SQL Injection de PortSwigger

ERIC SERRANO MARÍN

ÍNDICE

Retrieving hidden data (3 of 3) 2
Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data 2
Subverting application logic (2 of 2) 4
Lab: SQL injection vulnerability allowing login bypass 4
Determining the number of columns required (3 of 4) 6
Lab: SQL injection UNION attack, determining the number of columns returned by the query 6
Finding columns with a useful data type (2 of 2) 8
Lab: SQL injection UNION attack, finding a column containing text 8
Using a SQL injection UNION attack to retrieve interesting data (2 of 2) 11
Lab: SQL injection UNION attack, retrieving data from other tables 11
Retrieving multiple values within a single column (2 of 2) 13
Lab: SQL injection UNION attack, retrieving multiple values in a single column 13
Examining the database (5 of 5) 16
Lab: SQL injection attack, listing the database contents on non-Oracle databases 16
Exploiting blind SQL injection by triggering conditional responses (4 of 4) 20
Lab: Blind SQL injection with conditional responses 20
Error-based SQL injection (7 of 7) 27
Lab: Visible error-based SQL injection 27
Exploiting blind SQL injection by triggering time delays (3 of 3) 34
Lab: Blind SQL injection with time delays and information retrieval 34
Exploiting blind SQL injection using out-of-band (OAST) techniques (5 of 5) 42
Lab: Blind SQL injection with out-of-band data exfiltration 42
SQL injection in different contexts (2 of 2) 43
Lab: SQL injection with filter bypass via XML encoding 43

Retrieving hidden data (3 of 3)

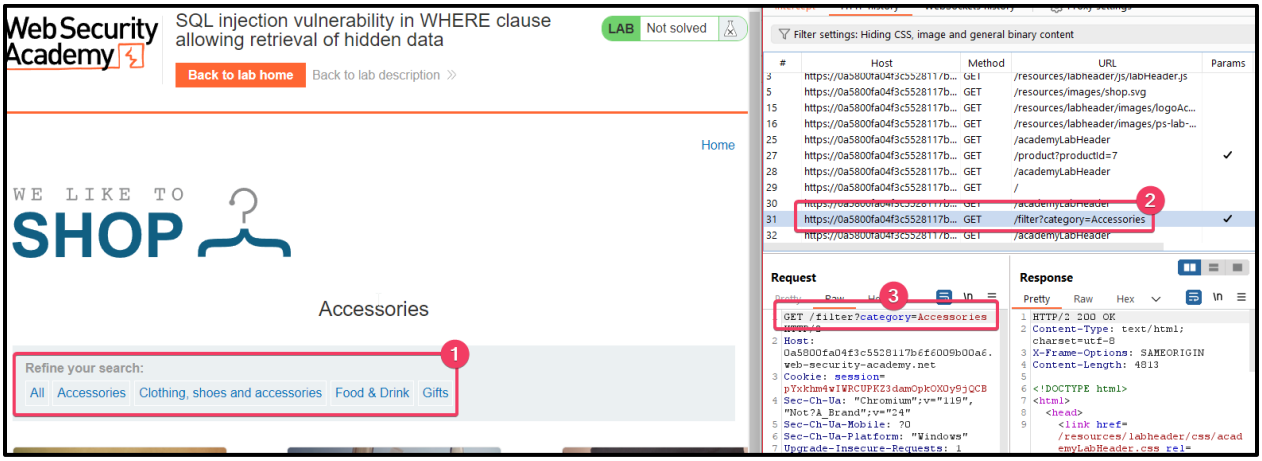
Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

This lab contains a SQL injection vulnerability in the product category filter. When the user selects a category, the application carries out a SQL query like the following:

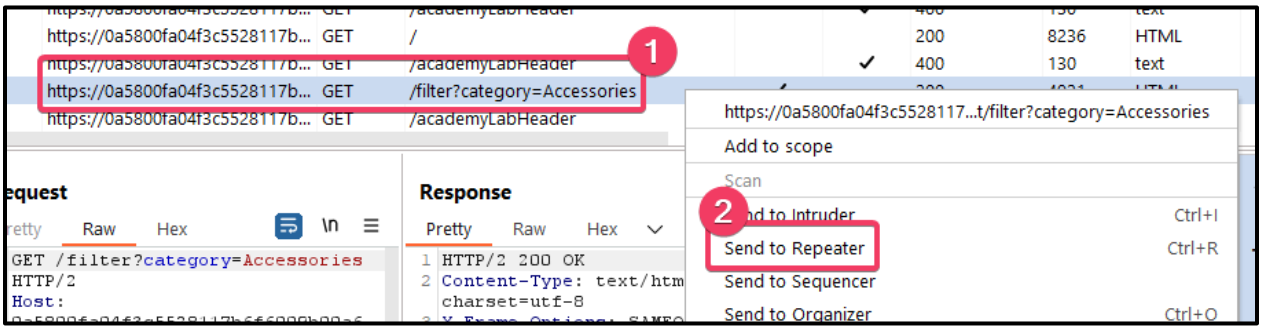
SELECT * FROM products WHERE category = 'Gifts' AND released = 1

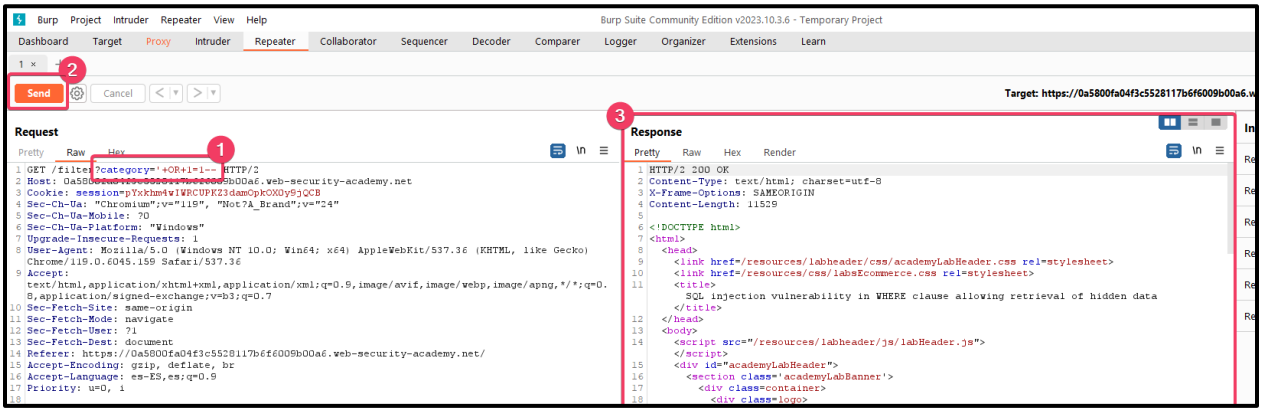
To solve the lab, perform a SQL injection attack that causes the application to display one or more unreleased products.

1. Primero vamos a hacer clic en algún producto para buscar y interceptaremos el request para poder cambiar el parámetro category.



2. Lo llevamos Repeater y cambiaremos el parámetro category.





Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

APPRENTICE

LAB Solved

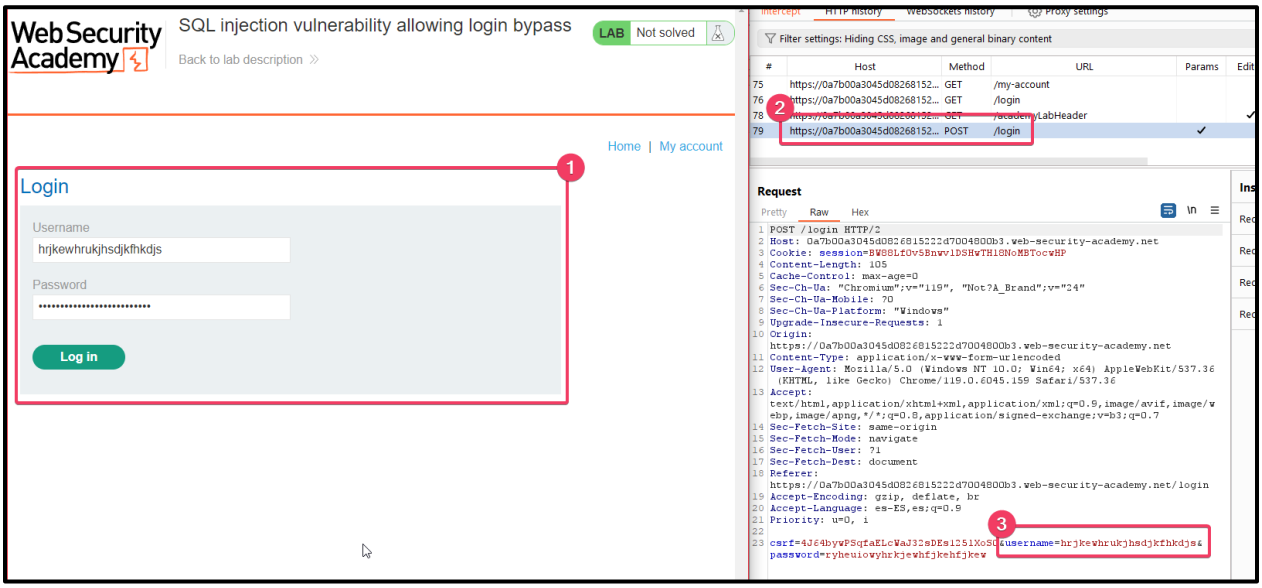
Subverting application logic (2 of 2)

Lab: SQL injection vulnerability allowing login bypass

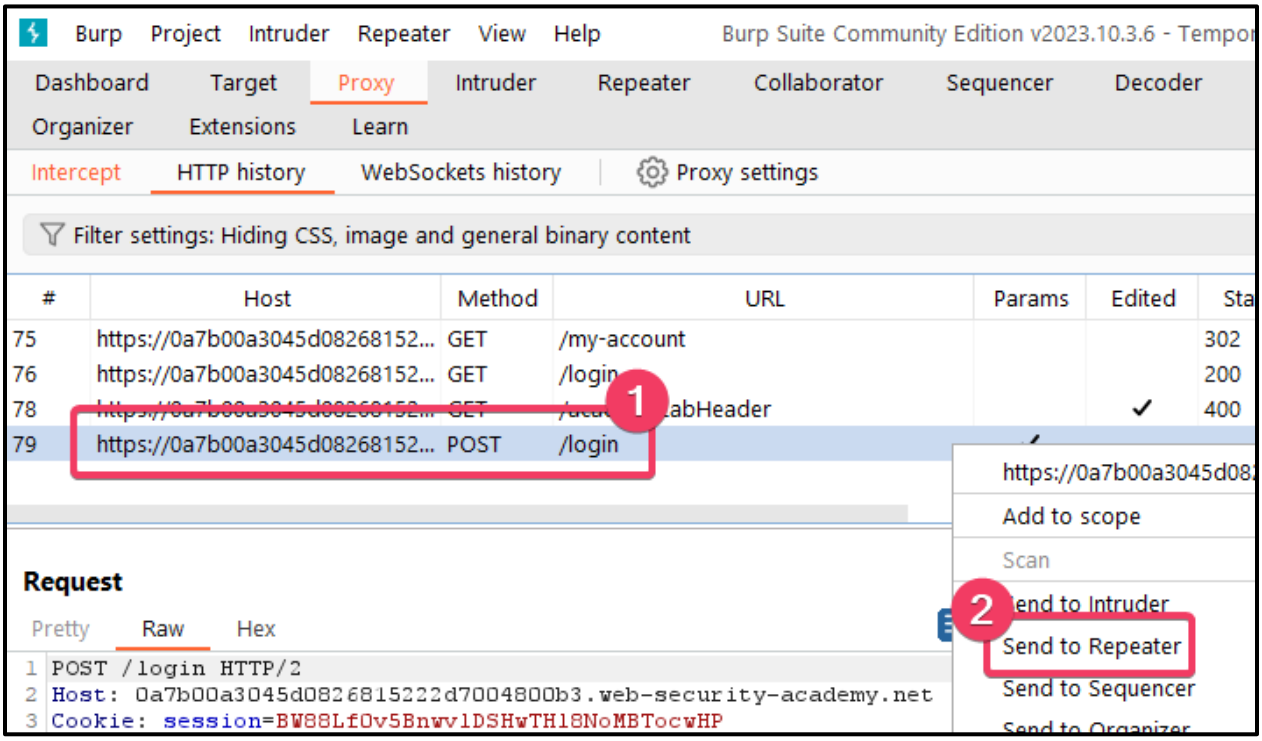
This lab contains a SQL injection vulnerability in the login function.

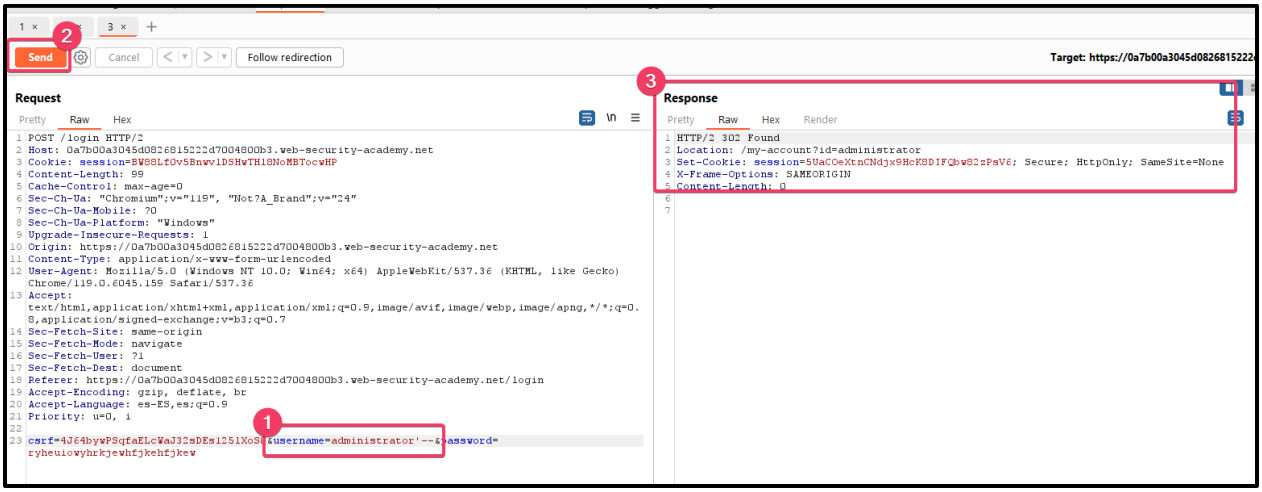
To solve the lab, perform a SQL injection attack that logs in to the application as the administrator user.

