

# **DC604 Web Application Security Workshop**

## **Featuring PortSwigger Academy Labs**

### **Login Page Attack Edition**

#### **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:

Choose file...

☐ **Open existing project**

Name	File

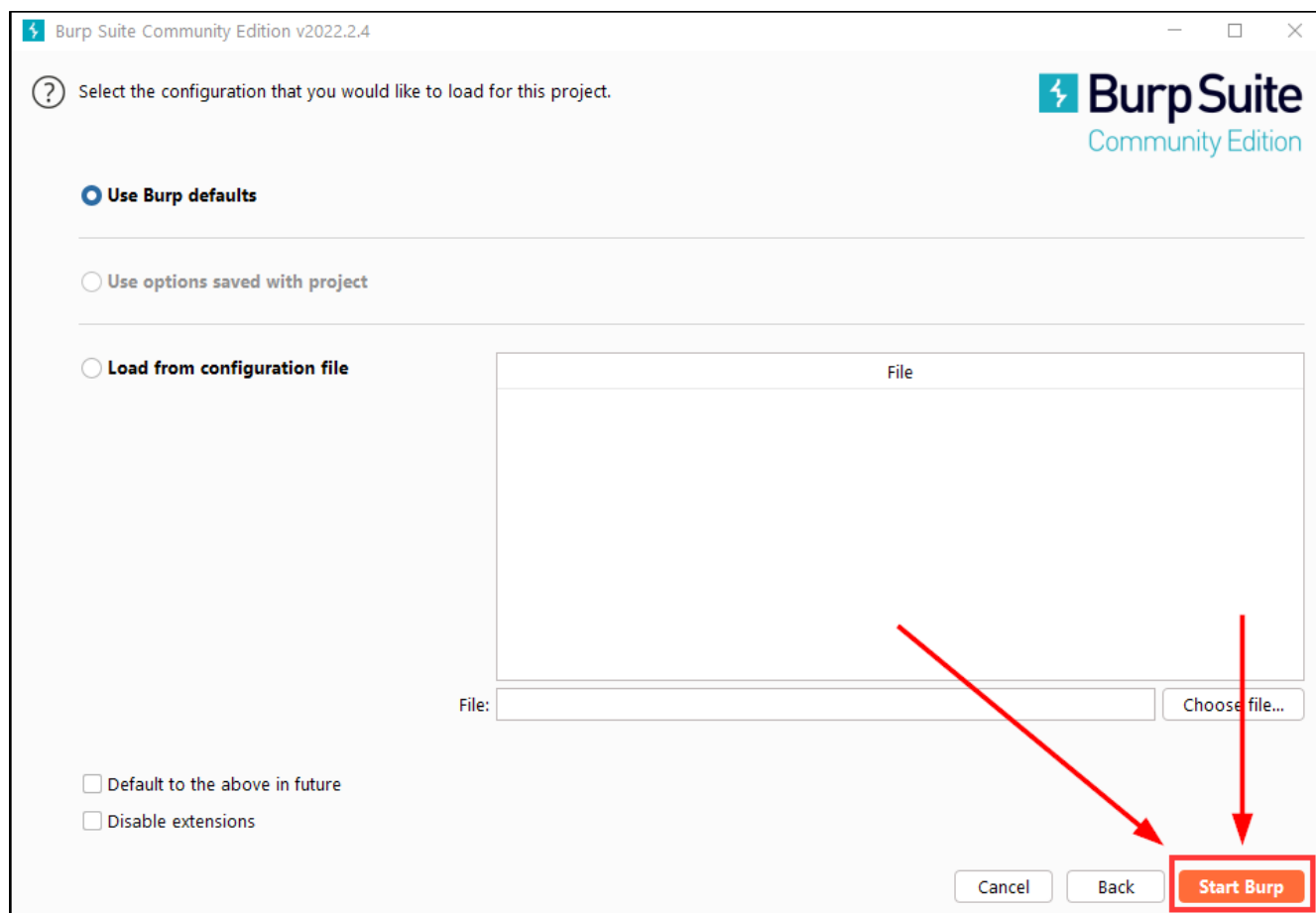
File:

Choose file...

☒ Pause Automated Tasks

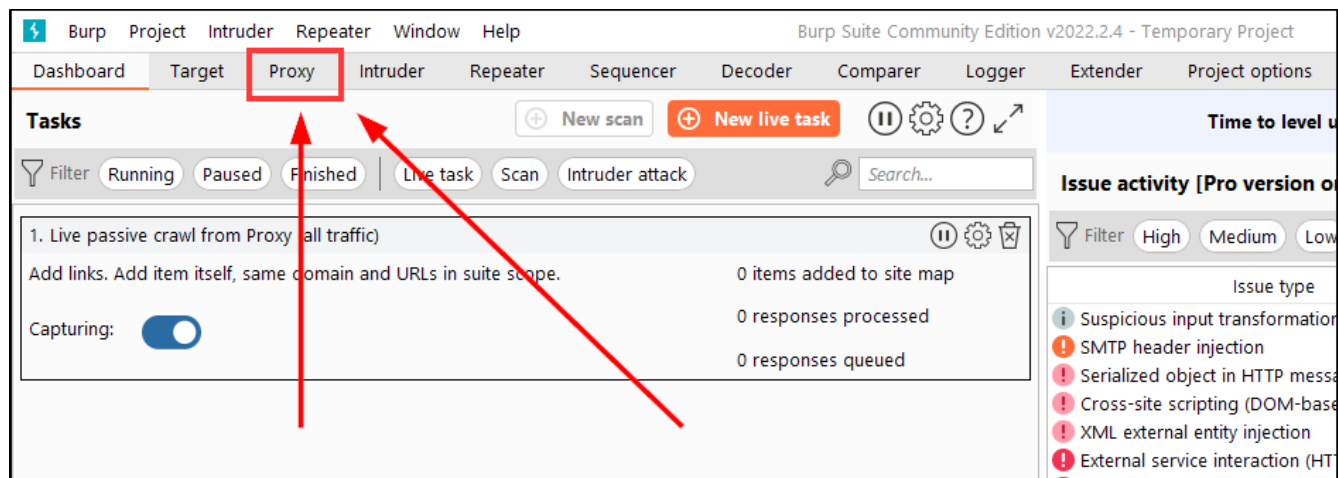
Cancel

Next



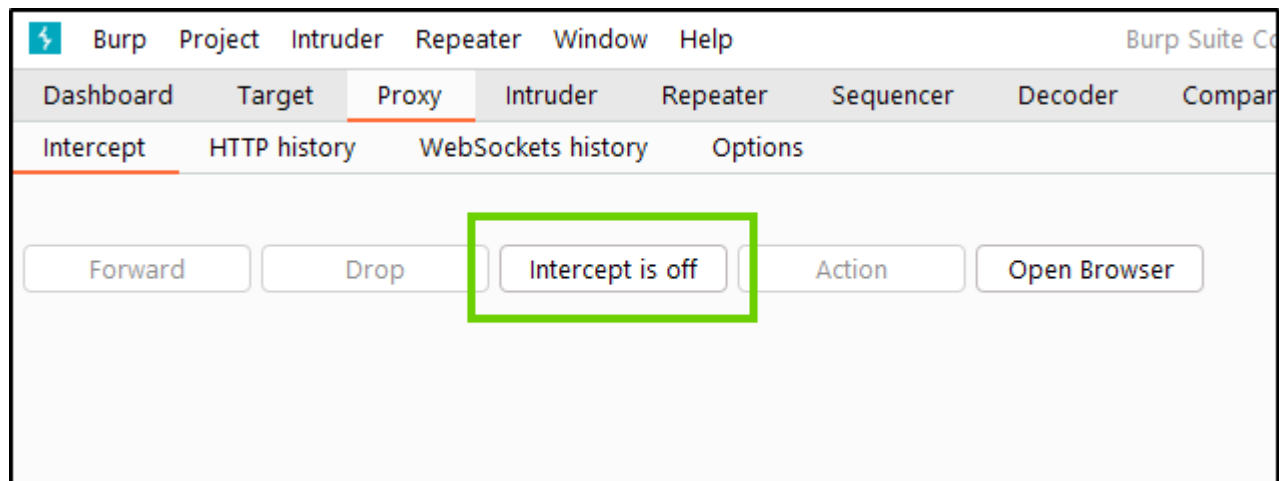
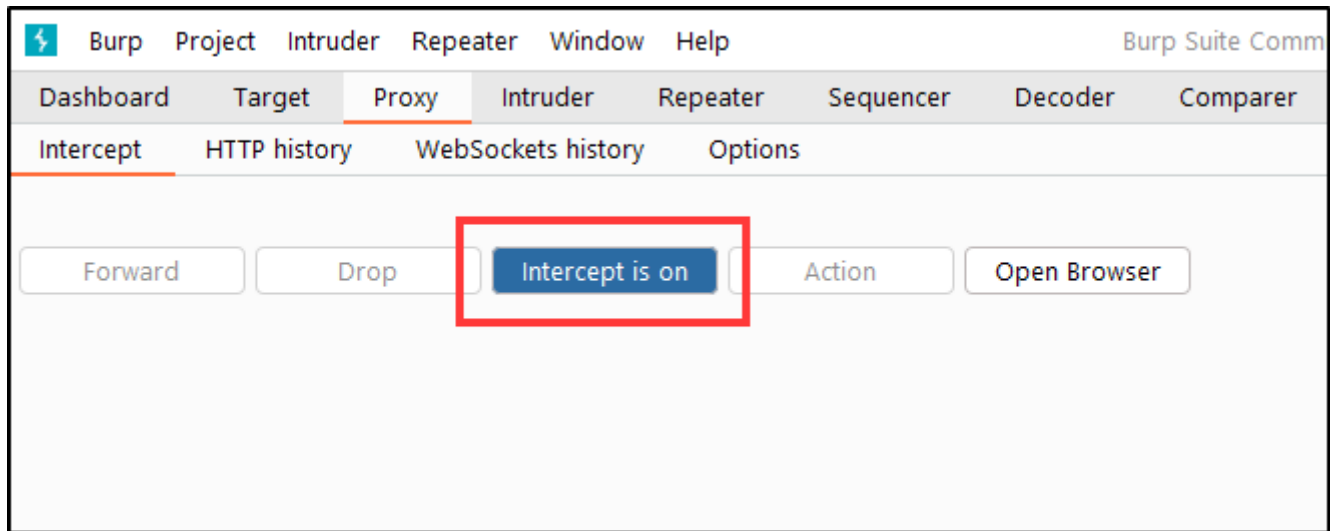
## Step 2 – Turn off Proxy Intercept

Click on the Proxy tab located in the top portion of the Burp interface, between the **Target** and **Intruder** tabs (see screenshot below)



