the top, between the Intruder and Sequencer tabs). Once there, edit the request by adding **`,"roleid":2`** after **`"email": "test@test.com"`**. The data should look like the screenshot below.

```
20 {  
    "email": "test@test.com",  
    "roleid": 2  
}
```

Once ready, click the orange **Send** button (above the Request window header). In the Response window, we should see that the JSON data that is returned includes **"roleid":2**, which will allow access the the **/admin** endpoint. See screenshot below (relevant value underlined in red).

```
7 {  
8   "username": "wiener",  
9   "email": "test@test.com",  
10  "apikey": "5Y3SBNMzRqV4XTuVgubhRPFyyTyNQXNB",  
11  "roleid": 2  
12 }
```

**Step 9** – Access the **/admin** endpoint in the web browser and delete the **carlos** user, completing the lab

## LAB TAKEAWAYS

This lab web app checks access to sensitive endpoints depending on values contained in the user's profile (as a JSON object). This isn't necessarily an insecure way to restrict access to sensitive pages, unfortunately the function that allowed the user to modify their email address also allowed the modification of other parameters (roleid). The following remediations could improve the security of this lab web app:

- The admin panel should be located at an endpoint that is not easily predictable by attackers
- The function that updates user profile emails should not be allowed to modify anything other than the email address

On to the next lab!

## Part 5

**Objective** – Complete the User ID controlled by request parameter lab

**Step 1** – Navigate to the following URL:

<https://portswigger.net/web-security/access-control/lab-user-id-controlled-by-request-parameter>

## Step 2 – Review the lab instructions

- There is a horizontal privilege escalation vulnerability on the user account page
- Utilize the vulnerability to obtain the API key for the user **carlos** and submit it as the solution to the lab
- Credentials for the lab app have been provided (wiener:peter)

## Step 3 – Click the green **Access the lab** button

## Step 4 – Login to the app using the **My Account** link as in previous labs with the **wiener:peter** credentials

## Step 5 – Click on the **My account** link again

Note that at the URL that is indicated at the top includes a URL parameter (**id=wiener**)

## CONTEXT

URL parameters that take values that attackers can easily predict (such as usernames) can be swapped out to potentially access data they're not supposed to see.

## Step 6 – Change the URL parameter in the URL bar from **wiener** to **carlos**

Observe that the **My Account** information now indicates that the username is **carlos**.

## Step 7 – Copy the API key on the page, then click on the orange **Submit solution** button at the top portion of the webpage and paste the value into the field, solving the lab

## LAB TAKEAWAYS

The security vulnerability in this lab app is that there are no access controls for user account pages, and because the app uses a URL parameter as a method of accessing specific user account pages, this allows the enumeration of any user's account information, as long as the user name is known. The following remediation could be performed to improve this app's security:

- Although it is not strictly a bad practice to retrieve user-specific information through URL parameters, the app should perform a check to determine if the user's session cookie matches the username info being retrieved, and deny access if the two usernames do not match

On to the last lab!

## Part 6

### Objective – Complete the Insecure direct object references lab

**Step 1** – Access the lab description page at the following URL:

<https://portswigger.net/web-security/access-control/lab-insecure-direct-object-references>

**Step 2** – Review the lab instructions

- Chat logs are stored on the server's file system, and they can be accessed with static URLs.
- The lab is solved by finding the password for the **carlos** user, and logging into their account

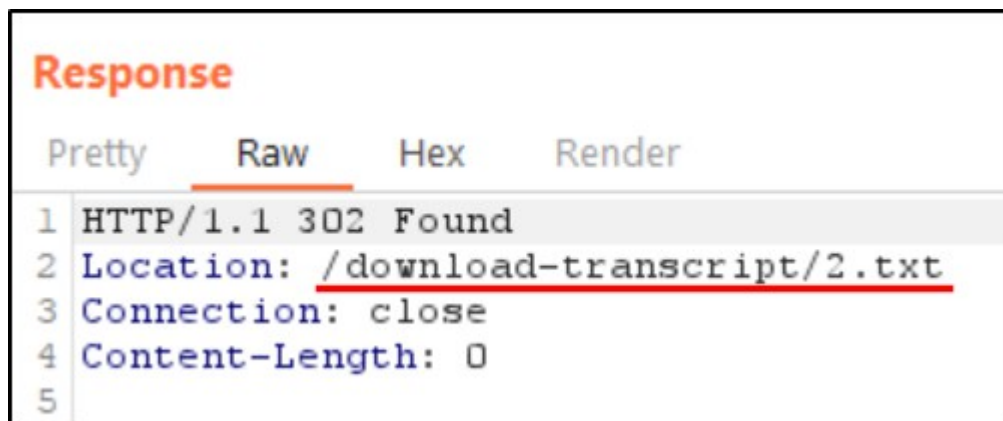
**Step 3** – Click the green **Access the lab** button