1. Pondremos cualquier usuario y cualquier contraseña, ya que lo que queremos es interceptar el login.



2. Lo llevamos a Repeater y cambiamos el parámetro username.





Lab: SQL injection vulnerability allowing login bypass

APPRENTICE

LAB Solved

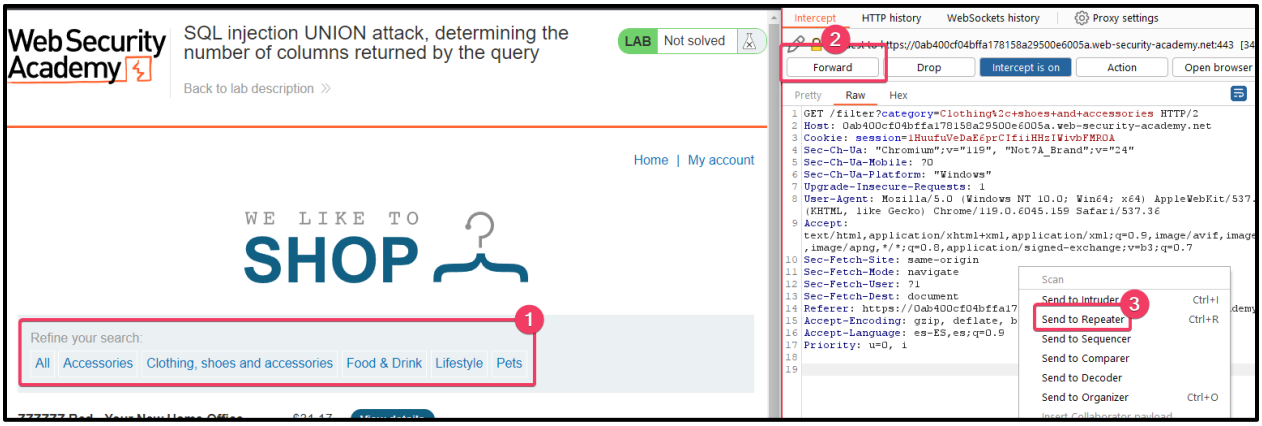
Determining the number of columns required (3 of 4)

Lab: SQL injection UNION attack, determining the number of columns returned by the query

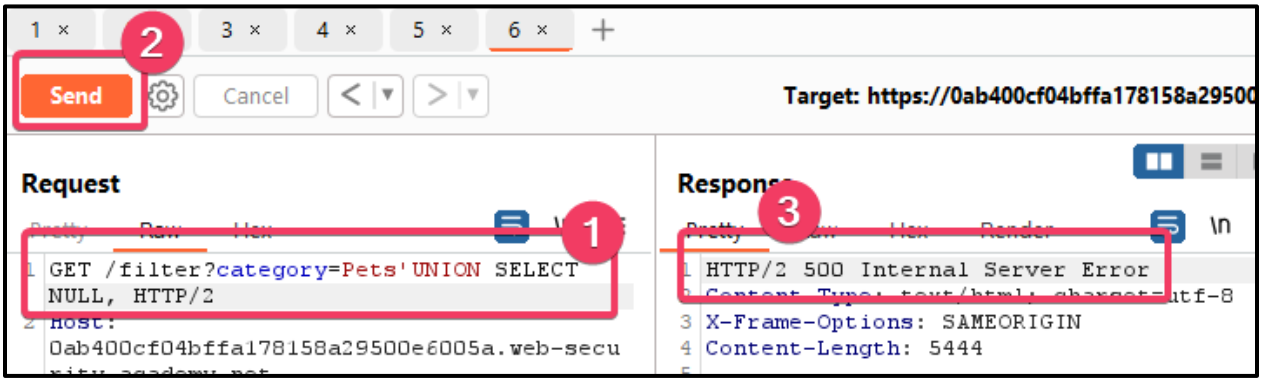
This lab contains a SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. The first step of such an attack is to determine the number of columns that are being returned by the query. You will then use this technique in subsequent labs to construct the full attack.

To solve the lab, determine the number of columns returned by the query by performing a SQL injection UNION attack that returns an additional row containing null values.

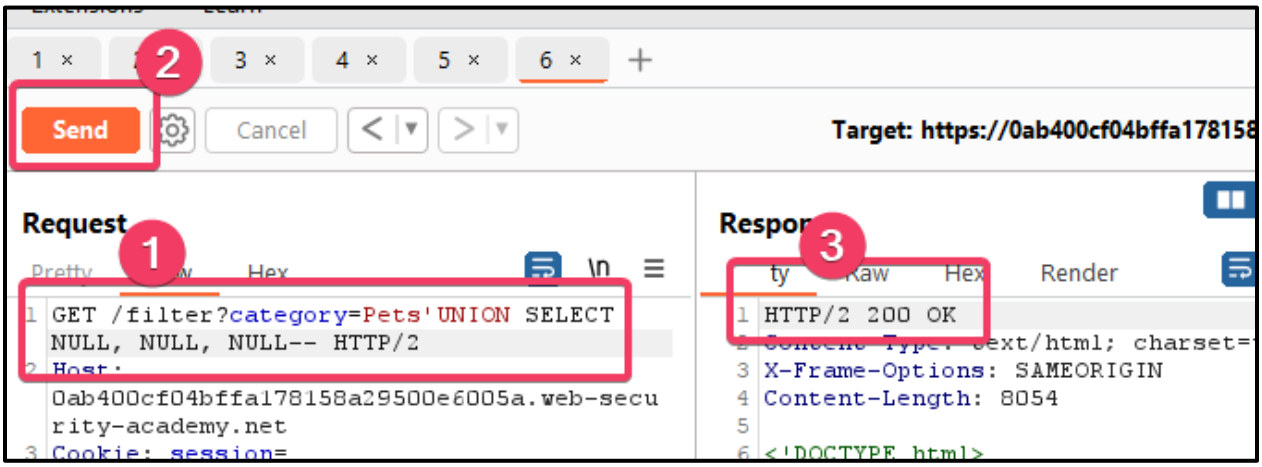
1. Vamos a hacer clic en alguna categoría para poder interceptar la petición y daremos a forward, como podemos observar ya tendremos la petición, en mi caso de clothing, shoes and accessories y lo mandaremos a Repeater.



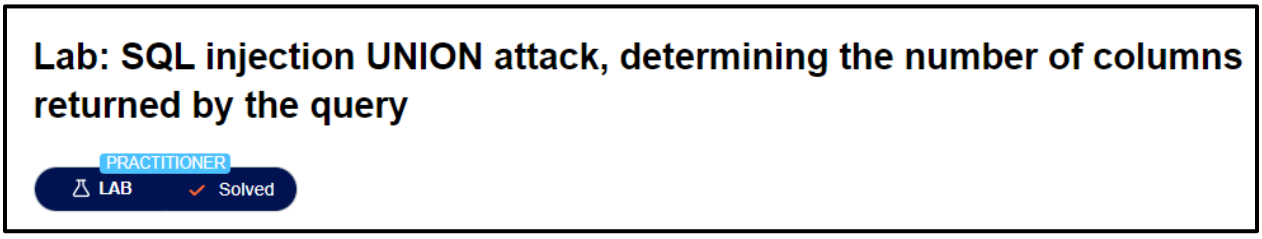
2. Probando cuantos NULL nos van a hacer falta.



Probando con 1 NULL



Respuesta correcta con 3 NULL.



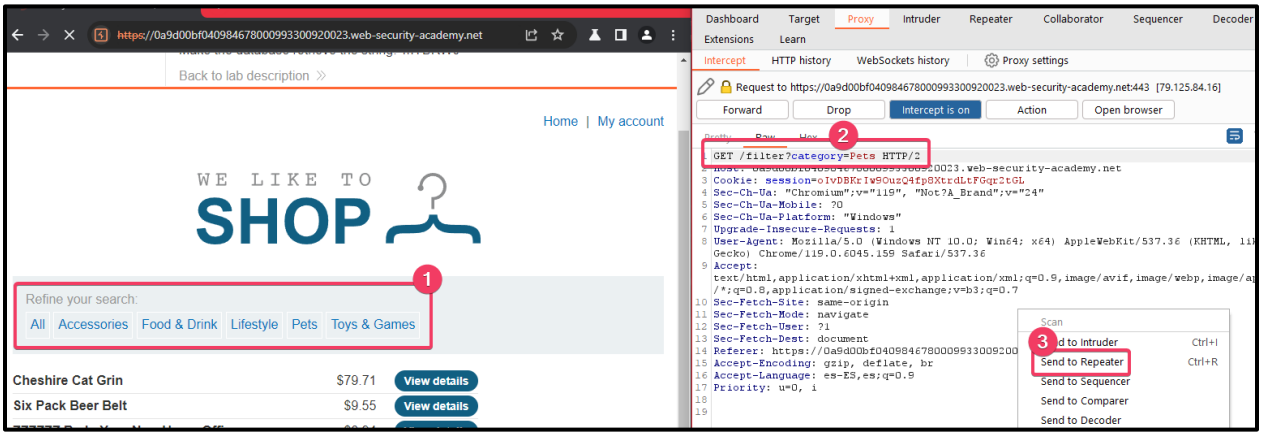
Finding columns with a useful data type (2 of 2)

Lab: SQL injection UNION attack, finding a column containing text

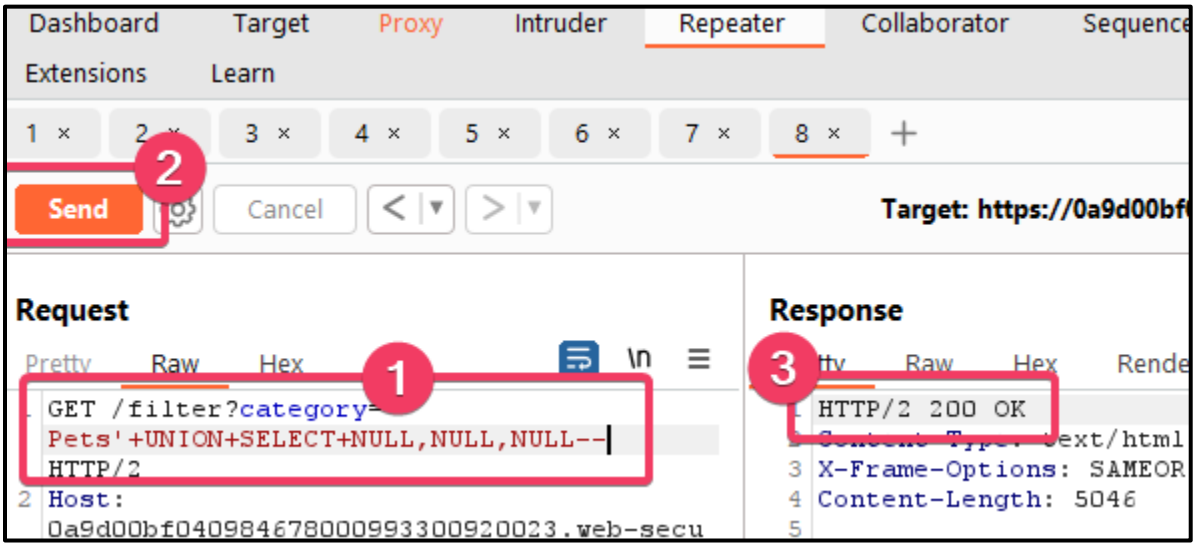
This lab contains a SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. To construct such an attack, you first need to determine the number of columns returned by the query. You can do this using a technique you learned in a previous lab. The next step is to identify a column that is compatible with string data.

The lab will provide a random value that you need to make appear within the query results. To solve the lab, perform a SQL injection UNION attack that returns an additional row containing the value provided. This technique helps you determine which columns are compatible with string data.

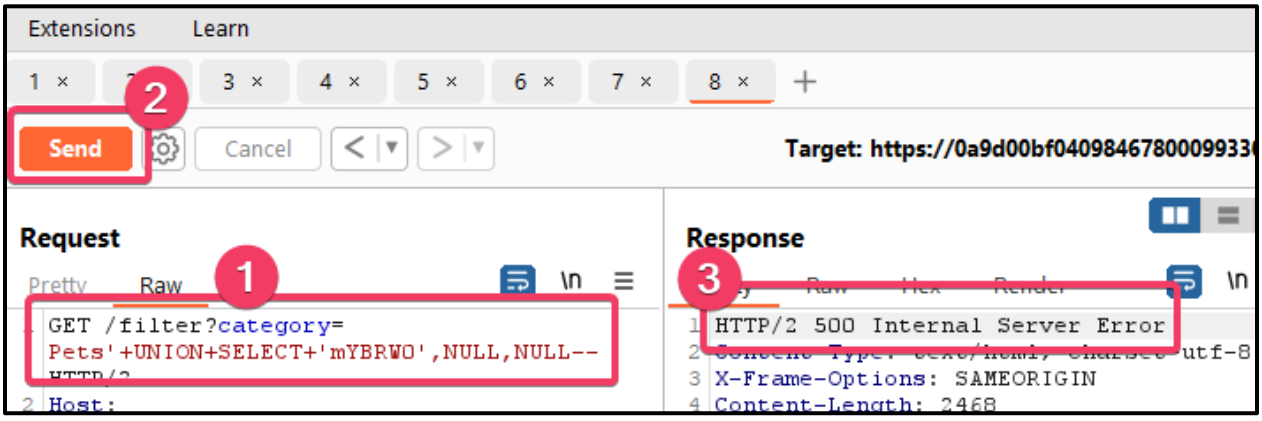
1. Haremos clic en alguna de las categorías y lo mandaremos a Repeater.



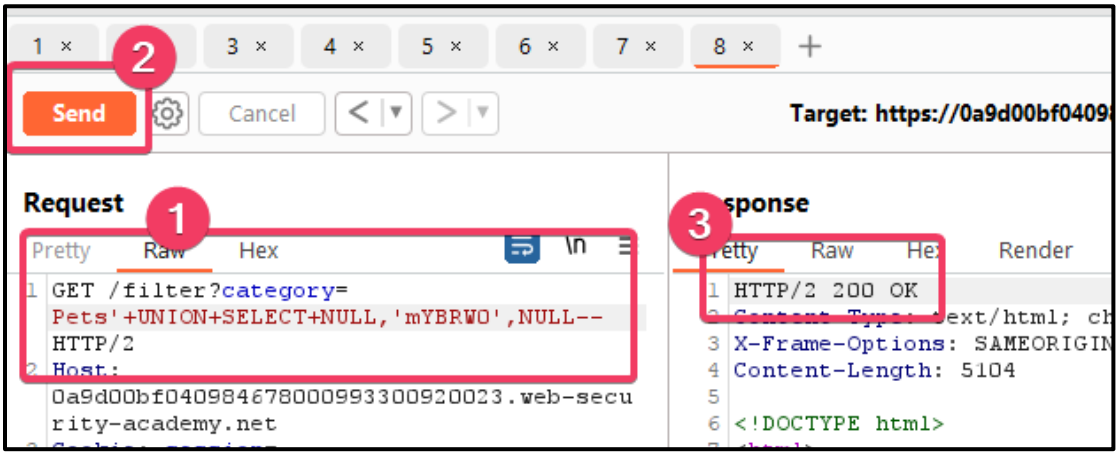
2. Nos hemos dado cuenta que el query nos da 3 columnas comprobando de una en una.



3. Comprobamos meter el ejemplo random que da el ejercicio en cada NULL, hasta que nos dé el resultado deseado.

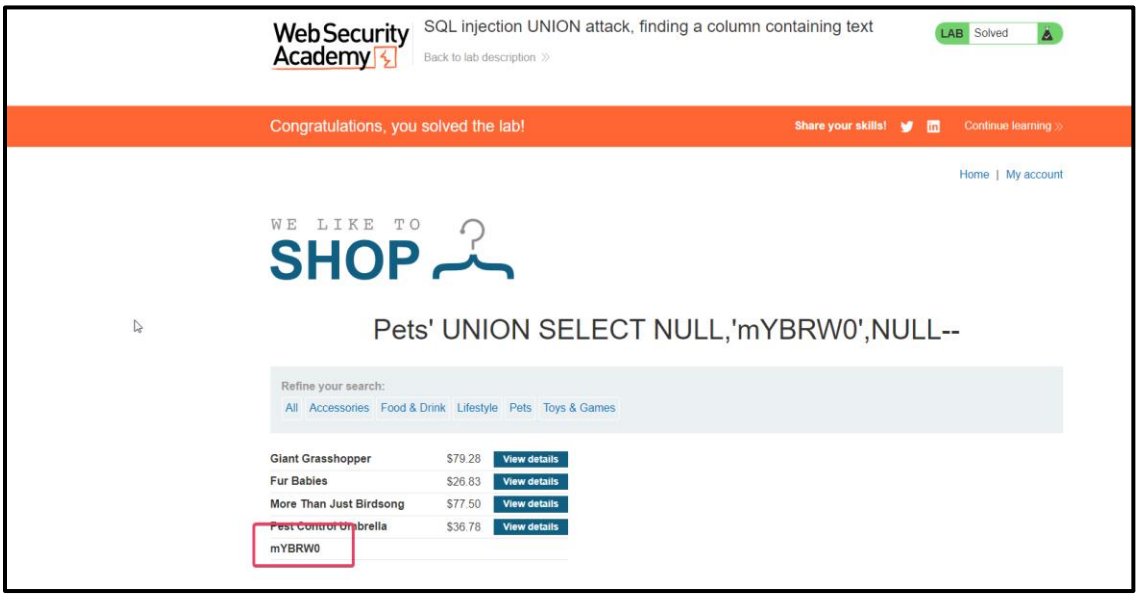


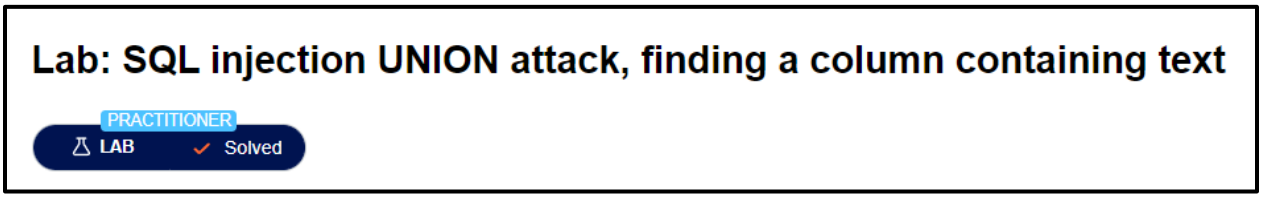
Primera posición



La segunda posición es la correcta

4. Vista desde el navegador.





Using a SQL injection UNION attack to retrieve interesting data (2 of 2)