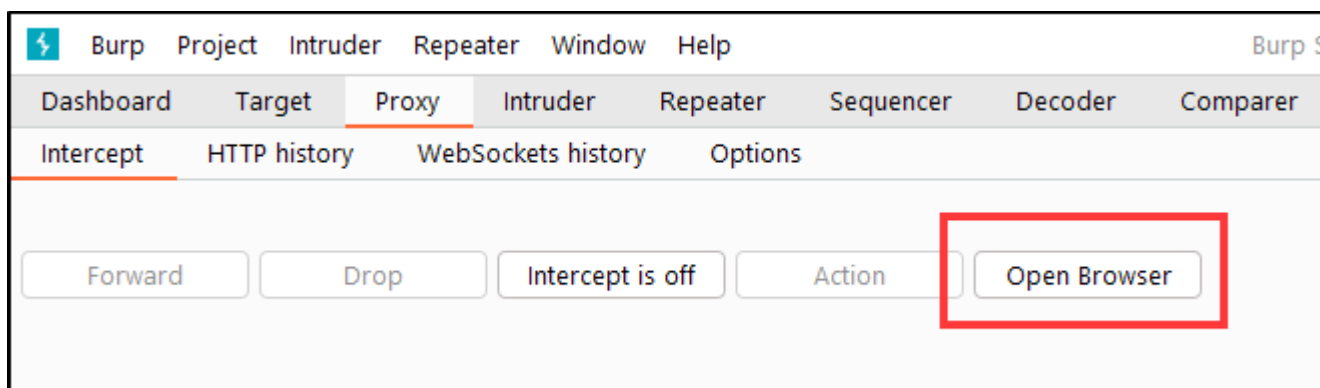
In the resulting tab, click on the blue **Intercept is on** button, so that it toggles to a white **Intercept is**

**off** button (see the following two screenshots):



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>

## Part 1

**Objective – Solve the Username enumeration via different responses lab**

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

<https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-different-responses>

**Step 2 – Right-click the green Access the lab button to open the Lab web app in a new browser tab.**

**Step 3 – Click on the My account link located on the right-hand side of the page.**

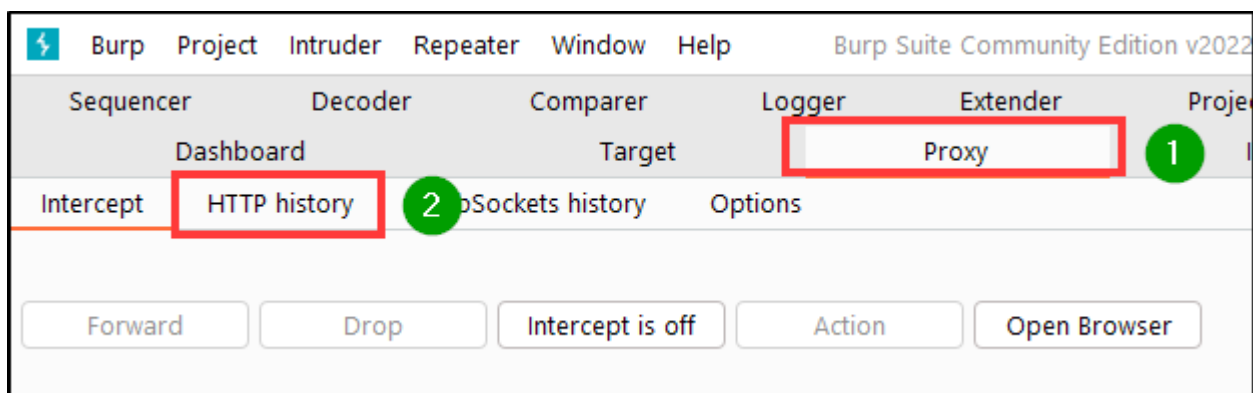
**Step 4 – Supply the login fields with the following credentials**

**Username: testUser**

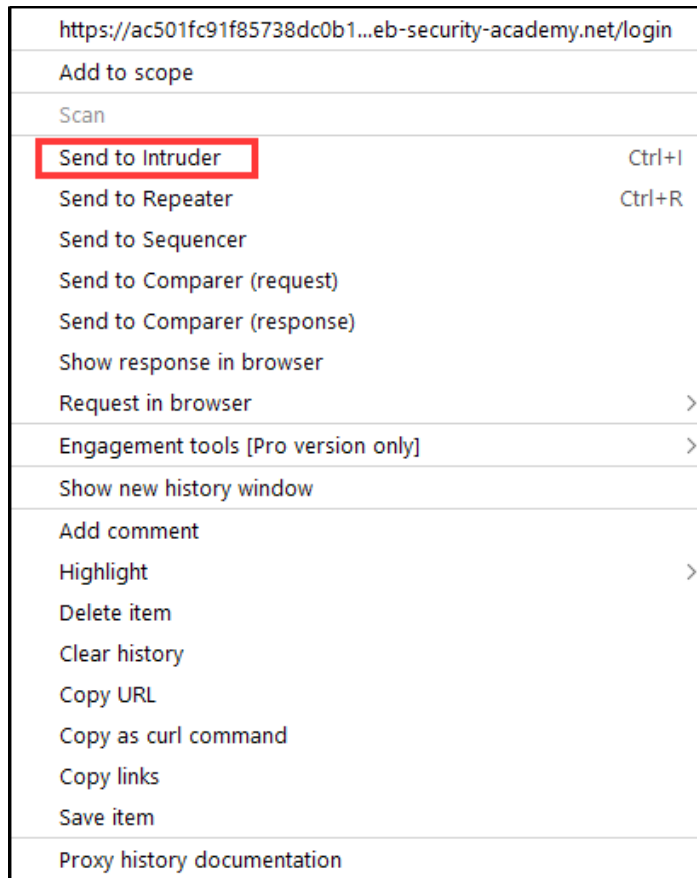
**Password: testPass**

Then click the green **Log in** button

Step 5 – In Burp Suite, click on the **Proxy → HTTP history** tab (see screenshot below):



Step 6 – In the list of requests, click on the Method column to sort the requests by HTTP request type, then scroll to the bottom of the list. Right-click on the request to the **/login** URL, then select Send to Intruder on the resulting menu (see screenshot below):



Step 7 – Click on the Intruder tab (located between the Proxy and Repeater tabs):

Step 8 – In the resulting **Payload Positions** section, do the following

1. Click the **Clear** button on the right-hand side of the window

- Highlight the string “testUsername” at the bottom of the window
- With the “testUsername” text highlighted, click the **Add** button

See the screenshot below

**1** Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target:  ☒ Update Host header to match target

Buttons: **3** Add \$, **1** Clear \$, Auto \$, Refresh

```

1 POST /login HTTP/1.1
2 Host: ac501fc91f85738dc0b1eff900fb002d.web-security-academy.net
3 Cookie: session=ujIv3LXDmQpCbbojT9Ju7MG2w0YNDrrs
4 Content-Length: 43
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "(Not:A:Brand";v="8", "Chromium";v="99"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://ac501fc91f85738dc0b1eff900fb002d.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.164 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://ac501fc91f85738dc0b1eff900fb002d.web-security-academy.net/login
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 username=2testUsername&password=testPassword

```

When finished, the **testUsername** string should look like the screenshot below:

```

22
23 username=$testUsername$&password=testPassword

```

Step 9 – Click on the **Intruder** → **Payloads** tab (see screenshot below):

Dashboard Target Proxy **Intruder** Repeater

1 x 2 x ...

Positions **Payloads** Resource Pool Options

**Choose an attack type** Start attack

Attack type:

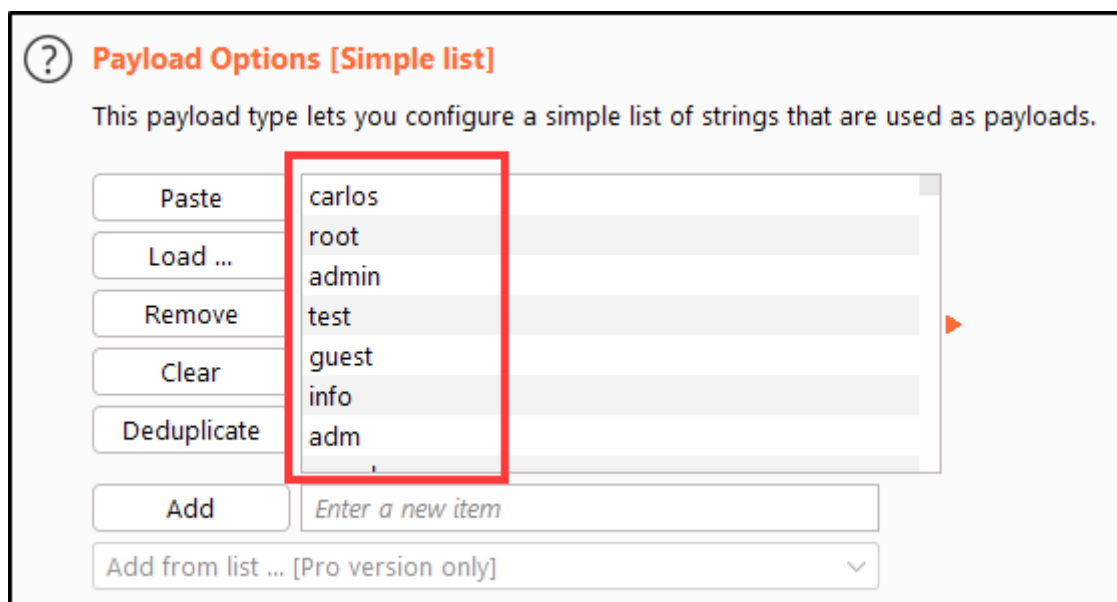
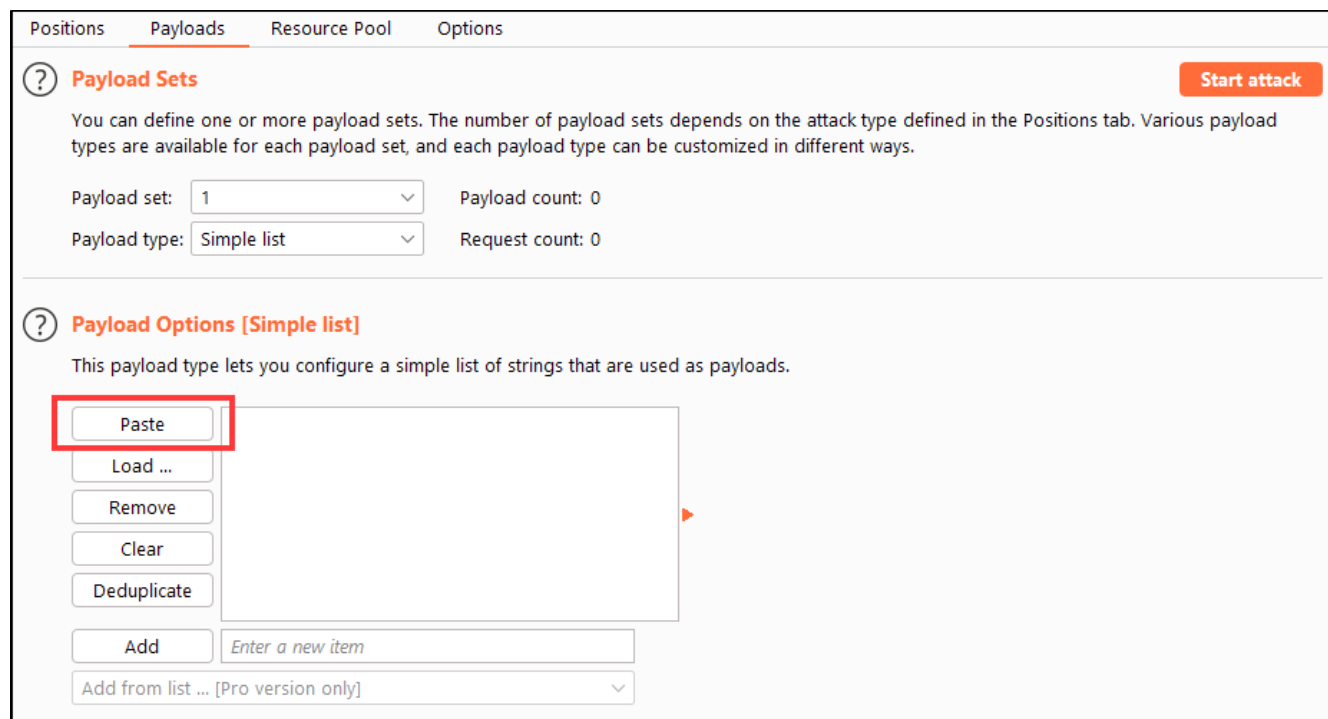
Step 10 – Copy the suggested usernames for the lab:

At the following URL copy the usernames on the webpage:

<https://portswigger.net/web-security/authentication/auth-lab-usernames>

Step 11 – Paste the copied usernames into the Burp Suite **Intruder Payload Options** window

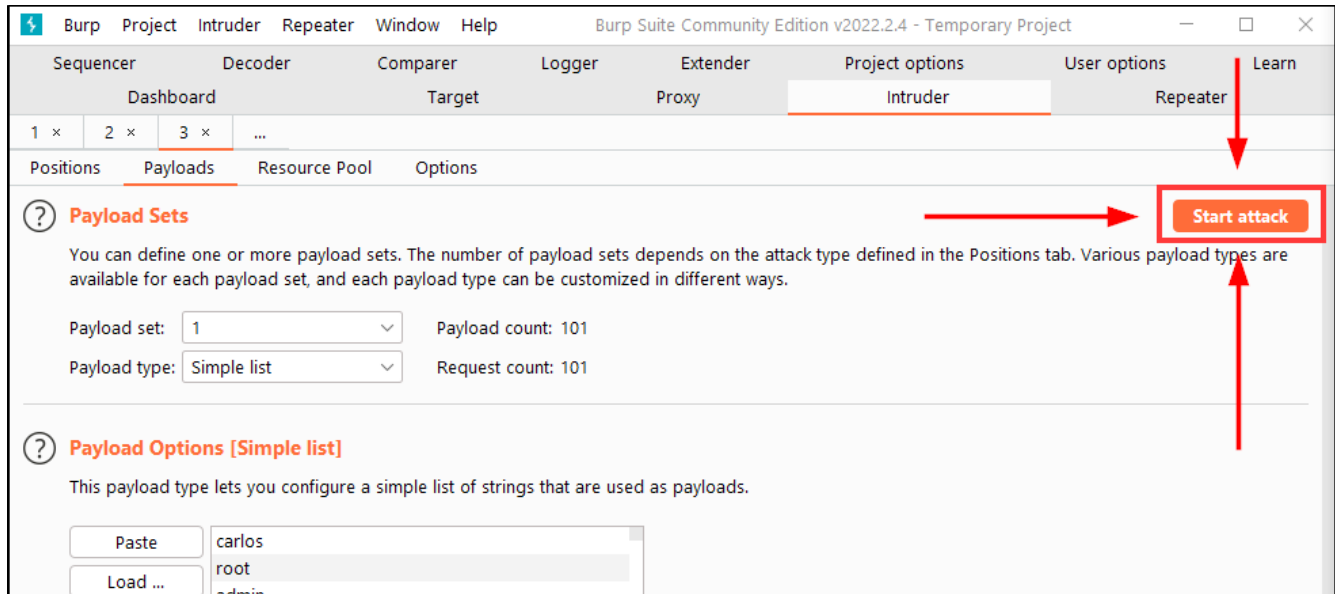
Under the Burp Suite **Intruder** → **Payloads** tab, under the **Payload Options (Simple List)** section, click on the **Paste** button. The list of usernames you copied from the previous step should appear (see the two screenshots below):





## Step 12 – Start the Intruder attack

Click the orange Start attack button at the top-right corner of the Intruder tab (see screenshot).



In the Intruder attack window, you will see the results of the different requests sent in the attack. Click on the Length column in the window (see screenshot below).

The screenshot shows the 'Attack' window with the 'Results' tab selected. The table displays the results of the intruder attack. The 'Length' column is highlighted with a red box. The table has columns: Request, Payload, Status, Error, Timeout, Length, and Comment. The data rows are as follows:

Request	Payload	Status	Error	Timeout	Length	Comment
101	autodiscover	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
69	apache	200	<input type="checkbox"/>	<input type="checkbox"/>	2986	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
1	carlos	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
2	root	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	

Eventually, one of the Requests will return a length that is different from the other requests. Click on that entry, then click on the Response tab (see screenshot below):

Request	Payload	Status	Error	Timeout	Length	Comment
101	autodiscover	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
69	apache	200	<input type="checkbox"/>	<input type="checkbox"/>	2986	1
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
1	carlos	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
2	root	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
3	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
4	test	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
5	guest	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
6	info	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
7	adm	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
8	mysql	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
9	user	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	
10	administrator	200	<input type="checkbox"/>	<input type="checkbox"/>	2984	

Request Response 2

Pretty Raw Hex

```

1 POST /login HTTP/1.1
2 Host: ac381fdc1f15b895c08f7fb7005400b3.web-security-academy.net
3 Cookie: session=LtTyUsEQHKNNJJGPLhiUgUKM3VPsrVipA
4 Content-Length: 37
5 Cache-Control: max-age=0

```

**NOTE:** The username that will return with the correct response is random for each instance of the lab, so the username in the screenshots will not necessarily be the correct one for your lab.

In the **Response** tab, scroll down, and you will see that the server responds with the message **Incorrect password** (see screenshot below).

Request Response

Pretty Raw Hex Render

```

50 </header>
51 <h1>
52 Login
53 </h1>
54 <section>
55 <div class=login-warning>
56 Incorrect password
57 </div>
58 <form class=login-form method=POST action=/login>
59 <label>
60 Username
61 </label>
62 <input required type=username name="username">
63 <label>
64 Password
65 </label>
66 <input required type=password name="password">

```

0 matches

Finished

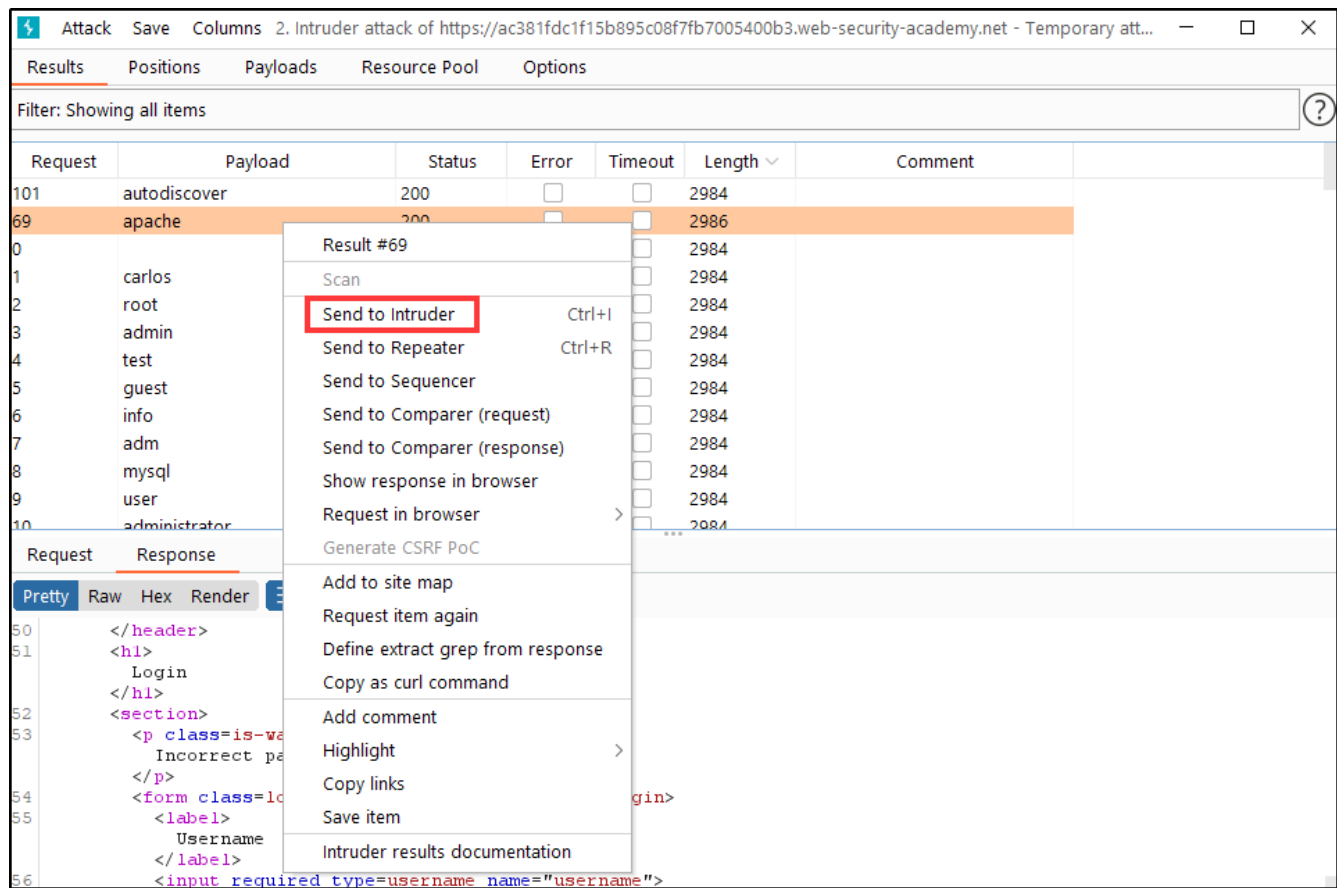
## CONTEXT

This web server reveals which usernames are valid by responding with a different message if the username is valid versus an invalid username.

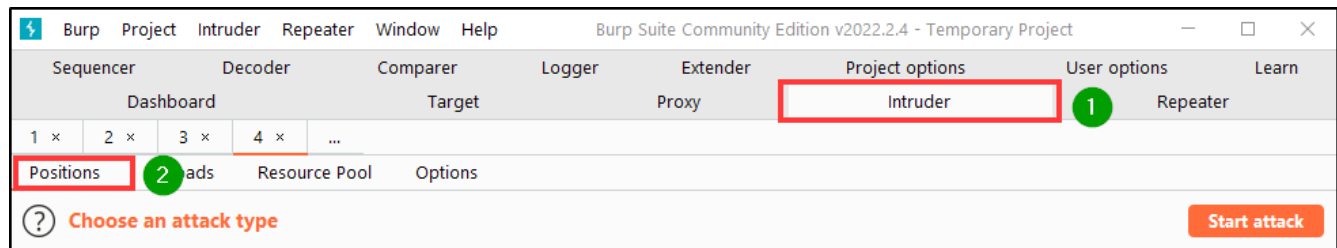
### Step 13 – Start another Intruder attack

In the Intruder attack window, right-click on the request that resulted in the **Incorrect password**

response, then click on the **Send to Intruder** option (see screenshot below).