**Step 4** – Click on the **Live chat** link located beside the usual **My account** link

**Step 5** – Click on the green **View transcript** button

**Step 6** – In Burp Suite, look at the HTTP request for the **View transcript** button

Click on the **Proxy** → **HTTP history** tabs, then click on the **Method** column, then scroll to the bottom of the list, then click on the POST request to the **/download-transcript** endpoint. In the Response portion of the request, note that the transcript file download URL is exposed. See screenshot below (download endpoint underlined in red).



### CONTEXT

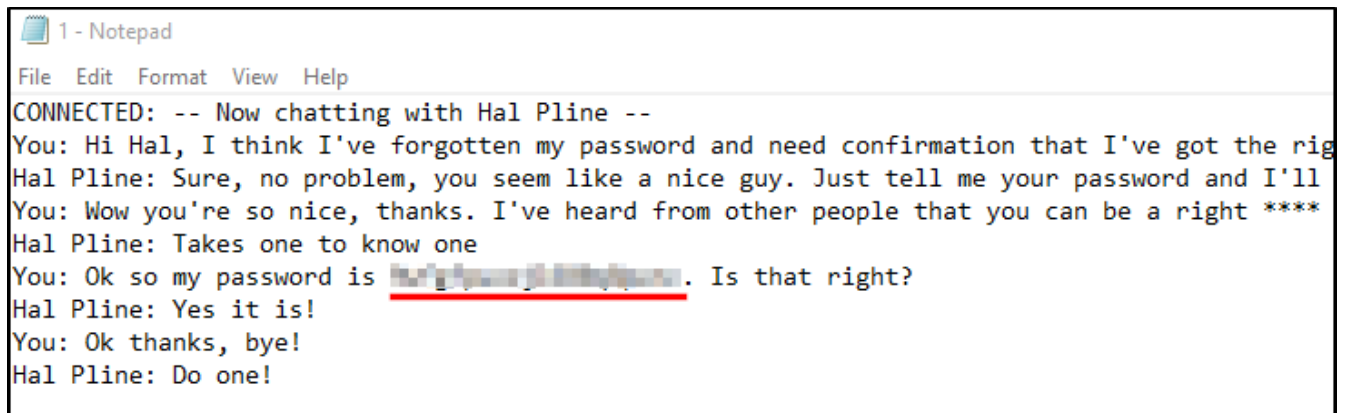
The app appears to be using a predictable file naming system for saving user chat logs. Let's try an earlier number in the order to download other log files.

**Step 7** – Iterate on the file name to download other user's chat logs

In the browser, navigate to the **/download-transcript/1.txt** endpoint in the lab app. The **1.txt** file should download.

**Step 8** – Using Notepad or another text editing program, open the **1.txt** file

The file contains the password for the user carlos. See screenshot below (password underlined in red).



```
1 - Notepad
File Edit Format View Help
CONNECTED: -- Now chatting with Hal Pline --
You: Hi Hal, I think I've forgotten my password and need confirmation that I've got the rig
Hal Pline: Sure, no problem, you seem like a nice guy. Just tell me your password and I'll
You: Wow you're so nice, thanks. I've heard from other people that you can be a right ****
Hal Pline: Takes one to know one
You: Ok so my password is halpline123456789. Is that right?
Hal Pline: Yes it is!
You: Ok thanks, bye!
Hal Pline: Do one!
```

**Step 9** – Login to the app using the username carlos and the captured password as the password to solve the lab

## LAB TAKEAWAYS

This lab app is vulnerable to an IDOR vulnerability where sensitive information is stored in a easily discernible location and takes on a predictable naming scheme.

## CONCLUSION

There are many ways in which login pages can be misconfigured to allow enumeration of valid usernames or brute-force passwords attacks, but with proper security awareness and testing these issues can be resolved. Most modern applications incorporate account lockout mechanisms (by username or IP) to prevent brute-force password attacks, but many older apps still have lack proper protections to prevent brute-force login attacks.

## CONTACT

Contact the instructor for this workshop on twitter at:

<https://twitter.com/theshyhat>

See you next time!