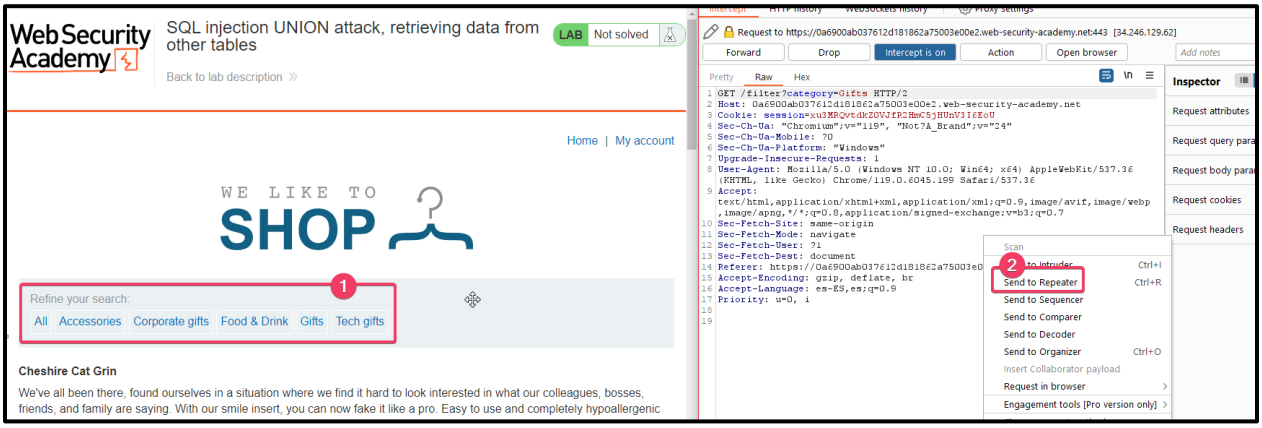
Lab: SQL injection UNION attack, retrieving data from other tables

This lab contains a SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. To construct such an attack, you need to combine some of the techniques you learned in previous labs.

The database contains a different table called users, with columns called username and password.

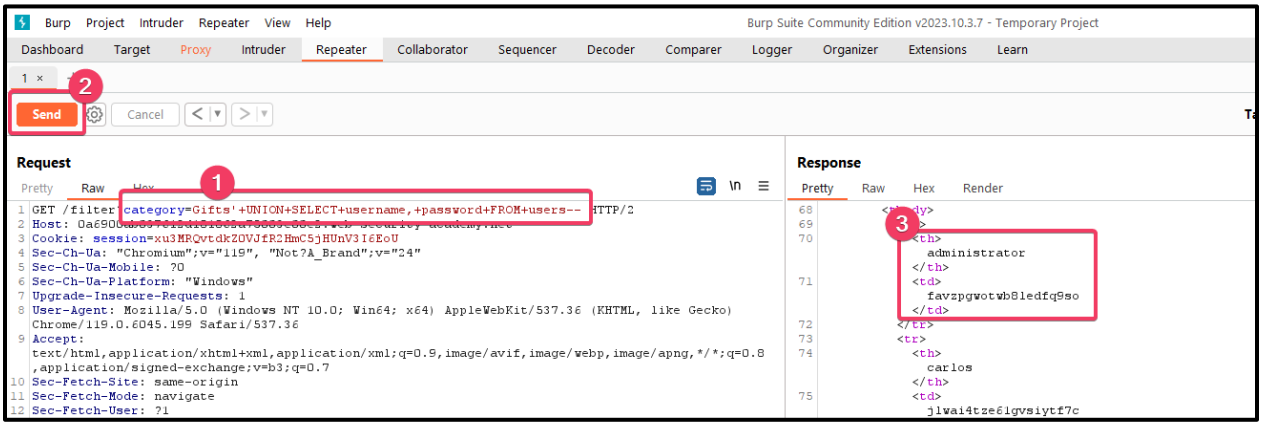
To solve the lab, perform a SQL injection UNION attack that retrieves all usernames and passwords, and use the information to log in as the administrator user.

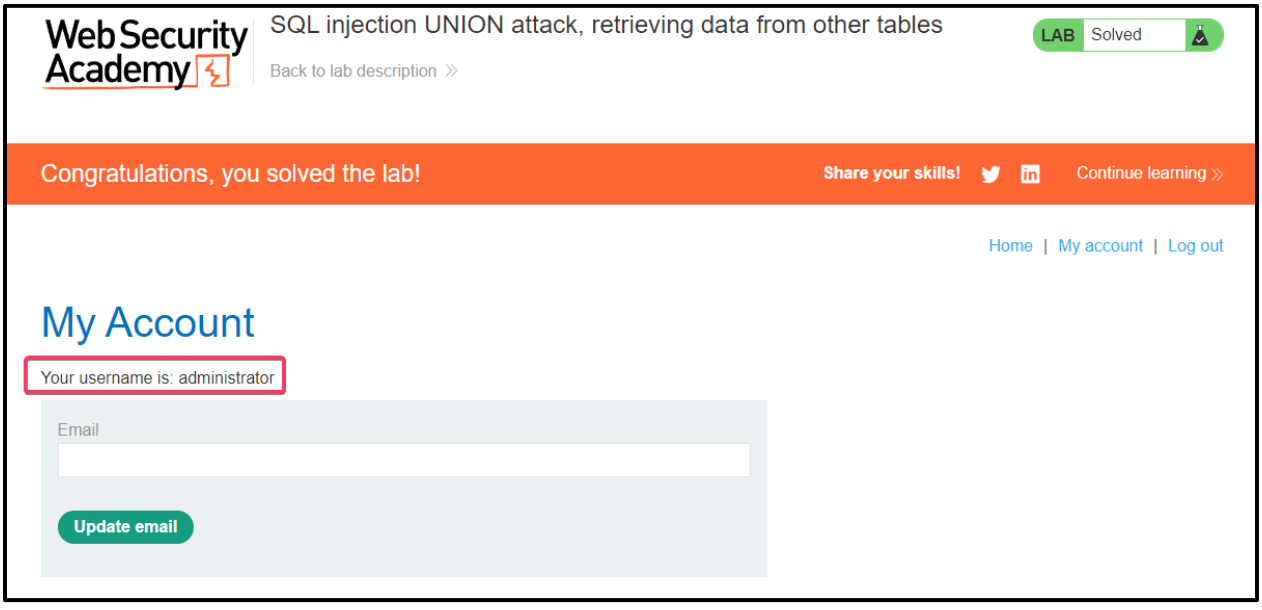
1. Interceptamos el request de la categoría seleccionada y lo mandamos al Repeater.



2. Intentamos determinar el número de columnas que nos devuelve el query que contengan datos de texto.

3. Y ahora que sabemos el número de columnas que devuelven datos de texto sacaremos los contenidos de la tabla Users.





Retrieving multiple values within a single column (2 of 2)

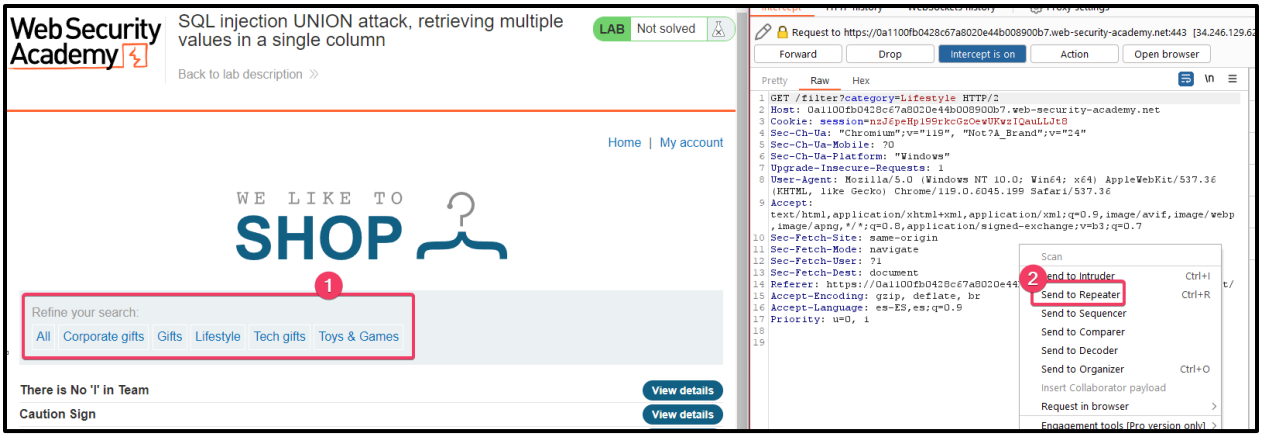
Lab: SQL injection UNION attack, retrieving multiple values in a single column

This lab contains a SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables.

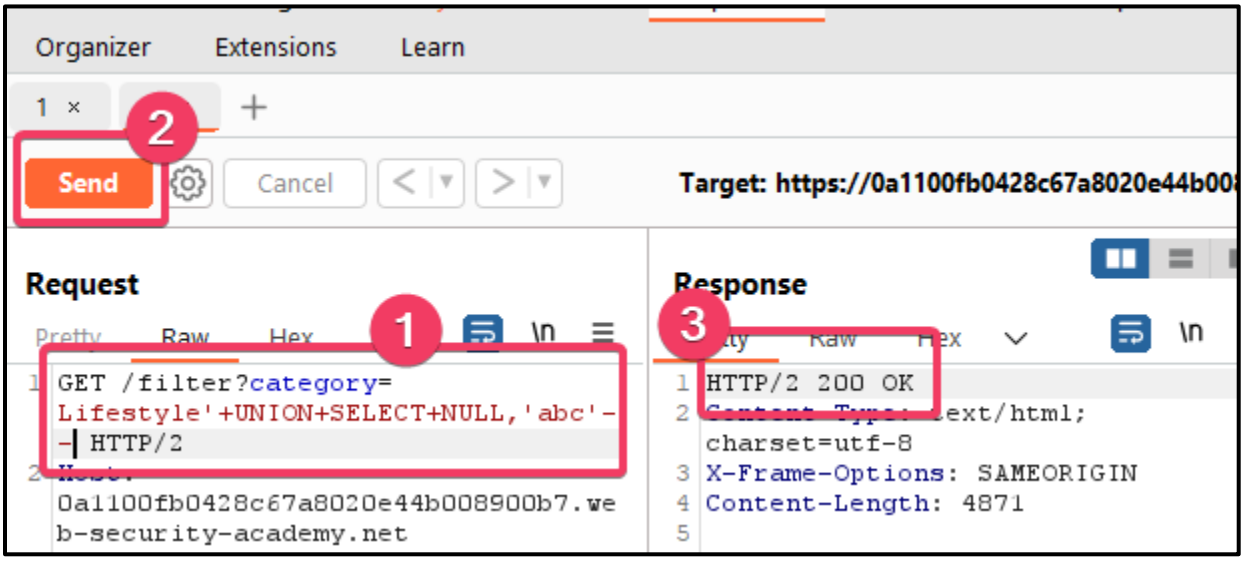
The database contains a different table called users, with columns called username and password.

To solve the lab, perform a SQL injection UNION attack that retrieves all usernames and passwords, and use the information to log in as the administrator user.

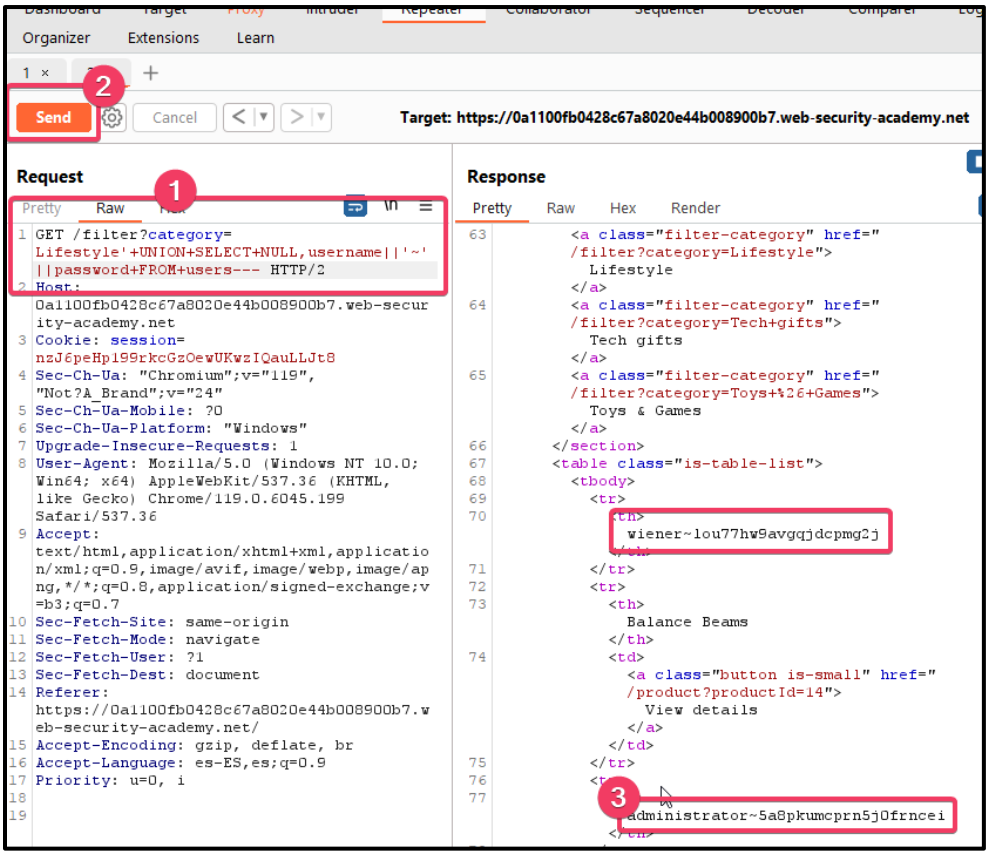
1. Interceptamos el request de category y lo mandamos al Repeater.

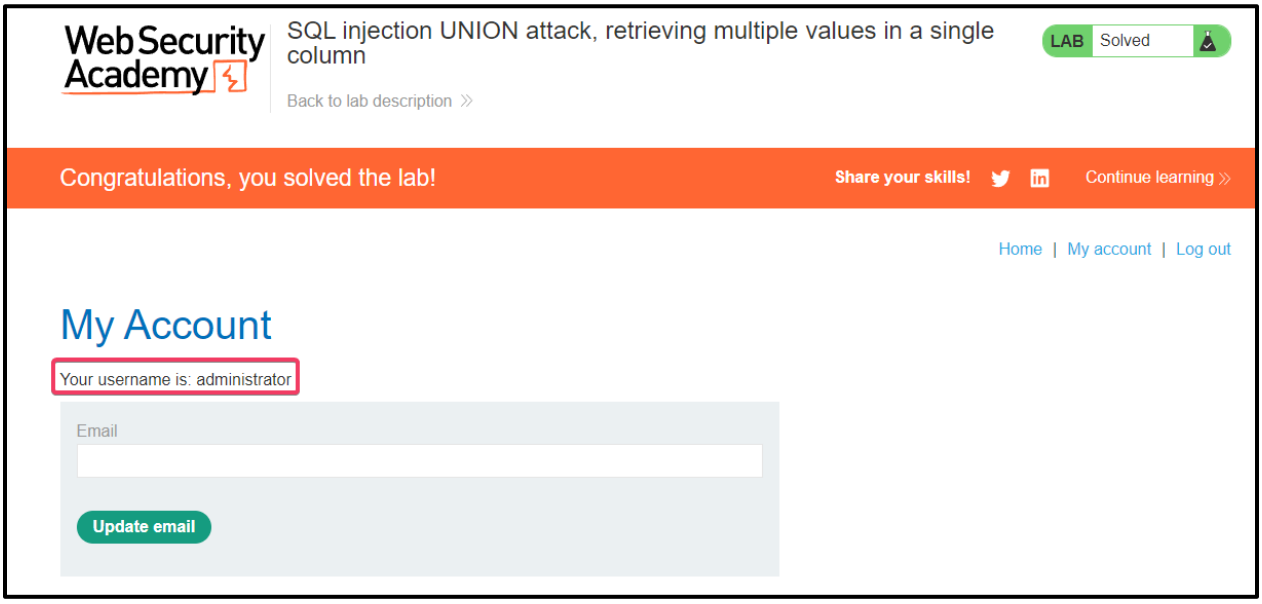


2. Como llevamos haciendo, vamos a determinar el número de columnas.



3. Ahora procederemos a sacar el contenido de la tabla users.




Web Security Academy
SQL injection UNION attack, retrieving multiple values in a single column
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: administrator
Email
Update email

Examining the database (5 of 5)

Lab: SQL injection attack, listing the database contents on non-Oracle databases

This lab contains a SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables.