Click on the **Intruder** → **Positions** tab (see screenshot below):



Repeat process for setting up the Intruder attack positions as detailed in Step 8, with the following changes:

Highlight the string “**testPassword**”, then click the **Add** button, instead of “**testUsername**”.

The resulting “**testPassword**” string should look similar to the screenshot below:

```
22  
23 username= &password=$testPassword$
```

Then setup the Payloads section of the Intruder attack, following the same process as we did in Steps 9-11, with the following changes:

The URL we will visit to obtain our list of payloads will be the following one:

<https://portswigger.net/web-security/authentication/auth-lab-passwords>

Then click the **Start attack** button like we did in Step 12.

In the resulting **Intruder attack** window, click on the **Status** column (see screenshot below):

Attack Save Columns 4. Intruder attack of https://aca91f1f1e04baa8c02633d9007200cc.x						
Results Positions Payloads Resource Pool Options						
Filter: Showing all items						
Request	Payload	Status ▾	Error	Timeout	Length	
100	moscow	200	<input type="checkbox"/>	<input type="checkbox"/>	3073	
56	klaster	302	<input type="checkbox"/>	<input type="checkbox"/>	170	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2986	

After a number of the Intruder requests have been processed, we should see that one of the requests resulted in a **302** status (see screenshot below).

Attack Save Columns 4. Intruder attack of https://aca91f1f1e04baa8c02633d9007200cc.web-s						
Results Positions Payloads Resource Pool Options						
Filter: Showing all items						
Request	Payload	Status ▾	Error	Timeout	Length	
100	moscow	200	<input type="checkbox"/>	<input type="checkbox"/>	3073	
56	klaster	302	<input type="checkbox"/>	<input type="checkbox"/>	170	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2986	

**NOTE:** Valid usernames and passwords are random for each instance of the lab environment, so the valid password in the screenshot above will not necessarily be the same as your lab enviroment.

## CONTEXT

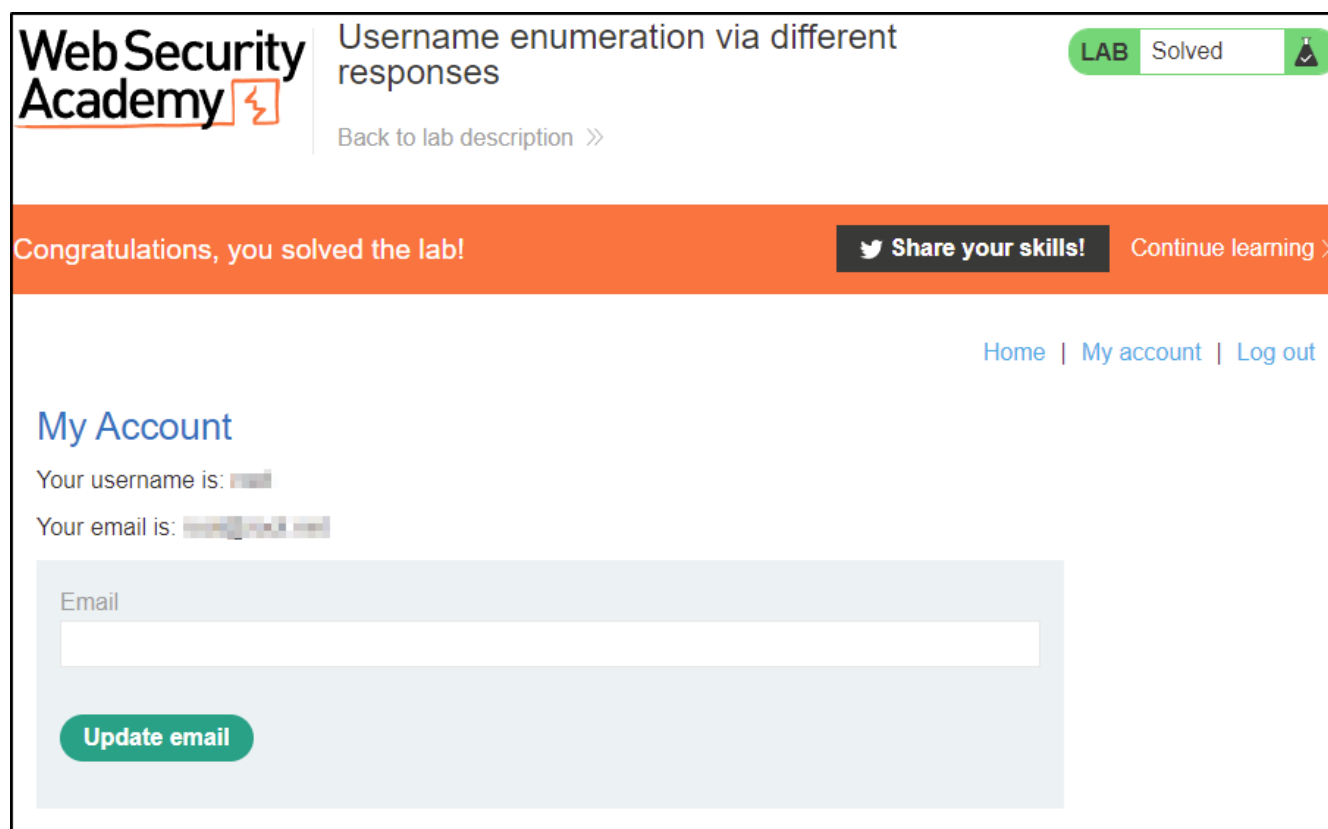
Now that we've finished our brute-force login attack, we have identified one valid username in the system and then identified the matching password to that username. All that remains now is to use those credentials to login and complete the lab.

### Step 14 – Use the found username and password to login

In the Burp web browser, navigate to the lab login page at the following URL:

**[https://<your\\_lab\\_subdomain>.web-security-academy.net/login](https://<your_lab_subdomain>.web-security-academy.net/login)**

Login with the username and password we found in the brute-force attacks. If we login successfully, we should see a screen similar to the screenshot below:



## CONTEXT

The reason we were able to successfully brute-force the login for this lab is because of the following security misconfigurations:

- Alternative responses to failed login attempts from valid usernames and invalid usernames.
- Lack of brute-force login protections, such as account lockout.

Let's move on to the next lab!

## Part 2

### Objective – Solve the 2FA simple bypass lab

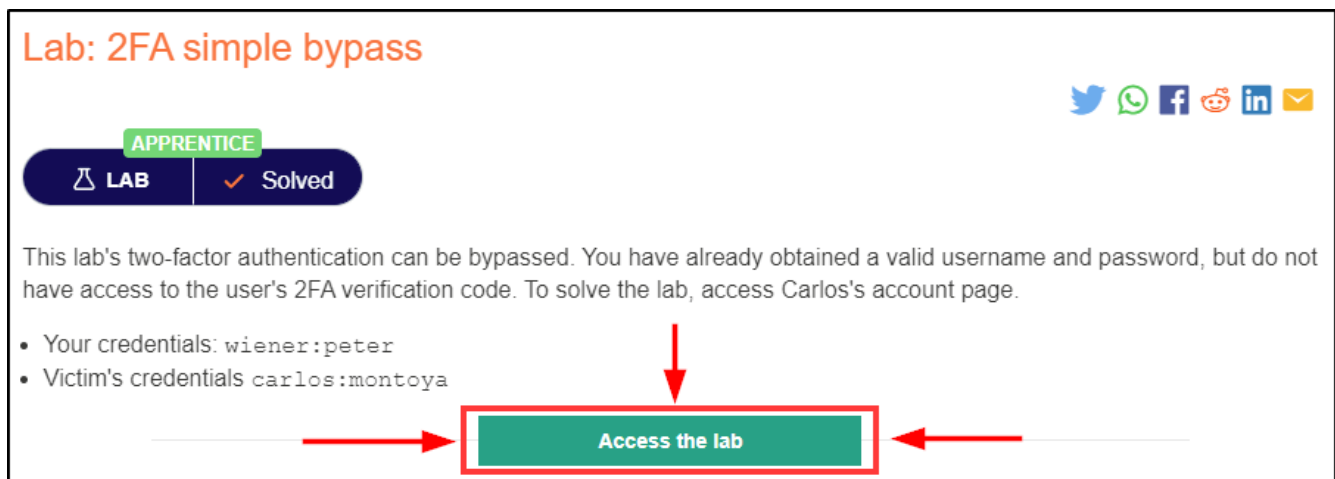
#### Step 1 – Navigate to the following URL:

<https://portswigger.net/web-security/authentication/multi-factor/lab-2fa-simple-bypass>

This lab supplies us with two sets of credentials, “our” user credentials (**wiener:peter**), and the credentials of the account we need to access to complete the lab (**carlos:montoya**).

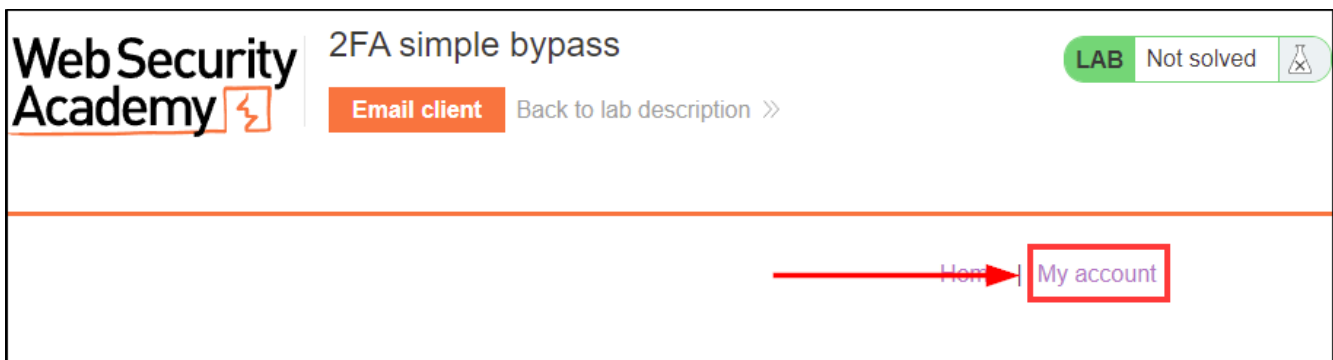
#### Step 2 – Start the lab instance

Click on the green Access the lab button (see screenshot below):



#### Step 3 – Login with our **wiener** account

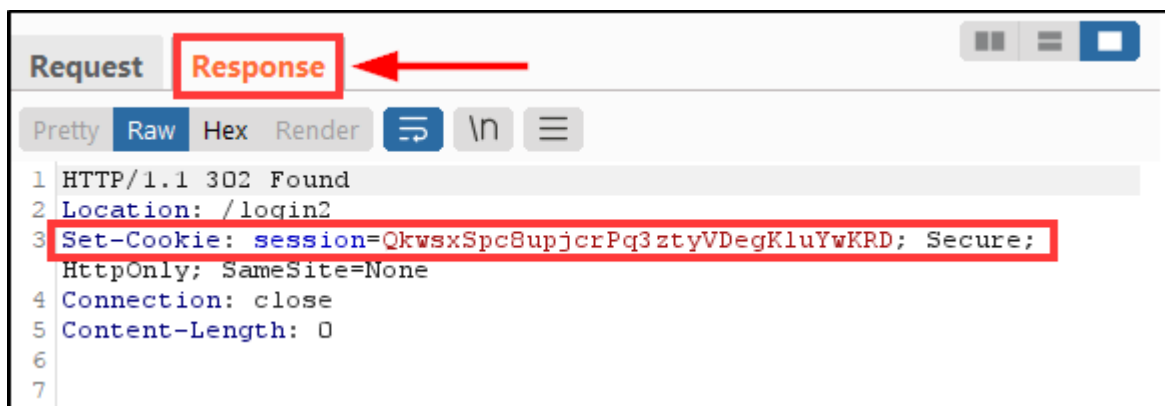
On the lab webpage, click on the **My account** button (see screenshot below):



On the next page, enter the username and password as **wiener** and **peter**, respectively, then click the **Log in** button.

#### Step 4 – Look at the requests and responses in Burpsuite

In the Burp **Proxy** → **HTTP history** tab, click on the **Method** column, then locate the POST request to the **/login** URL and click on it. Then, click on the **Response** tab, note that a session cookie was set (see screenshot below):



#### CONTEXT

The fact that the webserver has already assigned us a session cookie despite the fact that the 2FA portion of login hasn't been completed leads us to suspect that we're already logged in, and that we don't need to supply the 2FA code.

#### Step 5 – Confirm that we're currently logged in

From the 2FA login page, click on the Back to lab home button, which brings us to the lab website's landing page, then on the following page, click on the My account button (see next three screenshots below):

Web Security Academy

2FA simple bypass

LAB Not solved

Back to lab home

Email client

Back to lab description >>

Please enter your 4-digit security code

Login

Web Security Academy

2FA simple bypass

LAB Not solved

Email client

Back to lab description >>

Home

My account

Web Security Academy

2FA simple bypass

LAB Not solved

Email client

Back to lab description >>

Home | My account | Log out

My Account

Your username is: wiener

Your email is: wiener@exploit-ac731f6a1ff01083c0338794012000c7.web-security-academy.net

Email

Update email



After bypassing the 2FA login page, we navigated to our **My account** page and confirmed that we are indeed logged in. We can use this method to login as the **carlos** user since we already have their credentials.

Step 6 – Login as carlos to complete the lab

Click the Logout button located in the top-right portion of the webpage (see screenshot below):



Then click on the My account button on the resulting page. Log in with the following credentials:

Username: **carlos**

Password: **montoya**

On the resulting 2FA login screen, click on the orange **Back to lab home** button, then click on the **My account** button on the lab landing page. This should complete the lab with an orange banner indicating that this is the case (see screenshot below):



## CONTEXT

The app in this lab environment assigned a session cookie to the user after only completing the first half of the login process. Why? One possible answer is that the developer wanted to prevent unauthenticated access to the 2FA login page, which is understandable. One possible safer alternative in this case would be for the webserver to set a cookie specifically for the 2FA login page upon completion of the first part of the login process.

On to the next lab!

## Part 3

### Objective – Complete the Password reset broken logic lab

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

<https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-broken-logic>

In this lab our objective is to reset Carlos' password then log in as Carlos and access their **My account** page. We have been supplied a working set of credentials (**wiener:peter**), and the username of the account we want to access (**carlos**).

#### Step 2 – Activate the lab environment

Click the green **Access the lab** button.

#### Step 3 – Reset the password of our working user account

On the lab landing page, click the **My account** button. On the next page, click the **Forgot password?** button located above the green **Log in** button.

On this page, enter the username **wiener**, then click the green **Submit** button.

After the page responds with the **Please check your email for a reset password link** message, click on the orange **Email client** button located at the top of the page (see screenshot below):

**Web Security Academy**

LABNot solved

⚗️

Password reset broken logic

Back to lab home

Email client

Back to lab description >>

Home | My account

Please check your email for a reset password link.

In the resulting tab, click on the link beginning with **https://** (see screenshot below):

Your email address is **wiener@exploit-acc81ff71f08bb8ac1534a0801750054.web-security-academy.net**

Displaying all emails @exploit-acc81ff71f08bb8ac1534a0801750054.web-security-academy.net and all subdomains

Sent	From	Subject	Body	
2022-03-29 16:58:04 +0000	no-reply@ac901f911f69bbf4c1894a95003600e3.web-security-academy.net	Account recovery	<p>Hello!</p> <p>Please follow the link below to reset your password.</p> <p><b><a href="https://ac901f911f69bbf4c1894a95003600e3.web-security-academy.net/forgot-password?temp-forgotten-password-token=QjsSAwH20dCd3VuIWDaUY6xv9DiSp23P">https://ac901f911f69bbf4c1894a95003600e3.web-security-academy.net/forgot-password?temp-forgotten-password-token=QjsSAwH20dCd3VuIWDaUY6xv9DiSp23P</a></b></p> <p>Thanks, Support team</p>	<a href="#">View raw</a>

In the next tab, fill in the two fields with the following:

**New password:** **password**  
**confirm new password:** **password**

Then click the green **Submit** button.

## CONTEXT

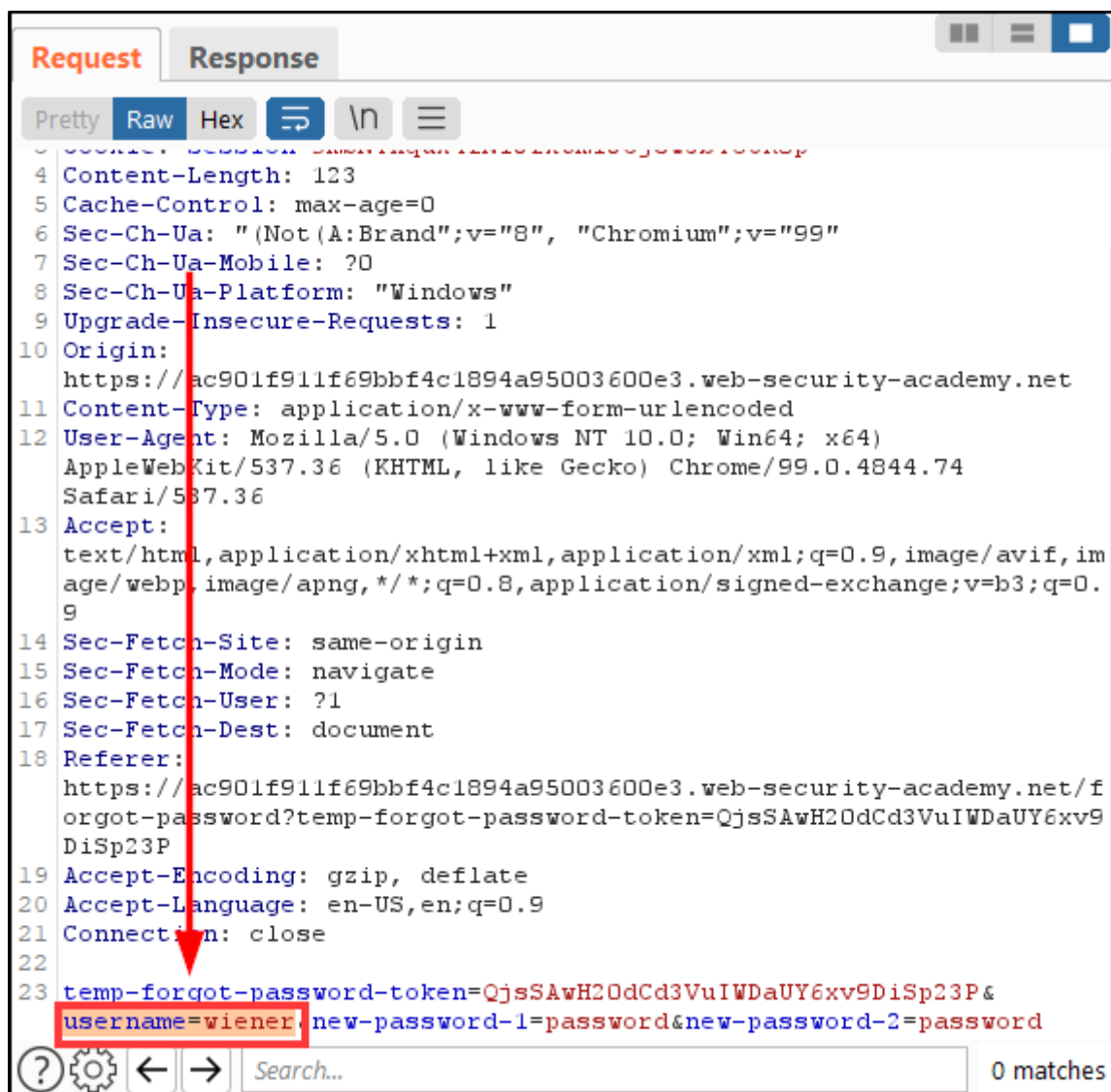
We have gone through the motions of resetting our default user's password in preparation for examining the requests in Burpsuite.