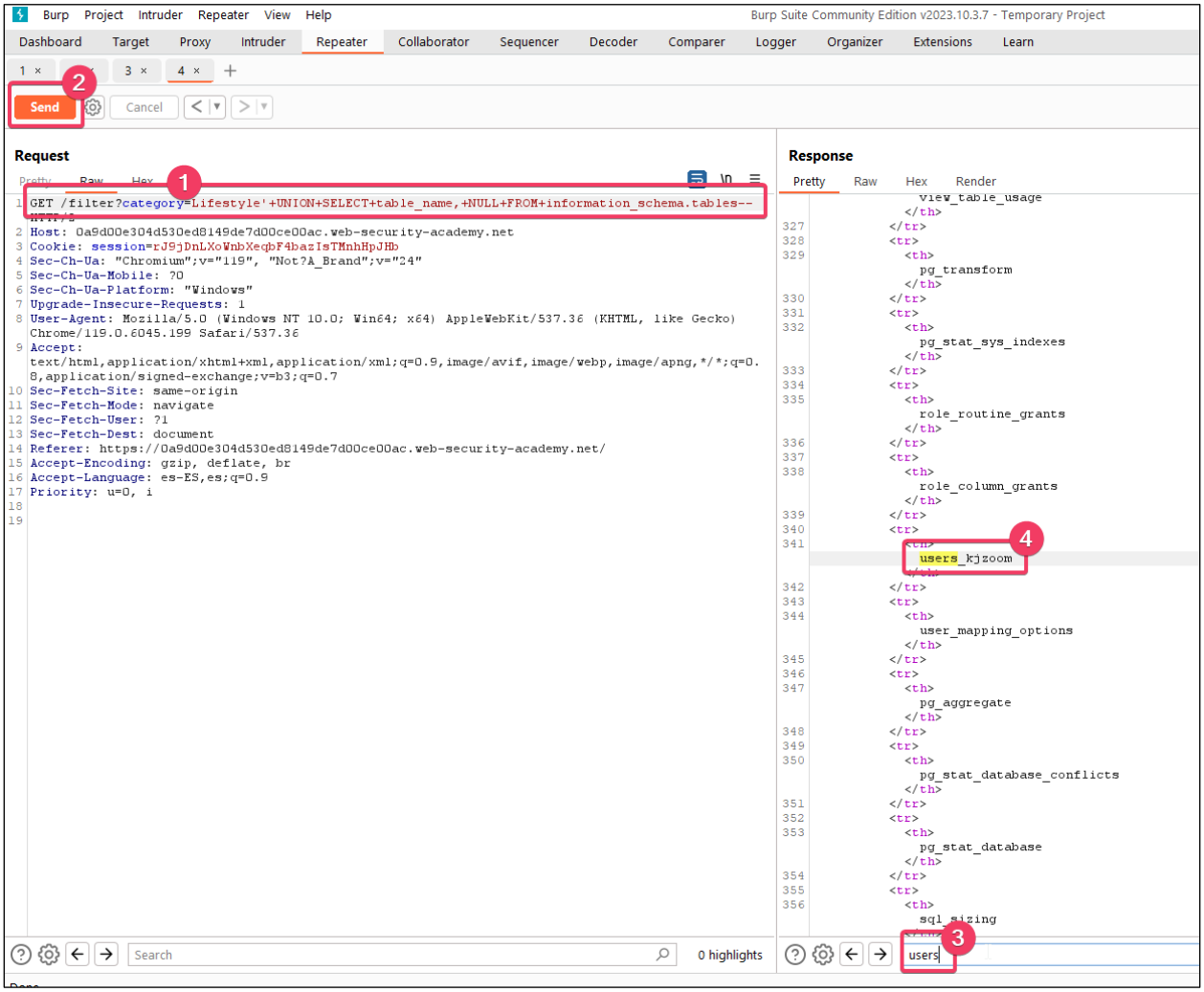
The application has a login function, and the database contains a table that holds usernames and passwords. You need to determine the name of this table and the columns it contains, then retrieve the contents of the table to obtain the username and password of all users.

To solve the lab, log in as the administrator user.

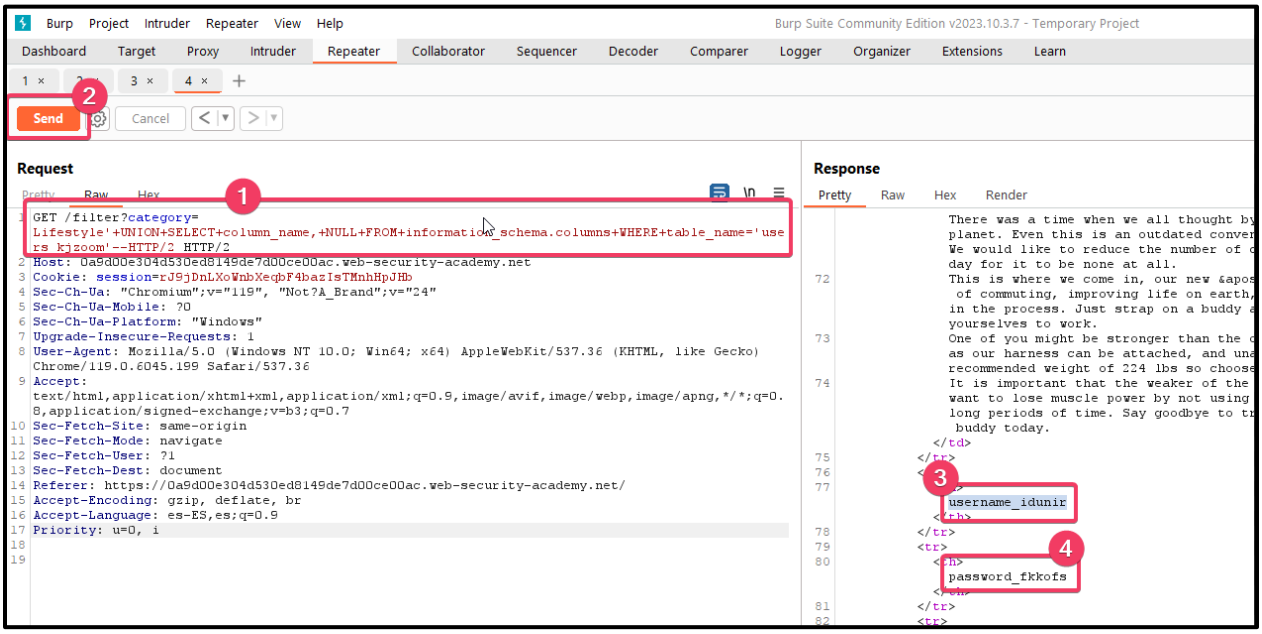
1. Interceptamos y lo mandamos a Repeater.

2. Ahora vamos a determinar 1 por 1 el número de columnas que nos da el query.

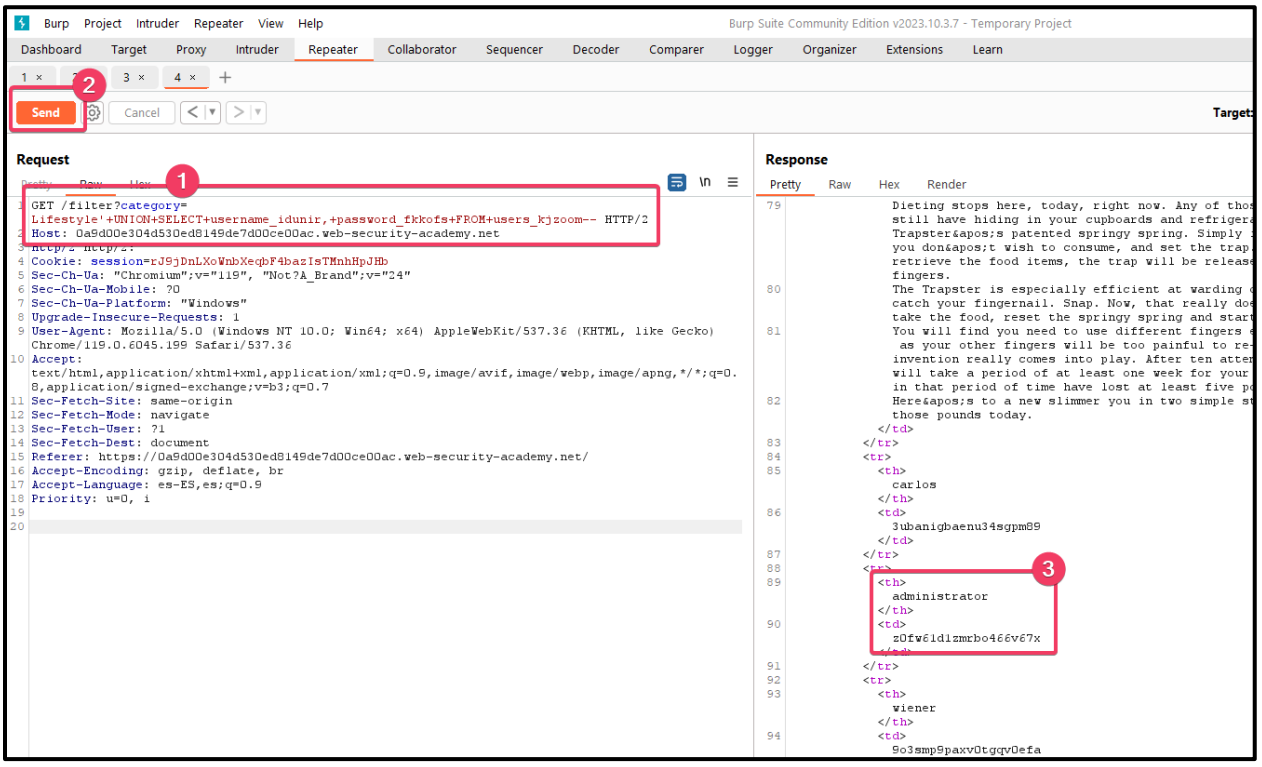
3. Buscamos la tabla que tiene las credenciales de los usuarios.

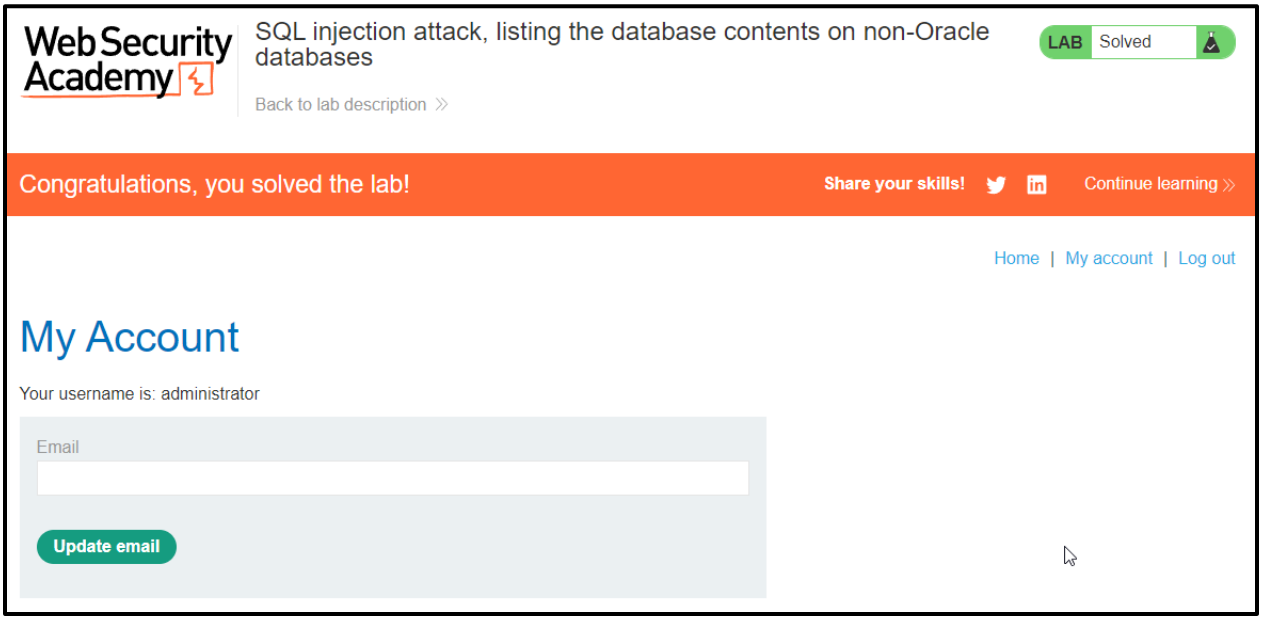


4. Buscamos el nombre de la columna que contiene nombres de usuario y contraseñas.



5. Reemplazamos username, password (columnas) y users (tabla).




Web Security Academy
SQL injection attack, listing the database contents on non-Oracle databases
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: administrator
Email
Update email

Exploiting blind SQL injection by triggering conditional responses (4 of 4)

Lab: Blind SQL injection with conditional responses

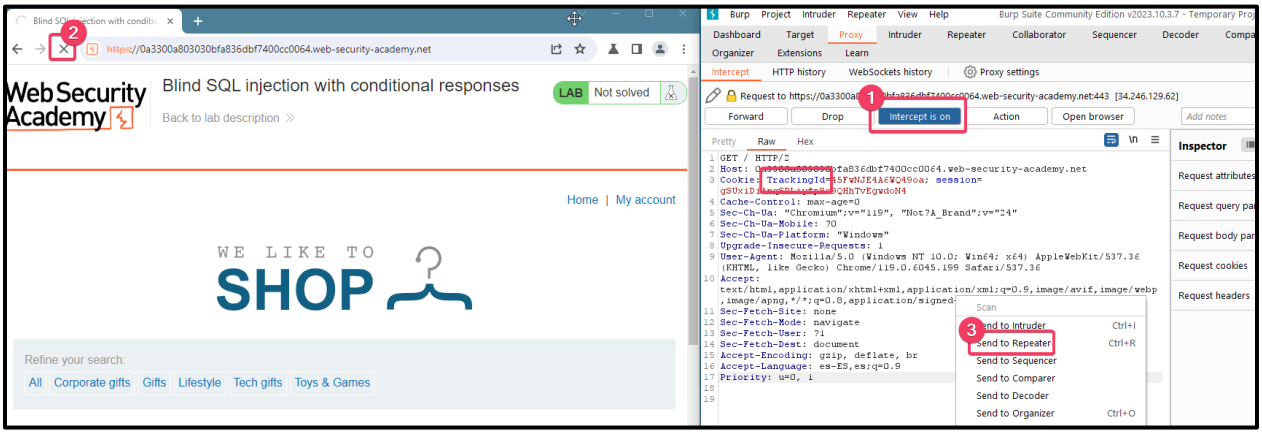
This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie.

The results of the SQL query are not returned, and no error messages are displayed. But the application includes a "Welcome back" message in the page if the query returns any rows.