**Step 4 – Look at the POST request for resetting the user password**

In the Burpsuite **Proxy** → **HTTP history** tab, click on the **Method** column, then scroll through the list until we find the last POST request to the **/forgot-password** URL with the **temp-forgot-password-token** parameter (see screenshot below):

Dashboard		Target	Proxy	Intruder	Repeater					
Intercept	HTTP history	WebSockets history	Options							
Filter: Hiding CSS, image and general binary content										?
#	Host	Meth... ^	URL	Params	Edited	Status	Length	MIME type	Exte	
108	https://www.youtube.com	POST	/youtubei/v1/log_event?alt=json&key=...	✓		200	510	JSON		
127	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=1204270682...	✓		200	617	text		
128	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=1204270682...	✓		200	616	text		
156	https://ac781f0a1fbe8034c0c46...	POST	/forgot-password	✓		200	2671	HTML		
175	https://ac781f0a1fbe8034c0c46...	POST	/forgot-password?temp-forgot-passw...	✓		302	73			
197	https://ac781f0a1fbe8034c0c46...	POST	/login	✓		302	170			
223	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=2002438339...	✓		200	617	text		
224	https://www.google-analytics.co...	POST	/j/collect?v=1&_v=j96&a=2002438339...	✓		200	616	text		
227	https://portswigger.net	POST	/users?returnUrl=%2Facademy%2Flab...	✓		200	1824	JSON		
257	https://ac901f911f69bbf4c1894...	POST	/forgot-password	✓		200	2671	HTML		
263	https://ac901f911f69bbf4c1894...	POST	/forgot-password	✓		200	2671	HTML		
281	https://ac901f911f69bbf4c1894...	POST	/forgot-password?temp-forgot-passw...	✓		302	73			

In the Request tab in the lower portion of the screen, scroll down until we can see the request body parameters, taking note of the **username** parameter (see screenshot below):



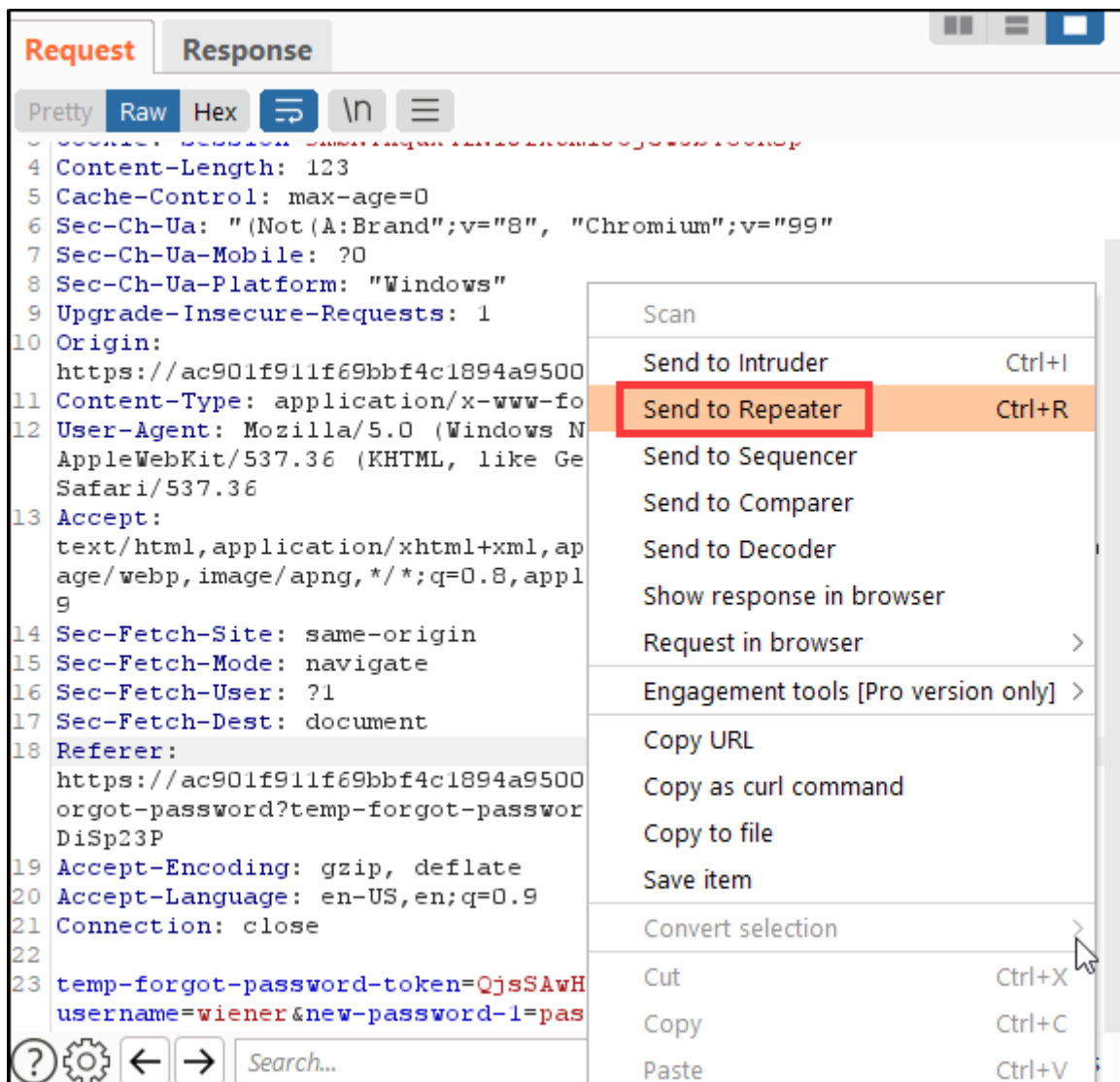
## CONTEXT

We see that password reset email link leads us to a page where we can submit a new password while passing in a parameter token named **temp-forgot-password-token**. Curiously, we also pass a parameter that appears to state which username the new password will be associated with. We should test whether or not:

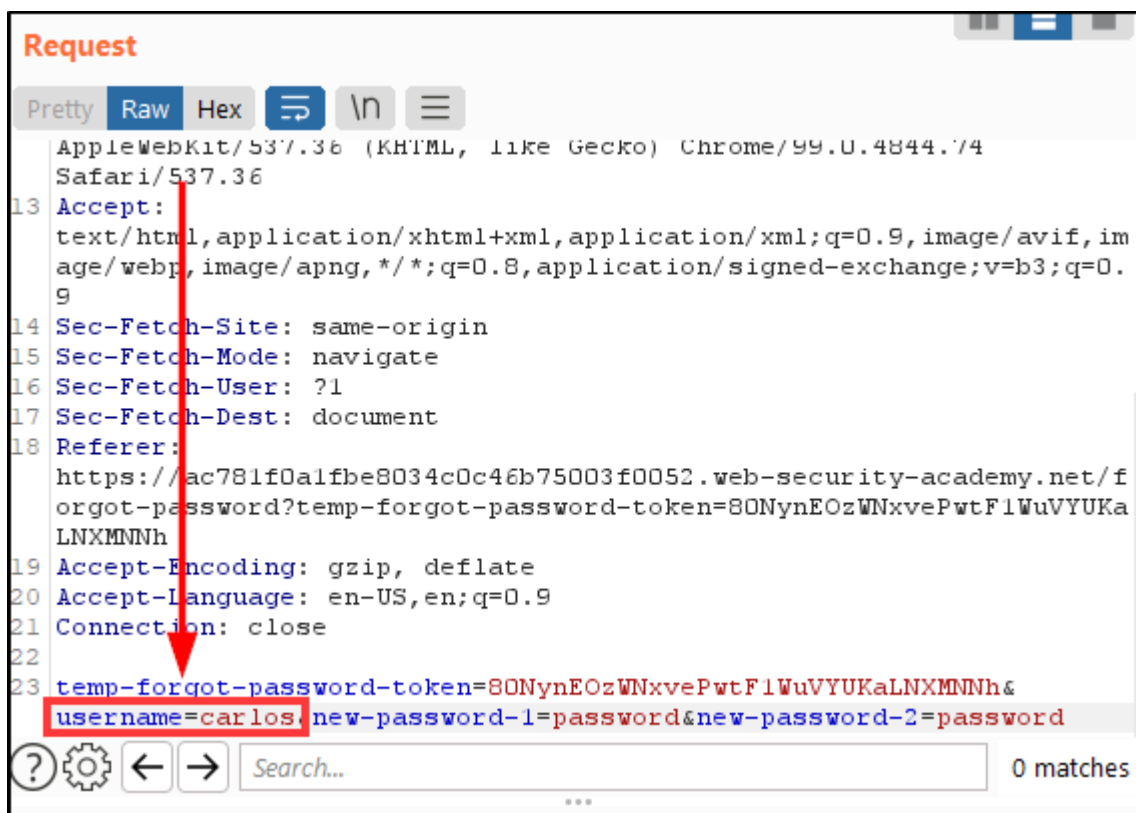
- A) The **temp-forgot-password-token** can be used more than once, and
- B) The **temp-forgot-password-token** is limited to changing the password for one specific username.

## Step 5 – Send the POST request to the Burp Repeater, then replay the request

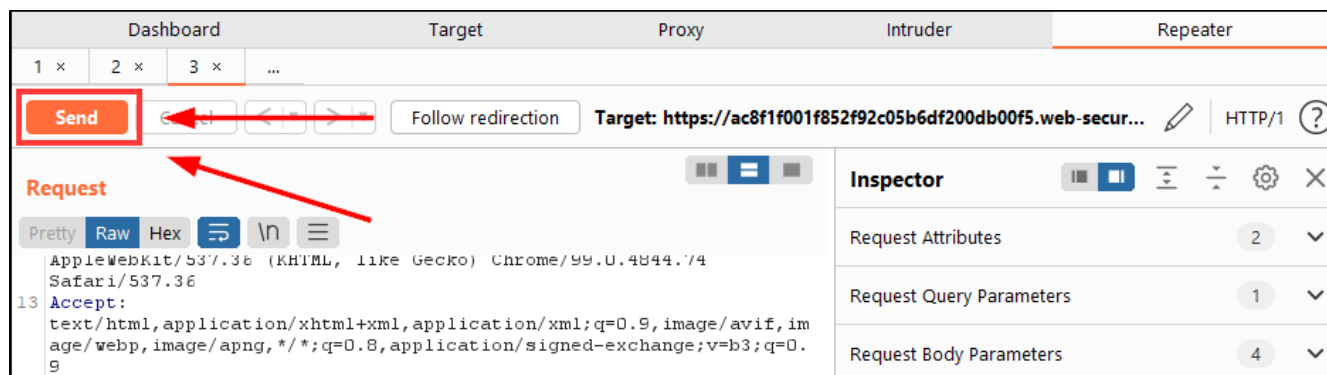
Right-click in the **Request** tab window and select **Send to Repeater** from the menu options(see screenshot below):



Click on the **Repeater** tab, located next to the **Intruder** tab. In the Request window, scroll down until you see the body parameters (**temp-forgot-password-token**, **username**, etc). Replace the value of the username parameter (**wiener**) with **carlos**(see screenshot below).



Click the orange **Send** button located above the Request header (see screenshot below):



After clicking the Send button, you should see a response identical to the one in the screenshot below:

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Location: /
3 Connection: close
4 Content-Length: 0
5
6
```

## CONTEXT

By sending the password reset POST request to the Burp Repeater, we were able to manipulate the values of the parameters in the request and re-send the request in the context of the target **carlos** user.

### Step 6 – Login as carlos to complete the lab

Click on the **My account** button, then on the login page, log in using the following credentials:

**Username:** carlos  
**Password:** password

Upon successful login, we should see the following message that indicates that we have completed the lab (see screenshot below):

 Password reset broken logic

LAB Solved

Back to lab description >>

Congratulations, you solved the lab!

Share your skills! Continue learning >>

Home | My account | Log out

My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

## CONTEXT

The security flaw in this lab was allowing password resets to occur through a POST request parameter without any other mechanism to ensure that only the password reset request could only be done in the



context of user who requested it.

One possible solution to securing this password reset function would be to configure the **temp-forgot-password-token** to be valid only with one username.

## Part 4

### Objective – Complete the Username enumeration via subtly different responses

Step 1 – Navigate to the following URL:

<https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-subtly-different-responses>

#### CONTEXT

This lab's objective is similar to the first lab in the workshop, but this time the criteria for determining which request was successful through brute-force login attempts will be more difficult to parse.

Step 2 – Activate the lab environment

Click on the green Access the lab button.

Step 3 – Attempt a test login

Click on the **My account** button. On the subsequent login page, attempt a login using the following credentials:

**Username:** testUsername

**Password:** testPassword

Step 4 – Find the login request in Burpsuite

In the Burpsuite **Proxy** → **HTTP history** tab, click on the **Method** column and locate the POST request to the **/login** URL. Right-click the request and select **Send to Intruder**.

Step 5 – Determine a valid username through brute-force login attack

Click on the Intruder tab. Set up the Payload Positions in the same manner as the first brute-force lab.

However, before starting the attack, we must do the following:

Click on the **Options** tab (beside the **Resource Pool** tab), and scroll down to the **Grep – Extract** section. In that section, click the **Add** button. In the resulting **Define extract grep item** window, scroll down until we locate the **Invalid username or password**. String, then highlight it (see screenshot below).