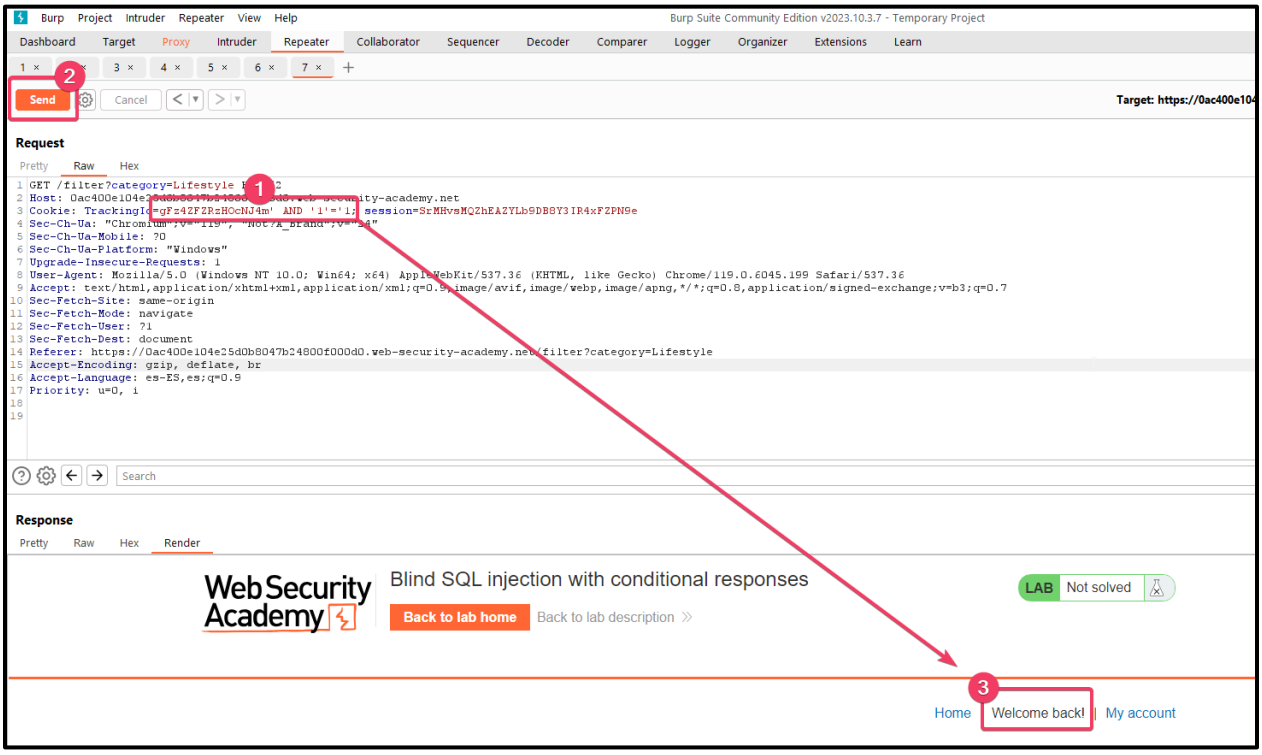
The database contains a different table called users, with columns called username and password. You need to exploit the blind SQL injection vulnerability to find out the password of the administrator user.

To solve the lab, log in as the administrator user.

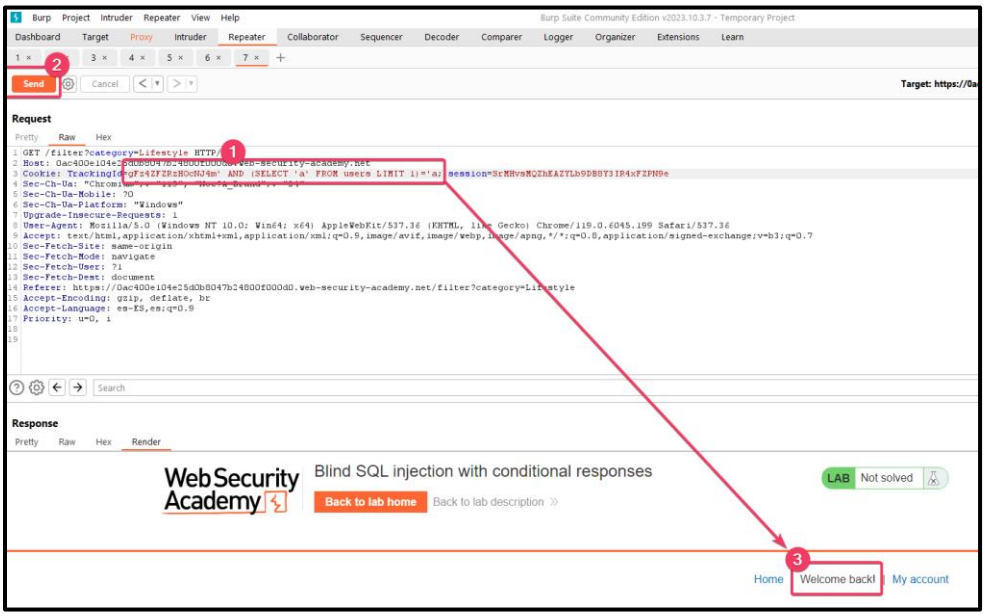
1. Vamos a activar la interceptación y vamos a relogear la página y mandamos lo interceptado al Repeater.



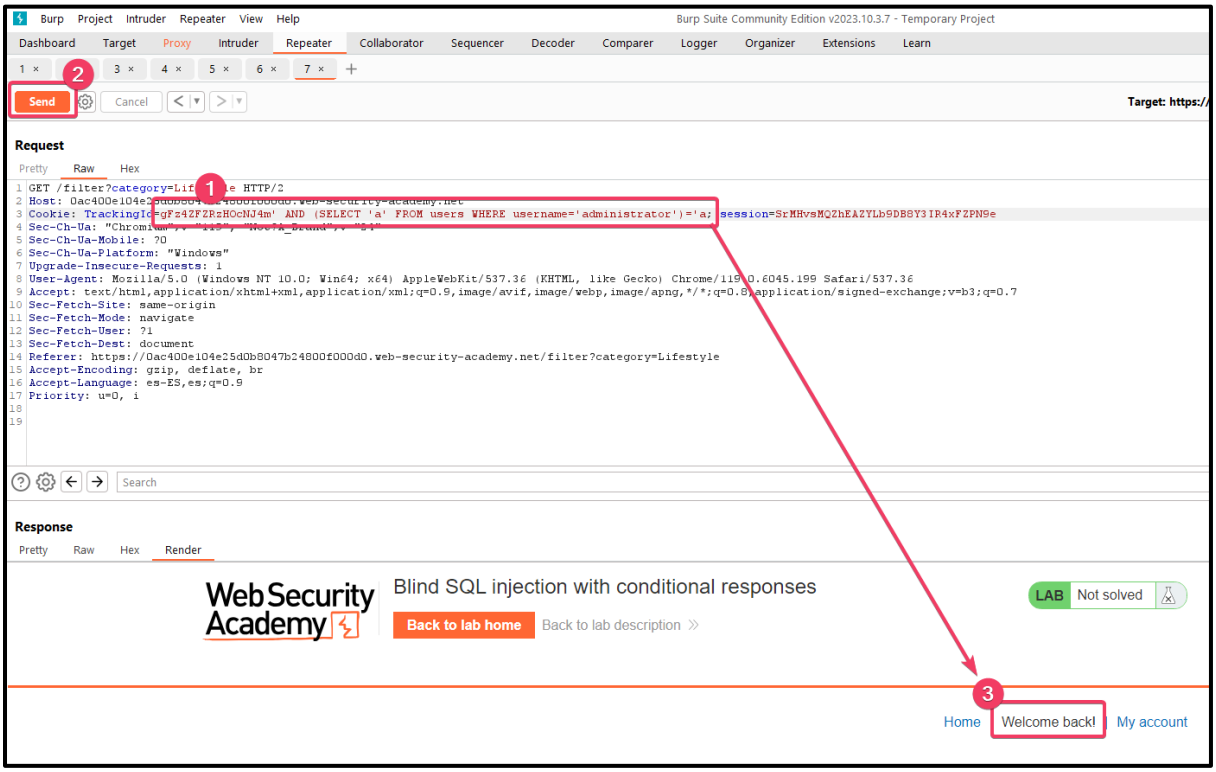
2. Vamos a comprobar como podemos probar una única condición booleana e interferir en el resultado.



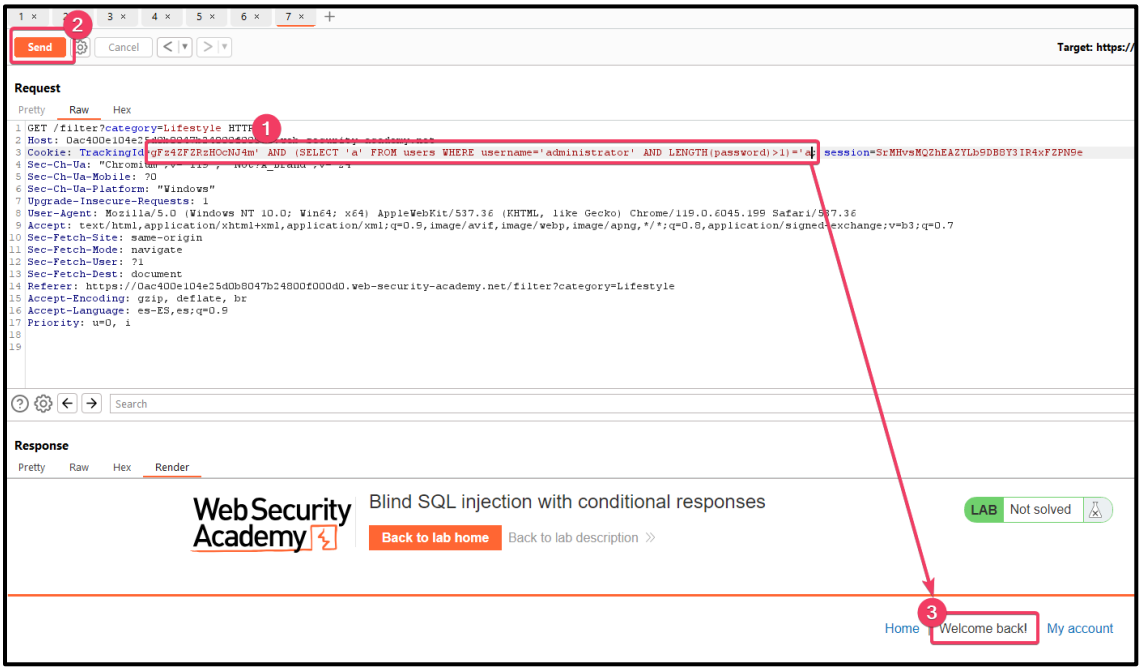
3. Comprobación de si hay una tabla llamada users.



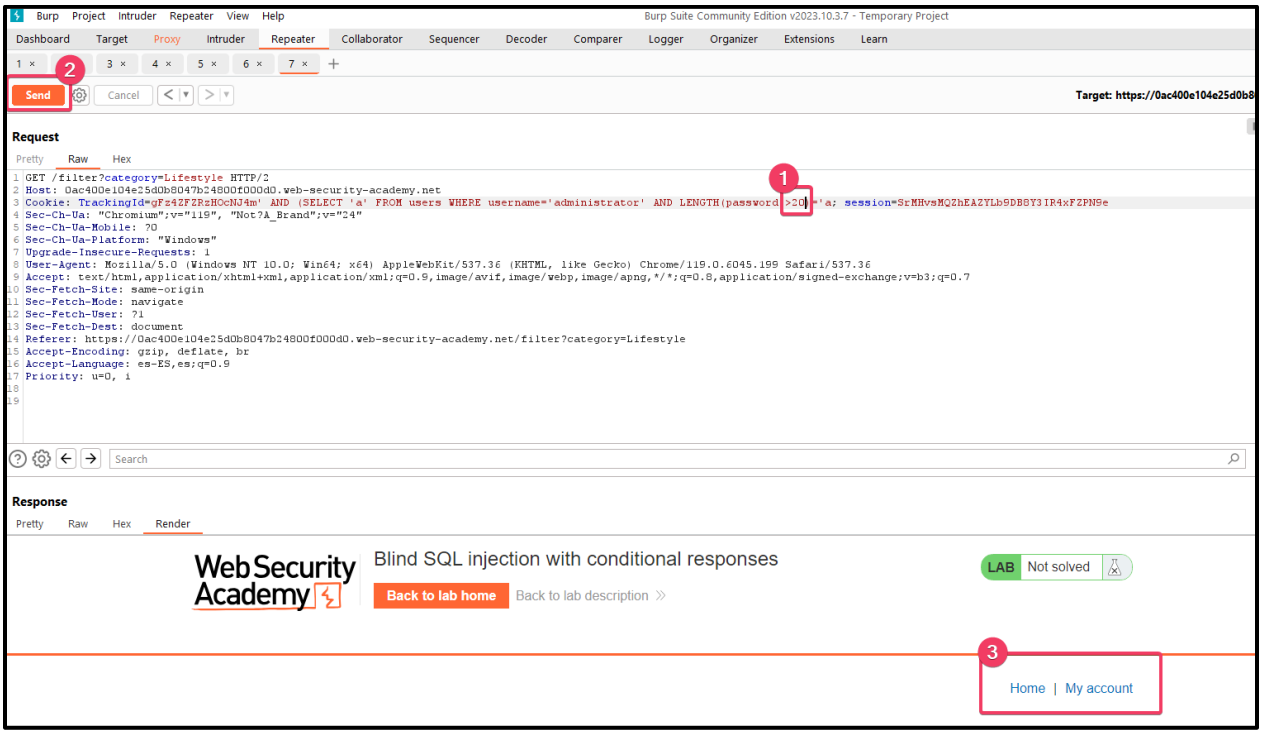
4. Comprobaremos si hay una tabla llamada administrador. (obviamente cada vez que nos encontremos un Welcome back! Significa que la condición es cierta)

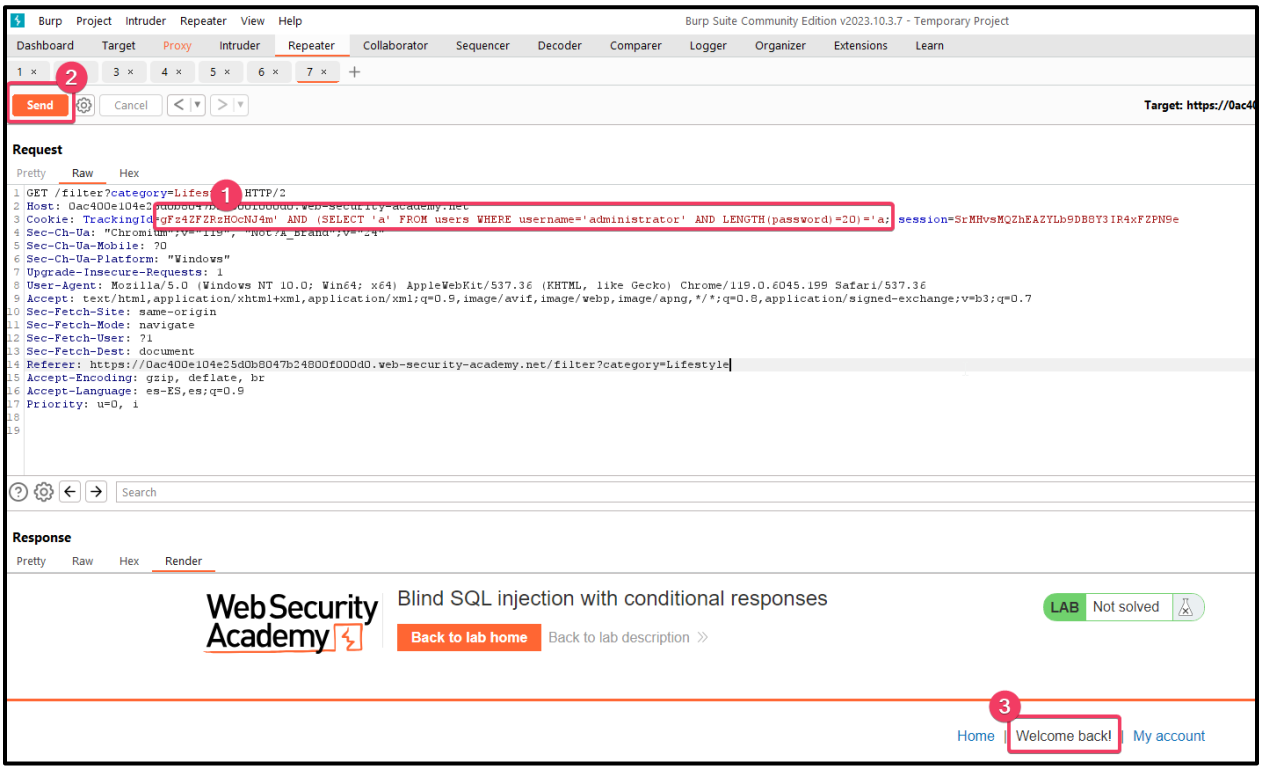


5. Ahora comprobaremos cuantos caracteres tiene la contraseña del user administrador. (Más de un carácter)



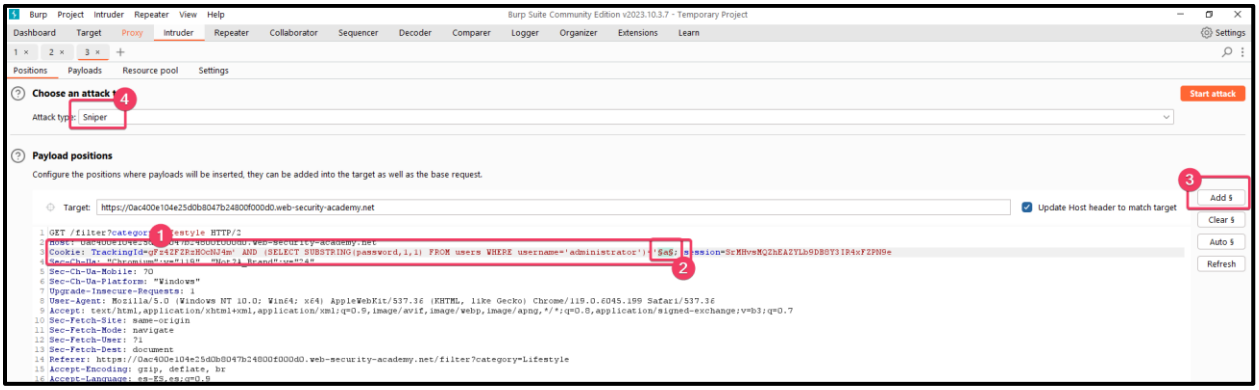
6. Ahora voy a ir cambiando el número 1 hasta que no aparezca el Welcome back, y asi determinar la longitud de la contraseña. En este caso la longitud es 20. Y voy a hacer un paso extra con un =20 para ver que de verdad es cierto.



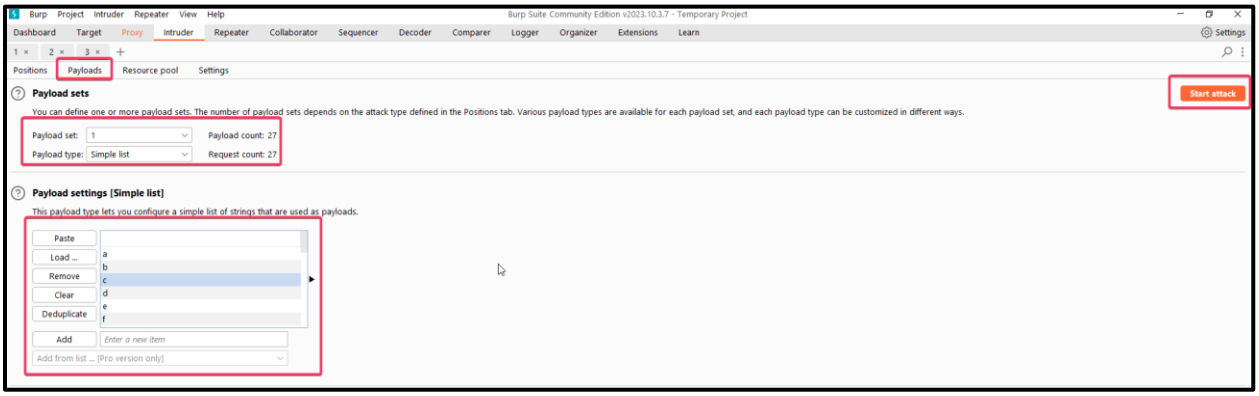


7. Ahora vamos a empezar a sacar la contraseña de administrador.

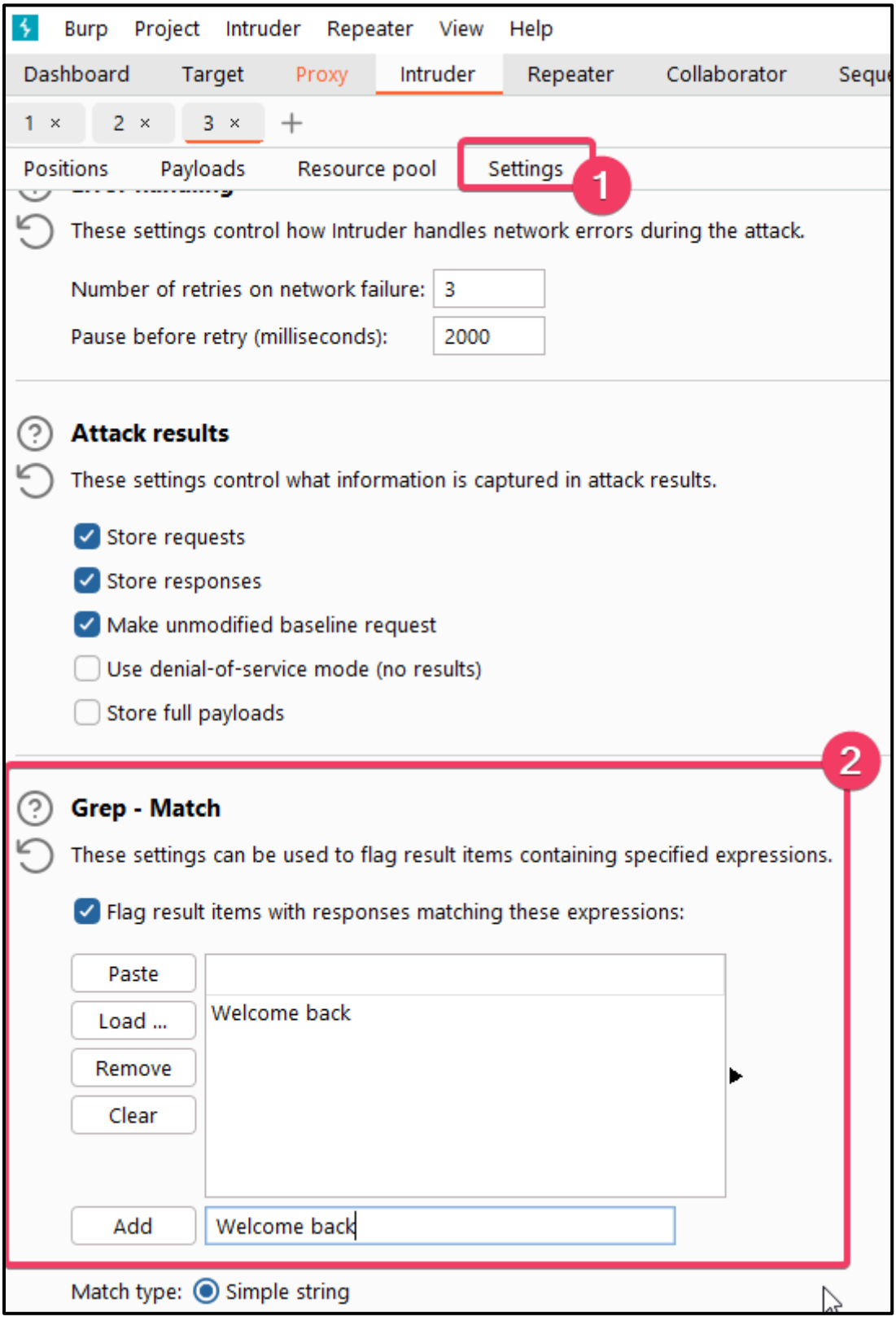
En intruder positions cambiamos el Id y señalamos la a con add para que nos busque la contraseña.



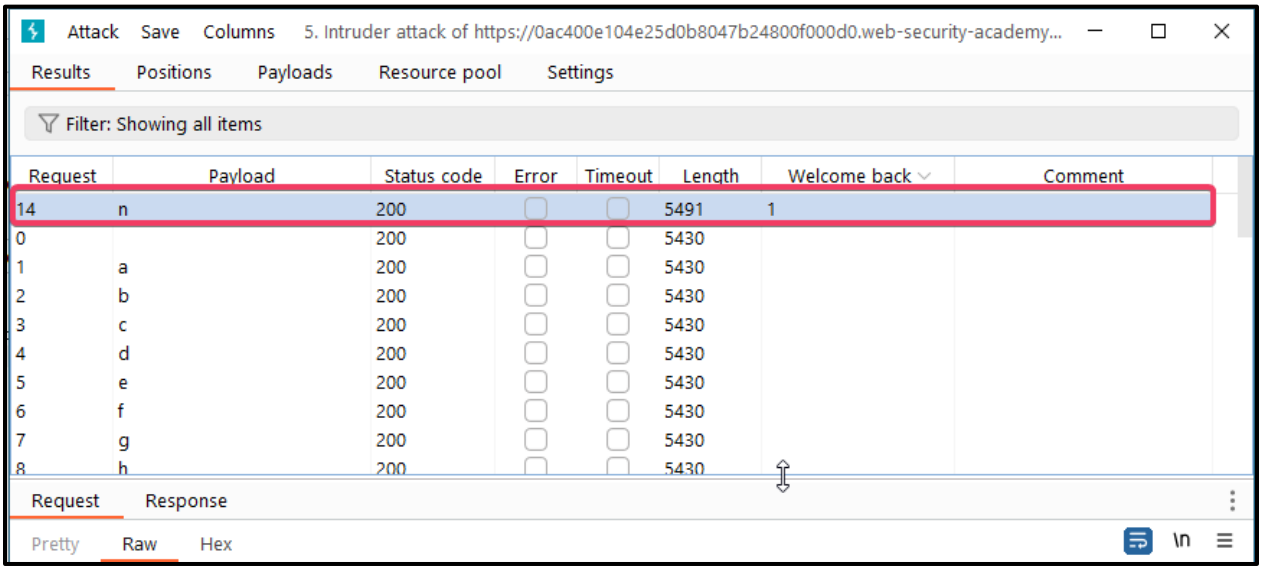
En Payloads tendremos que poner simple list y en settings añadiremos letras de la a-z y números del 0-9.



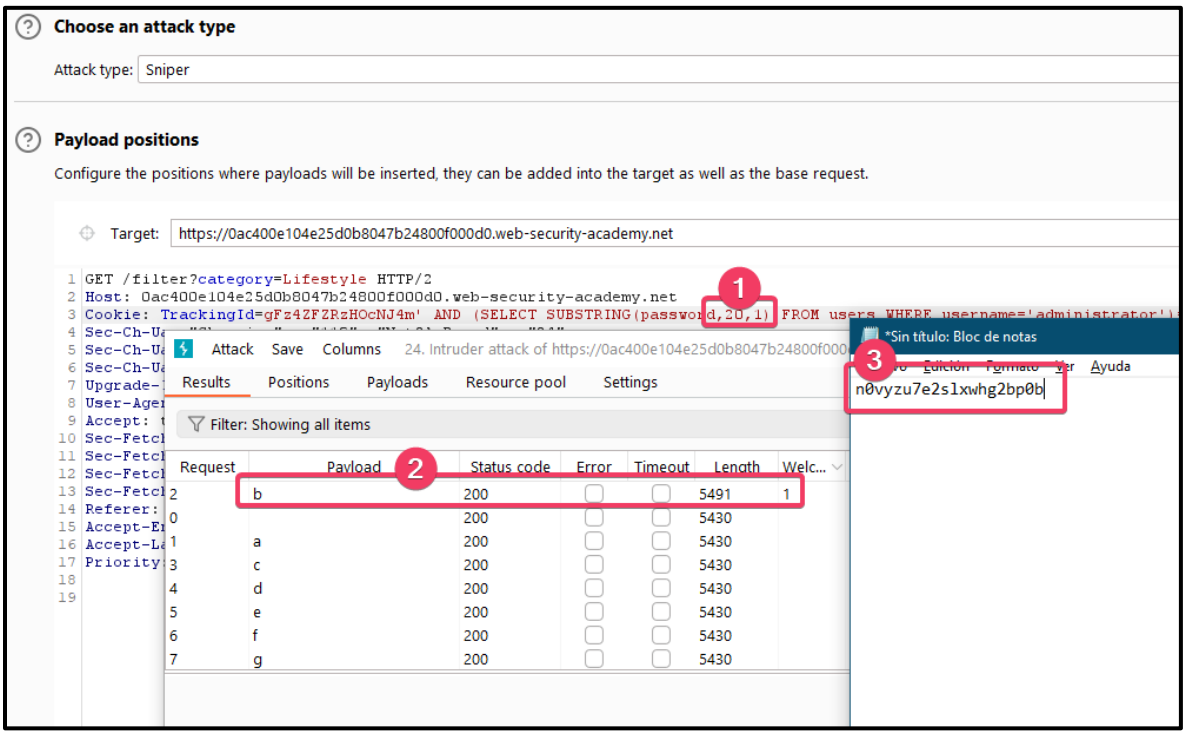
En Settings añadiremos en Grep-Match “Welcome Back”

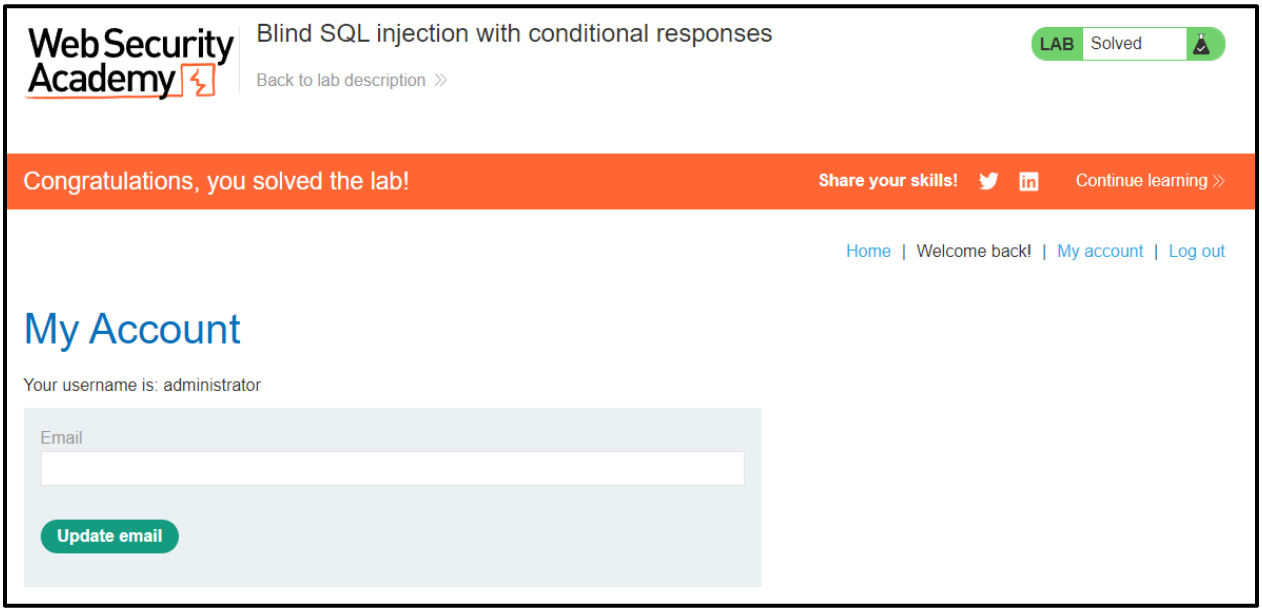


Hacemos el ataque y podemos observar que la primera letra de la contraseña es una "n".



8. Ahora tendremos que cambiar todo el rato la parte de la posición (rojo): **TrackingId=gFz4ZFZRzHOcNJ4m' AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username='administrator')='§a§,** en lugar de 1 pondremos 2 para la segunda posición. Solo voy a poner captura del último carácter (20).