```

50         </header>
51         <header class="notification-header">
52         </header>
53         <h1>Login</h1>
54         <section>
55             <p class=is-warning>Invalid username or password.</p>
56             <form class=login-form method=POST action=/login>
57                 <label>Username</label>
58                 <input required type=username name="username">
59                 <label>Password</label>
60                 <input required type=password name="password">
61                 <button class=button type=submit> Log in </button>
62             </form>
63         </section>
64     </div>
65 </section>
66 </div>
67 </body>
68 </html>
69

```

Click the **OK** button at the bottom of the window, then click the orange **Start attack** button at the top-right portion of the **Intruder** tab (you will need to scroll back up to the top of the tab).

In the resulting Intruder attack window, there will be a column for **-warning>**. Click on that column and eventually we will see one request where the warning will be “Invalid username or password”, as opposed to “Invalid username or password.”. Note the lack of a period on the warning message that is different from the others (see screenshot below).

-warning> ^
Invalid username or password.
Invalid username or password
Invalid username or password.
Invalid username or password.

Click on that request, then right-click, and select the **Send to Intruder** option.

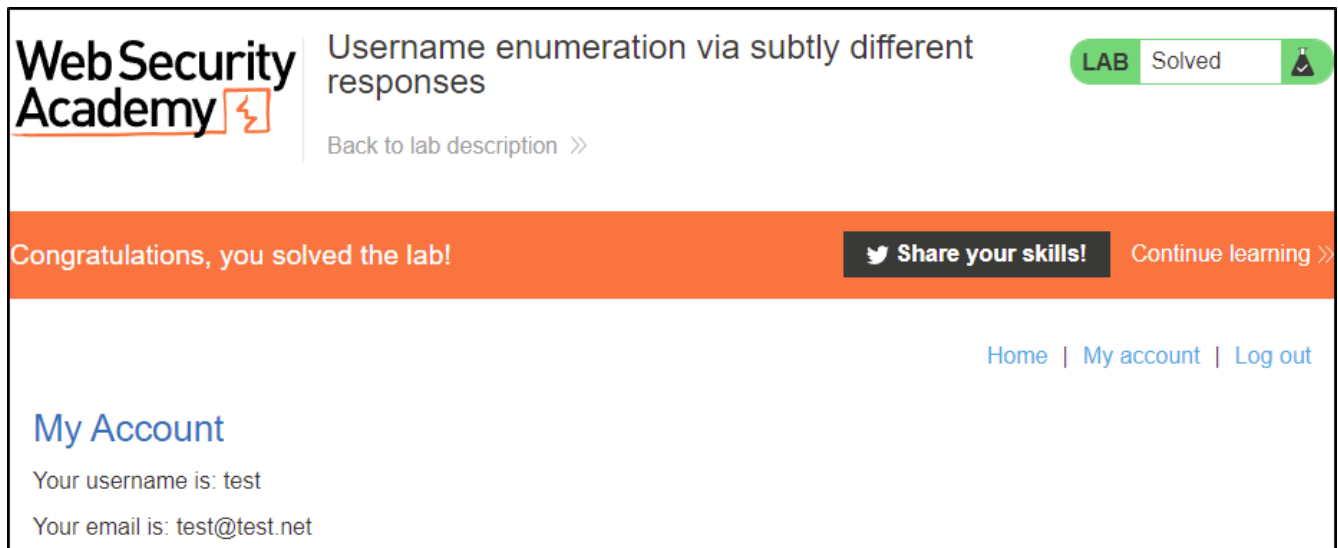
## Step 6 – Determine a valid password through brute-force login attack

In the **Intruder** tab, setup the **Payload Positions** to brute-force the password, like we did in the previous lab.

In the **Intruder attack** window, click on the **Status** column. Eventually there will be one request that returns a 302 status instead of the 200 status the other requests return. Take note of the **Payload** for this request.

Step 7 – Login with the discovered credentials to finish the lab

Using the username and password discovered in the previous two steps, login to the application. On successful login, we should see the following message in an orange banner to indicate we've finished the lab (see screenshot below):



## CONTEXT

The security issue in this lab was very subtle, in that the difference of the failed login messages between a valid username and an invalid username was only a difference of a single character (the missing period). This kind of security issue could appear as a result of input error from the developers. If identified, this issue would be very easy to fix.

## Part 5

### Objective – Complete the Username enumeration via response timing lab

Step 1 – Navigate to the following URL:

<https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-response-timing>

In this lab, we'll determine a valid username for brute-force attack through server response times. This lab also implements IP-based brute-force protection, which we'll have to bypass by using HTTP request headers.

Step 2 – Start up the lab environment

We also need to change the type of attack we're performing. In the **Choose an attack type** heading, change the **Attack type** from the default **Sniper**, to **Pitchfork** (see screenshot below):

Positions Payloads Resource Pool Options

? Choose an attack type Start attack

Attack type: Pitchfork

? Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Next, in the Payloads tab, we need to set **Payload set 1** to a **Payload type** of **Numbers of Sequential** numbers from **000** to **100**, with a Step of **1**. When complete, the settings should look similar to the following screenshot:

Positions Payloads Resource Pool Options

? Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type. The number of payload sets available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 101

Payload type: Numbers Request count: 101

? Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From: 000

To: 100

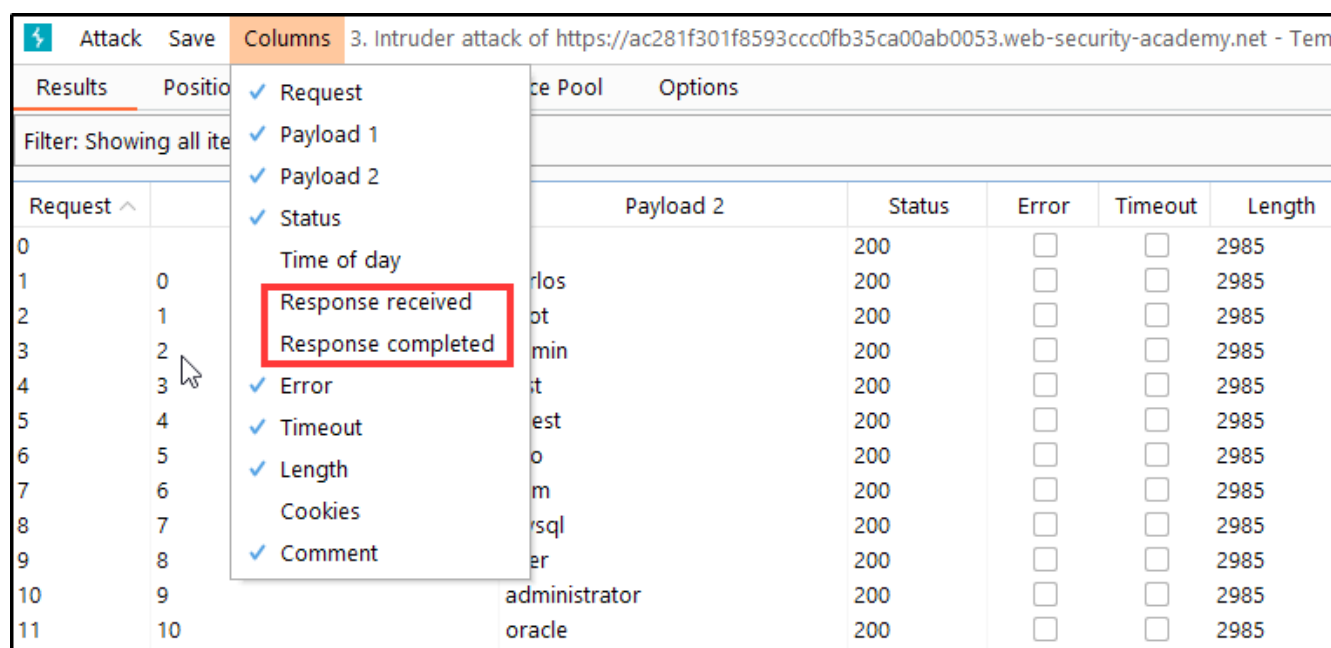
Step: 1

How many:

**Payload set 2** is the usual list of usernames that we've been using in previous brute-force labs. When ready, click the **Start attack** button.

While the attack is running, in the **Intruder attack** window, click the **Columns** tab at the top of the window, then select **Response received** and **Response completed** from the menu (see screenshot

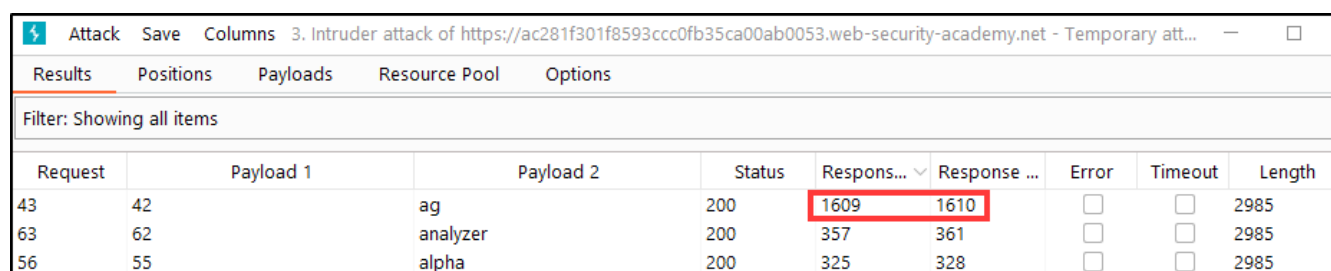
below):



The screenshot shows the Burp Suite interface with the 'Columns' menu open. The menu items are: Request, Payload 1, Payload 2, Status, Time of day, Response received (highlighted with a red box), Response completed (highlighted with a red box), Error, Timeout, Length, Cookies, and Comment. The background table shows a list of requests with columns: Request, Position, Payload 2, Status, Error, Timeout, and Length.

Request	Position	Payload 2	Status	Error	Timeout	Length
0			200	<input type="checkbox"/>	<input type="checkbox"/>	2985
1	0	los	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
2	1	ot	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
3	2	min	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
4	3	it	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
5	4	est	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
6	5	o	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
7	6	m	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
8	7	sql	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
9	8	er	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
10	9	administrator	200	<input type="checkbox"/>	<input type="checkbox"/>	2985
11	10	oracle	200	<input type="checkbox"/>	<input type="checkbox"/>	2985

Then click on the **Response received** column. Eventually, one of the requests will have a **Response received** value that is much much higher than the rest of the requests (see screenshot below).



The screenshot shows the Burp Suite interface with the 'Response received' column highlighted. The table has columns: Request, Payload 1, Payload 2, Status, Response received (highlighted with a red box), Response length, Error, Timeout, and Length.

Request	Payload 1	Payload 2	Status	Response received	Response length	Error	Timeout	Length
43	42	ag	200	1609	1610	<input type="checkbox"/>	<input type="checkbox"/>	2985
63	62	analyzer	200	357	361	<input type="checkbox"/>	<input type="checkbox"/>	2985
56	55	alpha	200	325	328	<input type="checkbox"/>	<input type="checkbox"/>	2985

Step 5 – Obtain the password associated

Right-click on this request and select **Send to Intruder** from the menu. Going back to the **Intruder** tab, the **Payload Positions** section will need to set the following:

```
X-Forwarded-For: numbers  
password=testPassword
```

When the **Payload Positions** settings are complete, it should look similar to the following screenshot:

```
22 X-Forwarded-For: $numbers$
23
24 username=<img alt="redacted" data-bbox="425 103 445 113"/>&password=$testPassword$
```

The **Payloads** tab should be set similar to the last Intruder attack, but with **Payload 2** set to the list of passwords, instead of the list of usernames. When ready, start the attack.

During the attack, click on the **Status** column. Eventually one of the requests will return a 302 status response instead of the 200 status the rest of the requests are returning (see screenshot below).

Attack Save Columns 4. Intruder attack of https://ac281f301f8593ccc0fb35ca00ab0053.web-security-academy.net - Temp

ResultsPositionsPayloadsResource PoolOptions

Filter: Showing all items

Request	Payload 1	Payload 2	Status ▾	Error	Timeout	Length
80	79	love	200	<input type="checkbox"/>	<input type="checkbox"/>	3072
74	73	ginger	302	<input type="checkbox"/>	<input type="checkbox"/>	170
0			200	<input type="checkbox"/>	<input type="checkbox"/>	2985

**NOTE:** Valid usernames and passwords are random for each instance of the lab environment, so the valid password in the screenshot above will not necessarily be the same as your lab environment.

### Step 6 – Use the captured credentials to login and complete the lab

Access the **My account** page and use the password and the username captured from the brute-force attacks to login. Upon successful login, we should see output similar to this screenshot:

**Web Security Academy**

Username enumeration via response timing

LAB Solved

Back to lab description >>

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account | Log out

### My Account

Your username is: <img alt="redacted" data-bbox="225 861 245 871"/>

Your email is: <img alt="redacted" data-bbox="195 883 265 893"/>

## CONTEXT

The main security issue with the web app in this lab is that it does not process the login request if the username is not valid. This issue can be resolved by directing the server to process the login POST request whether the username is valid or not.

On to the next lab!

## Part 6

### Objective – Complete the Broken brute-force protection, IP block lab

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

<https://portswigger.net/web-security/authentication/password-based/lab-broken-bruteforce-protection-ip-block>

In this lab there is a logic flaw in the password brute-force protection, so we will exploit that flaw to brute-force the password. We have been provided with our default user account and password(**wiener:peter**), and the username of the account we need to access to complete the lab (**carlos**).

### Step 2 – Activate the lab environment

Click the green **Access the lab** button.

### Step 3 – Test the brute-force protection

Attempt a few failed logins with the **carlos** account to see how many failed logins activates the account lockout. After three failed login attempts, we see this message in the screenshot below:



# Login

You have made too many incorrect login attempts. Please try again in 1 minute(s).

Username

Password

Log in

Now let's try a legit log after 2 failed ones and see if that resets the failed login count. We will find that two failed logins, followed by one successful one will reset the count for account lockout.

#### Step 4 – Prepare the Intruder brute-force attack

Locate the login POST request in the **Proxy** → **HTTP History** tab, then send it to the Intruder. In the **Intruder** → **Positions** tab, set the username and password as positions, and set the type of attack from **Sniper** to **Pitchfork**. For the Payload Sets 1 and 2, we will use two custom lists, with valid credentials for the **wiener** user as the first entry in a set of three, with the other two entries using **carlos** as the username and items from our standard password list. We can find the username and password list at the following URLs:

<https://github.com/0xSaihat/workshops/blob/main/BurpSuiteLab6Usernames.txt>

<https://github.com/0xSaihat/workshops/blob/main/BurpSuiteLab6Passwords.txt>

One last thing we need to do before we start the attack is to click on the **Intruder** → **Resource Pool** tab and click **Create new resource pool**, setting the **Maximum concurrent requests** to **1** (see next two screenshots below):

Dashboard		Target	Proxy	Intruder	Repeater
1 x	2 x	3 x	4 x	5 x	...
Positions	Payloads	Resource Pool	Options		
<div><div>?</div><h3>Resource Pool</h3><p>Specify the resource pool in which the attack will be run. Resource pools are used to manage the usage of system resources across multiple tasks.</p></div>					<div>Start attack</div>

☒ Create new resource pool

Name: Custom resource pool 1

☒ Maximum concurrent requests: 1

☐ Delay between requests: milliseconds

☒ Fixed

## Step 5 – Start the attack and brute-force discover the password

Click the **Start attack** button. During the attack, click on the **Status** column and eventually there will be a single request with the **carlos** payload that returns with a 302 status code among the many **wiener** payloads which all return 302 (see screenshot below).

Attack Save Columns 8. Intruder attack of https://ac3c1fa81f17762bc0dab3fd00c300cc.web-security-academy.net - Temporary att... <span>?</span>							
Results Positions Payloads Resource Pool Options							
Filter: Showing all items							
Request	Payload 1	Payload 2	Status ▾	Error	Timeout	Length	Comment
121	wiener	peter	302	<input type="checkbox"/>	<input type="checkbox"/>	170	
124	wiener	peter	302	<input type="checkbox"/>	<input type="checkbox"/>	170	
127	wiener	peter	302	<input type="checkbox"/>	<input type="checkbox"/>	170	
128	<b>carlos</b>	matthew	<b>302</b>	<input type="checkbox"/>	<input type="checkbox"/>	170	
130	wiener	peter	302	<input type="checkbox"/>	<input type="checkbox"/>	170	

Take note of the password payload for this request (every lab instance will have a different password).

## Step 6 – Use the brute-force captured password to login and complete the lab

At the app's login page, use the following credentials to log in:

Username: **carlos**

Password: [password found in previous step]

Upon successful login, there should be an orange banner that appears that indicates we've finished the lab (see screenshot below):

**Web Security Academy** 

Broken brute-force protection, IP block

LAB Solved 

[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >](#)

[Home](#) | [My account](#) | [Log out](#)

**My Account**

Your username is: carlos

## CONTEXT

This lab's web app had a logic flaw in its anti-brute-forcing function. The function was supposed to lockout and prevent logins from a particular IP if there were three failed login attempts from the same IP address in a short amount of time. However, a successful login from the same IP address would reset the number of failed login attempts for that IP address. This enables brute-force username and password attacks if a set of valid credentials is available. This issue could be resolved by eliminating the reset of the failed login counter when a successful login occurs from any given IP address.

On to the final lab!

## Part 7

### Objective – Complete the Username enumeration via account lock lab

Step 1 – Access the lab description page

Navigate to the following URL:

<https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-account-lock>

The lab description indicates that there is a logic flaw in the account locking functionality that allows brute-force attacks on both usernames and passwords.

### Step 2 – Start the lab instance

Click on the green **Access the lab** button.

### Step 3 – Obtain a baseline failed login request

On the app's **My account** page, provide invalid credentials to get a baseline failed login request in the BurpSuite HTTP history. In the **Proxy** → **HTTP History** tab, locate that request and send it to the Intruder.

#### Step 4 – Conduct multiple Intruder attack to obtain a valid username

In the **Intruder** → **Positions** tab, we will set payload positions for the **username** parameter as usual and use the standard username list in the **Payload** tab. However, since we want to test account lockout for specific usernames, we will start 4 Intruder attacks in quick succession. In the fourth Intruder attack window, click on the **Length** column. Eventually, there will be one request that has a different length than the other requests (see screenshot below):

99	auth	200	<input type="checkbox"/>	<input type="checkbox"/>	2976
100	auto	200	<input type="checkbox"/>	<input type="checkbox"/>	2976
79	aq	200	<input type="checkbox"/>	<input type="checkbox"/>	3028
101	autodiscover	200	<input type="checkbox"/>	<input type="checkbox"/>	2976

If we look at the response for this request, there is a message that indicates that this user account has been locked due to excessive failed login attempts (see screenshot below):

Request	Response
<div>Pretty Raw Hex Render</div> <pre>52     &lt;/h1&gt; 53     &lt;section&gt; 54       &lt;p class=is-warning&gt; 55         You have made too many incorrect login attempts. Please try again in 1 minute(s). 56       &lt;/p&gt; 57       &lt;form class=login-form method=POST action=/login&gt; 58         &lt;label&gt; 59           Username 60         &lt;/label&gt; 61         &lt;input required type=username name="username"&gt; 62         &lt;label&gt;</pre>	

Right-click the request and send it to the Intruder.

#### Step 5 – Conduct multiple Intruder attack to determine password

Setup the standard Sniper Intruder attack to determine the password for the username that we discovered in the last step, with the standard password list as the Payload. Start the attack, then click on the Status column. Eventually there will be one payload that returns a 302 status (see screenshot below:)

Attack Save Columns 16. Intruder attack of https://acf81fd71e3aefd8c022381e00e3000d.web-security-academy.net - Tem

Results Positions Payloads Resource Pool Options


Filter: Showing all items

Request	Payload	Status ▾	Error	Timeout	Length	
100	moscow	200	<input type="checkbox"/>	<input type="checkbox"/>	3115	You have made too many incorre
37	zxcvbnm	302	<input type="checkbox"/>	<input type="checkbox"/>	170	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	2976	Invalid username or password.

Step 6 – Login to the app and complete the lab.


In the lab app's login page, use the captured username and password to login on the **My account** page. Upon successful login, there will be an orange banner that indicates we have completed the lab (see screenshot below):

NOTE: According to the official lab solution, logging in during the account lockout period should not be possible, but if the correct username and password is provided it will have a shorter response length than the rest of the requests.




## Username enumeration via account lock

[Back to lab description >>](#)

LAB
Solved


Congratulations, you solved the lab!


[Share your skills!](#)
[Continue learning >>](#)

[Home](#) | 
 [My account](#) | 
 [Log out](#)

### My Account

Your username is:

Your email is:

## CONTEXT

In this lab application there was an account lockout mechanism that was linked to specific user accounts, rather than IP addresses. We found the lockout mechanism could be abused to enumerate valid usernames in the app. Additionally, the server was not coded to provide the same error message during account lockout to requests with invalid passwords vs valid passwords. Although it is difficult to prevent username enumeration via account lockout, it is more important to prevent enumeration of passwords during account lockout due to misconfigured error messages.

## 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>