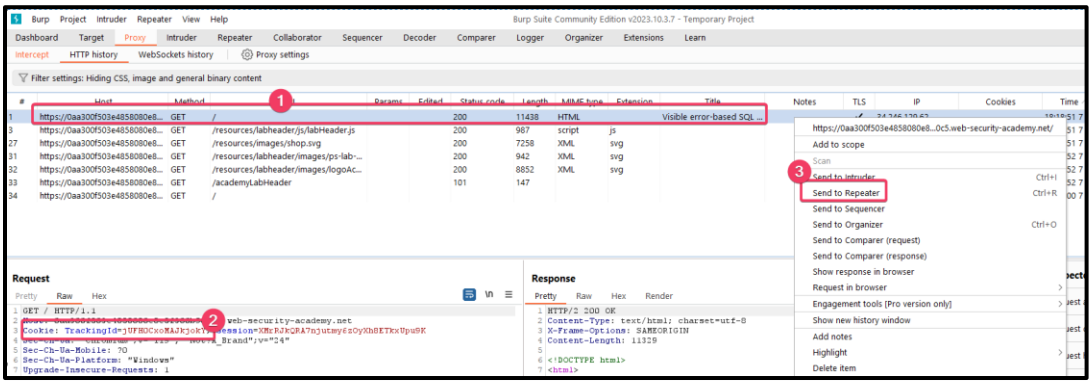
Error-based SQL injection (7 of 7)

Lab: Visible error-based SQL injection

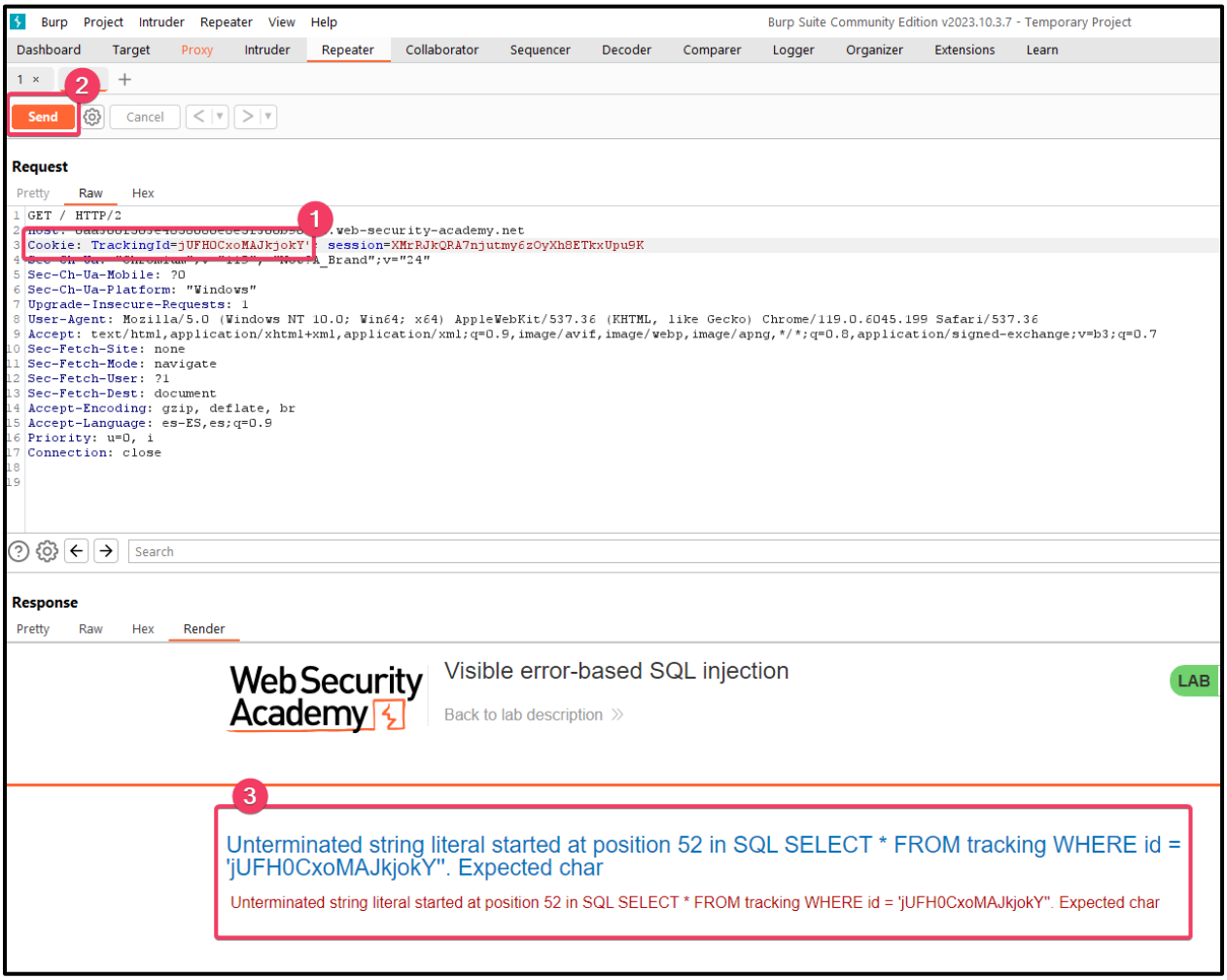
This lab contains a SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie. The results of the SQL query are not returned.

The database contains a different table called users, with columns called username and password. To solve the lab, find a way to leak the password for the administrator user, then log in to their account.

1. Buscaremos la request que contiene la TrackingId cookie y la mandaremos al Repeater.

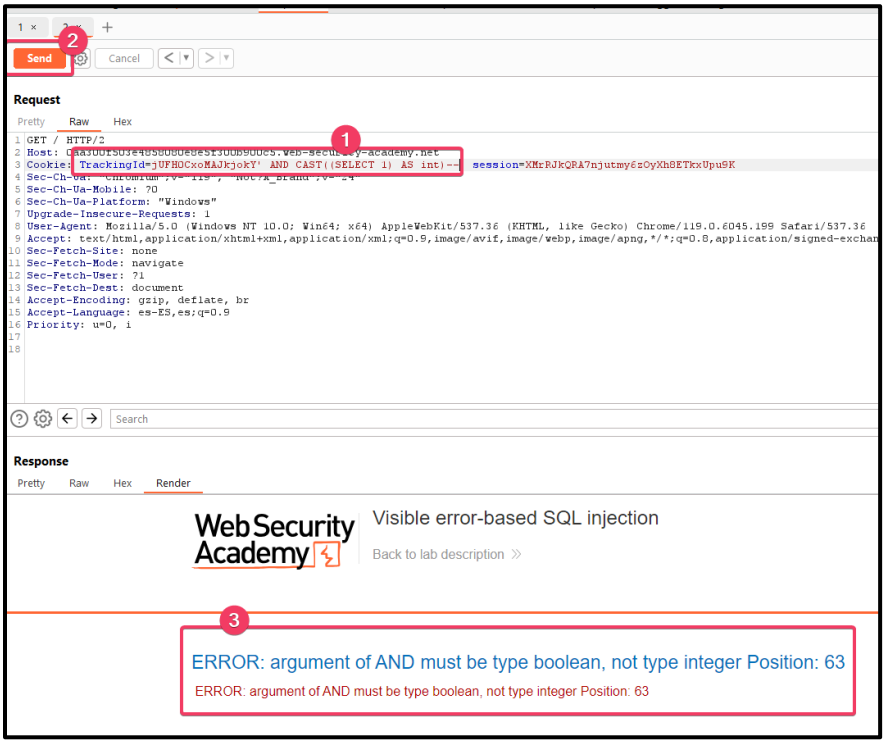


2. Probaremos con una comilla simple y observamos que la inyección aparece dentro de una cadena entre comillas simples.



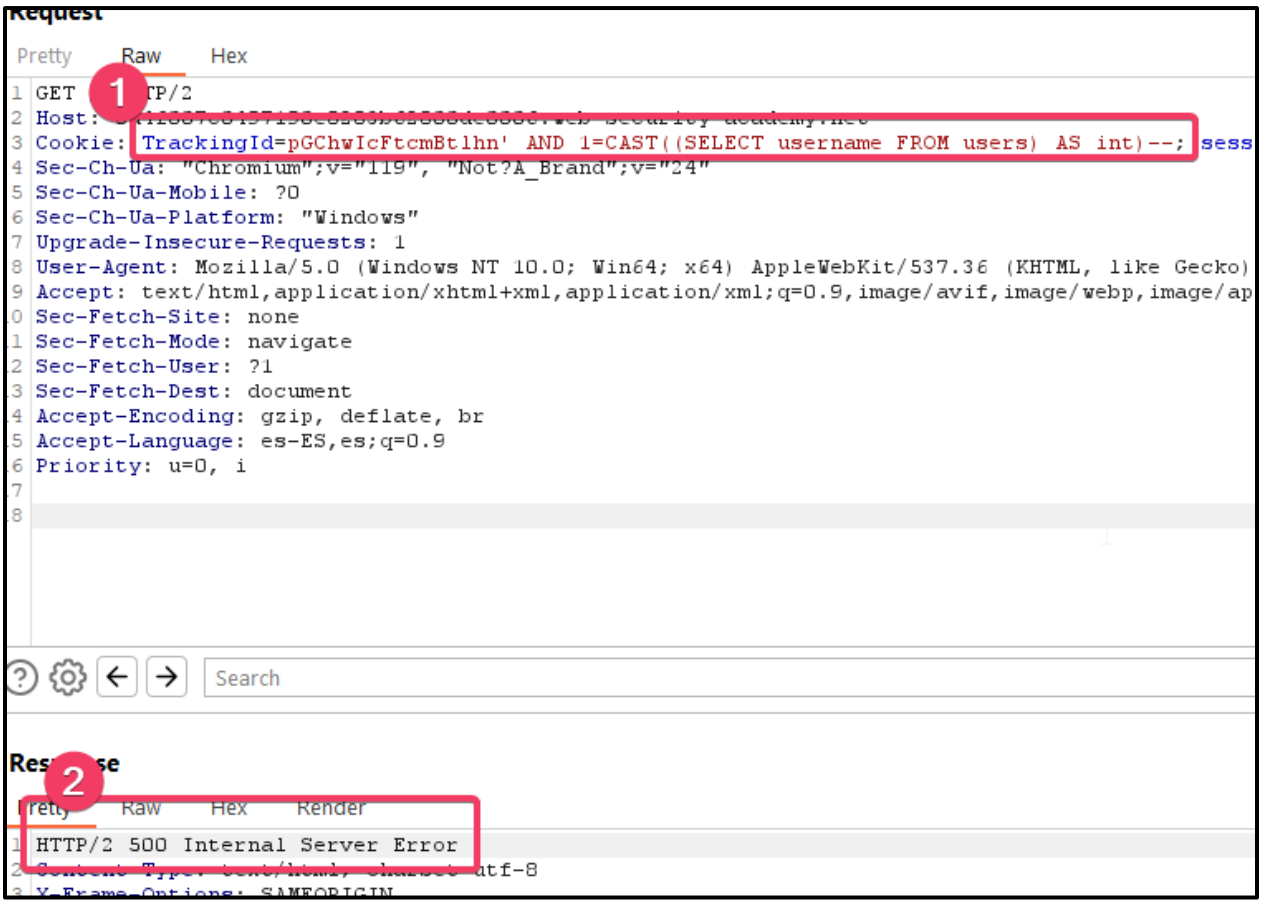
3. Ahora vamos a agregar caracteres de comentario. Y si no da error significará que la sintaxis de la consulta es válida.

4. Ahora vamos a incluir una subconsulta SELECT.

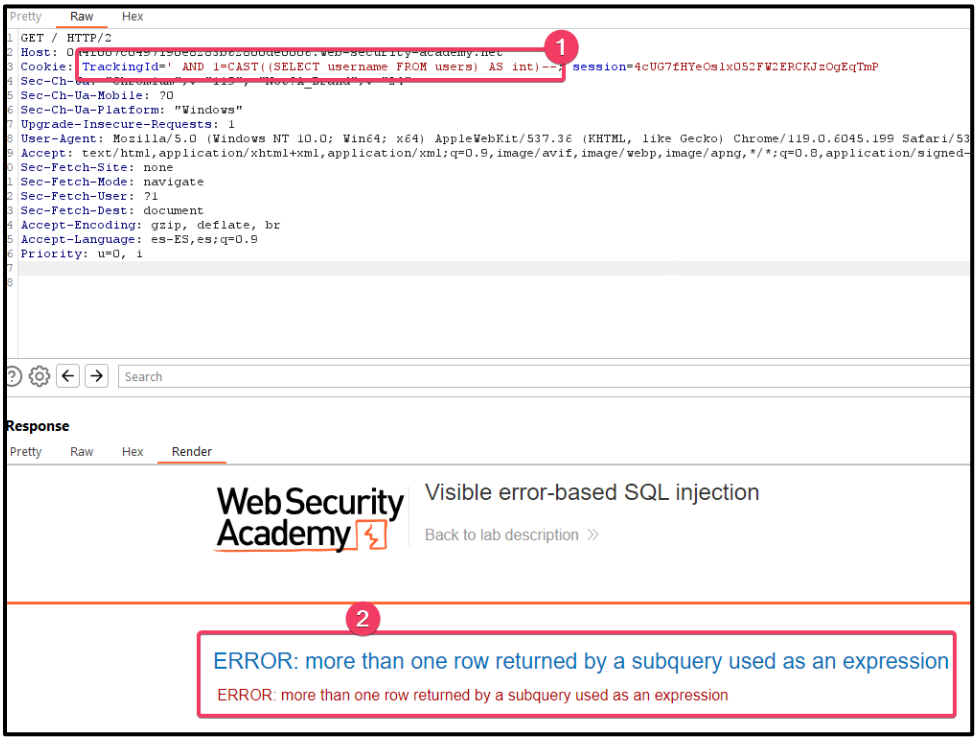


5. Vamos a agregar un operador de comparación. Y ya no nos dará error.

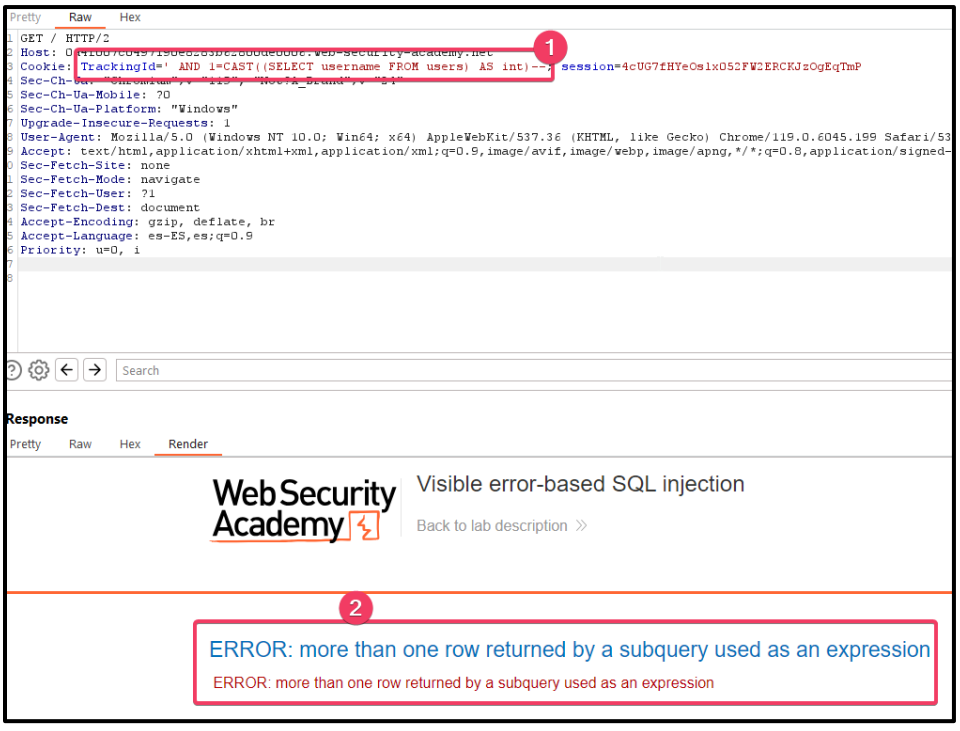
6. Adaptamos el SELECT para recibir nombres de usuario de la base de datos.



7. Nos da un error, vamos a borrar el valor original de la TrackingId cookie.

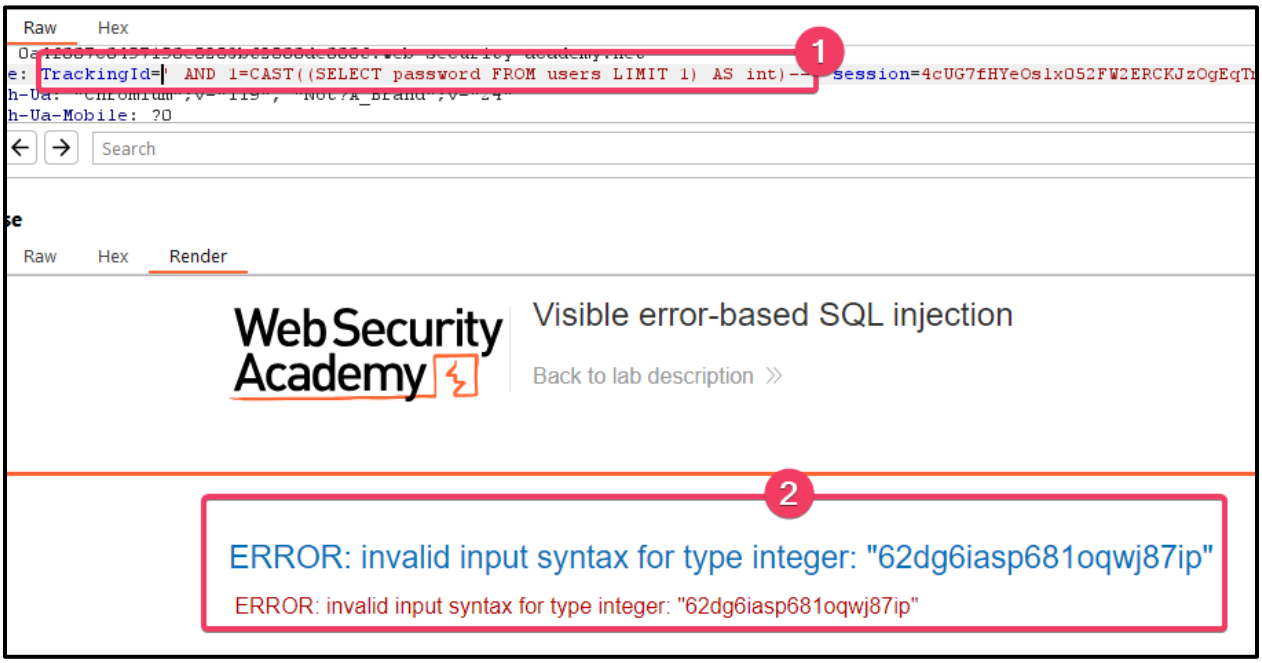


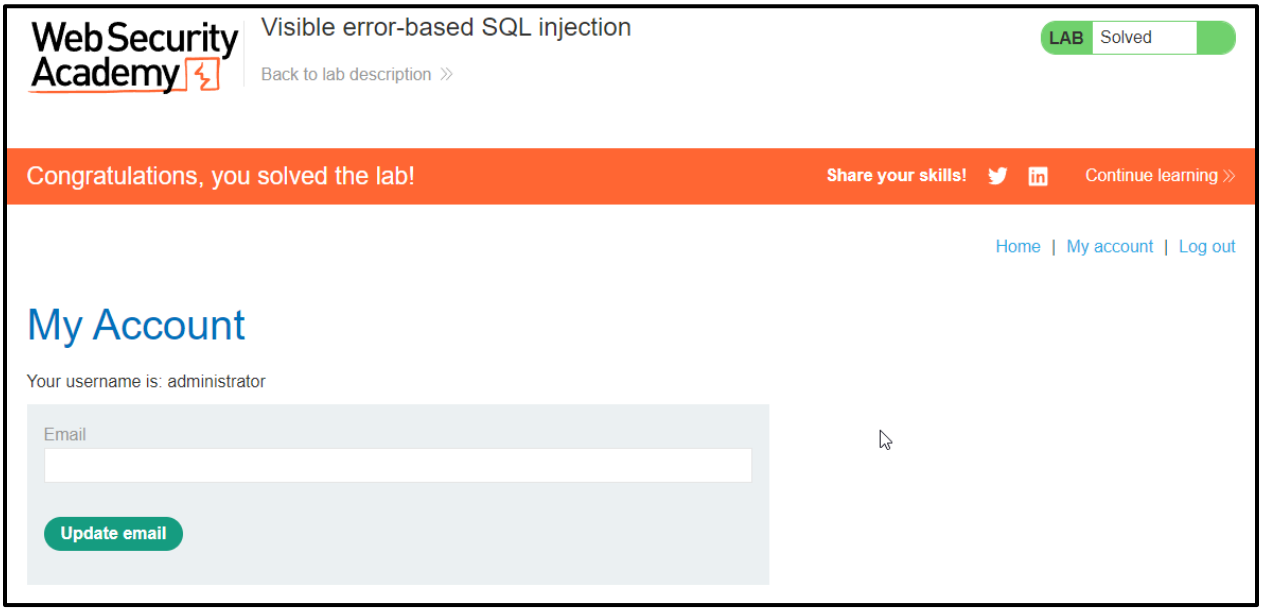
8. Nos sigue dando error, ya que nos devuelve más de una fila.



9. Vamos a modificar y solo enviar una fila. Y nos vamos a dar cuenta que nos devuelve el primer username de la tabla users, que es administrador.

10. Ahora que conocemos que administrador es el primer usuario en la tabla users vamos a sacar la contraseña.




Web Security Academy
Visible error-based SQL injection
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: administrator
Email
Update email

Exploiting blind SQL injection by triggering time delays (3 of 3)

Lab: Blind SQL injection with time delays and information retrieval

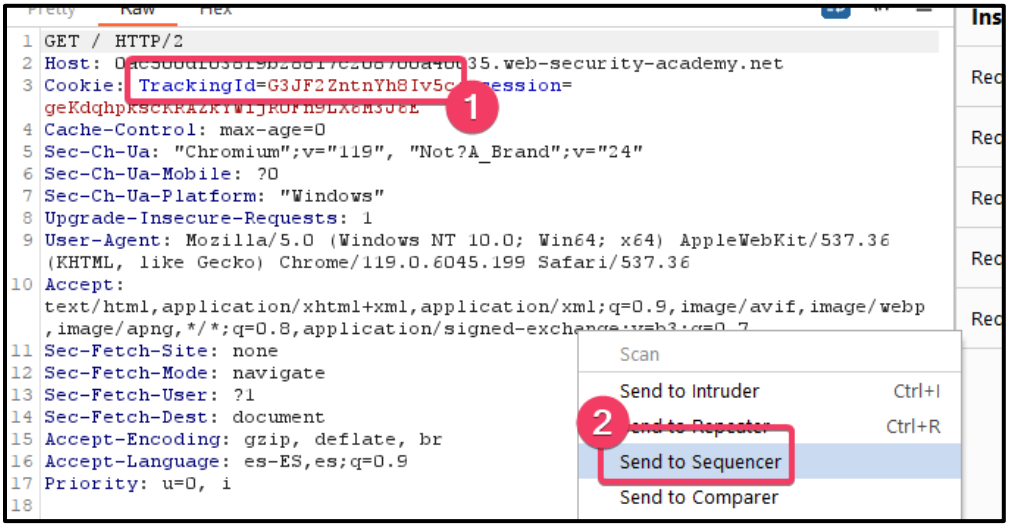
This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie.

The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information.

The database contains a different table called users, with columns called username and password. You need to exploit the blind SQL injection vulnerability to find out the password of the administrator user.

To solve the lab, log in as the administrator user.

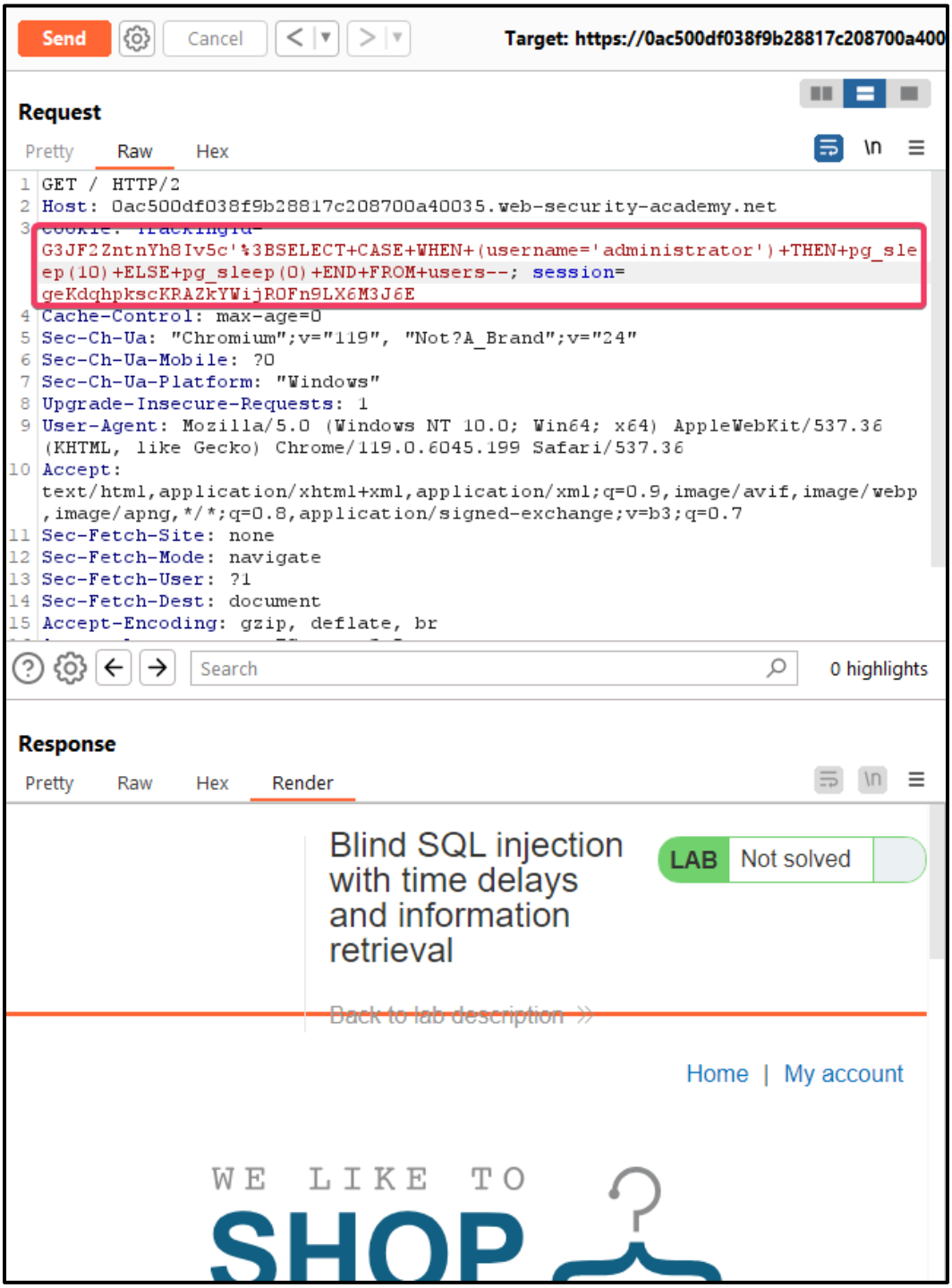
1. Interceptamos el request que tiene la cookie TrackingId de la Front page y la mandamos al Repeater.



2. Vamos a comprobar que la página tarde 10 segundos en cargar añadiendo lo siguiente (funciona correctamente).

3. Ahora vamos a poner lo mismo pero con una codición booleana falsa para ver si hace el delay de 10 segundos o no, que no debería hacerlo (no lo hace).

4. Comprobaremos si existe el usuario administrador, si la condición es verdadera.



5. Ahora que sabemos que existe el usuario administrador vamos a comprobar cuantos caracteres tiene la contraseña del usuario administrador. Con este paso comprobamos que la contraseña tiene más de 1 carácter.

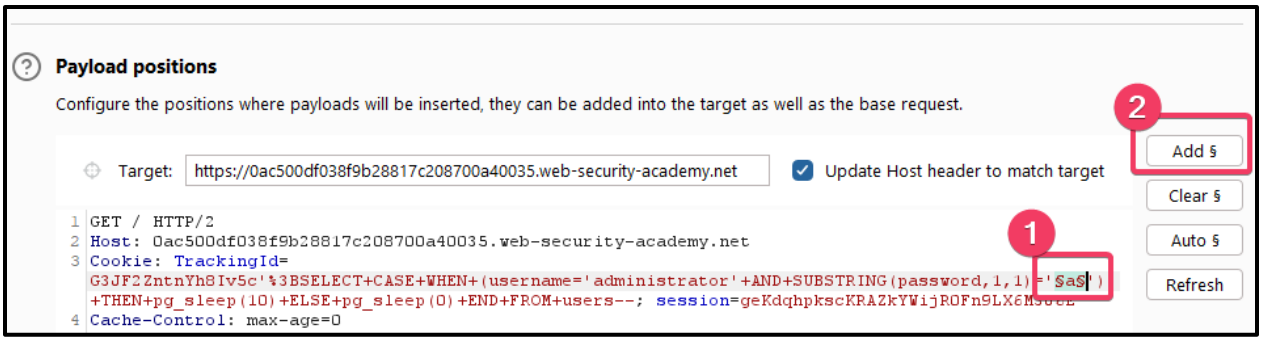
6. Ahora vamos a ir cambiando el número para saber cuántos caracteres tiene la contraseña.***'BSELECT+CASE+WHEN+(username='administrator'+AND+LENGTH(password)>1)+THEN+pg_sleep(10)+ELSE+pg_sleep(0)+END+FROM+users—***

Con el número 19 espera los 10 segundos de delay, y con el 20 ya no los espera, por lo que la contraseña tiene 20 caracteres.

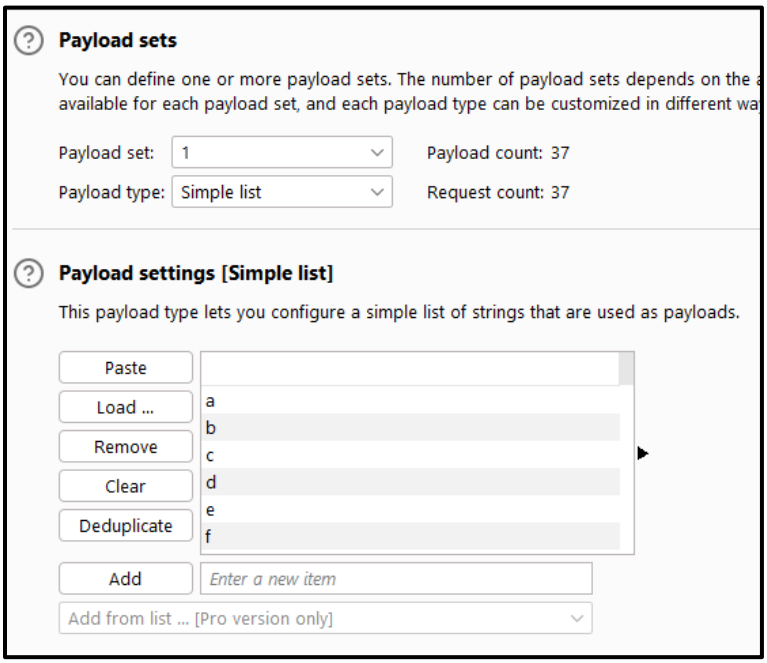
Gif comprobación 19 caracteres: https://i.imgur.com/QHTsxHa.gif

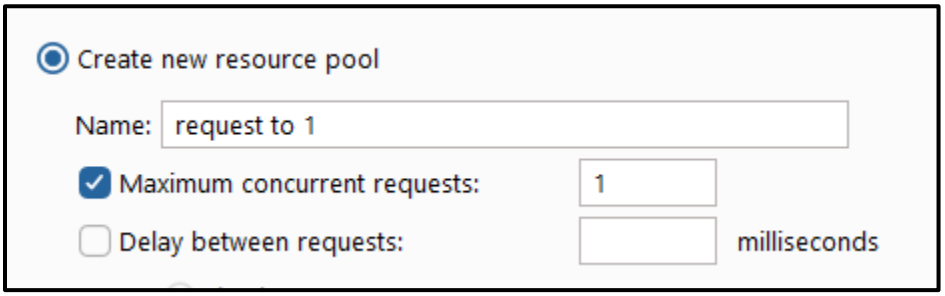
Gif comprobación 20 caracteres: https://i.imgur.com/H1QIzg3.gif

7. Para el siguiente paso vamos a pasar al Intruder para empezar a sacar la contraseña de administrador con la función SUBSTRING().

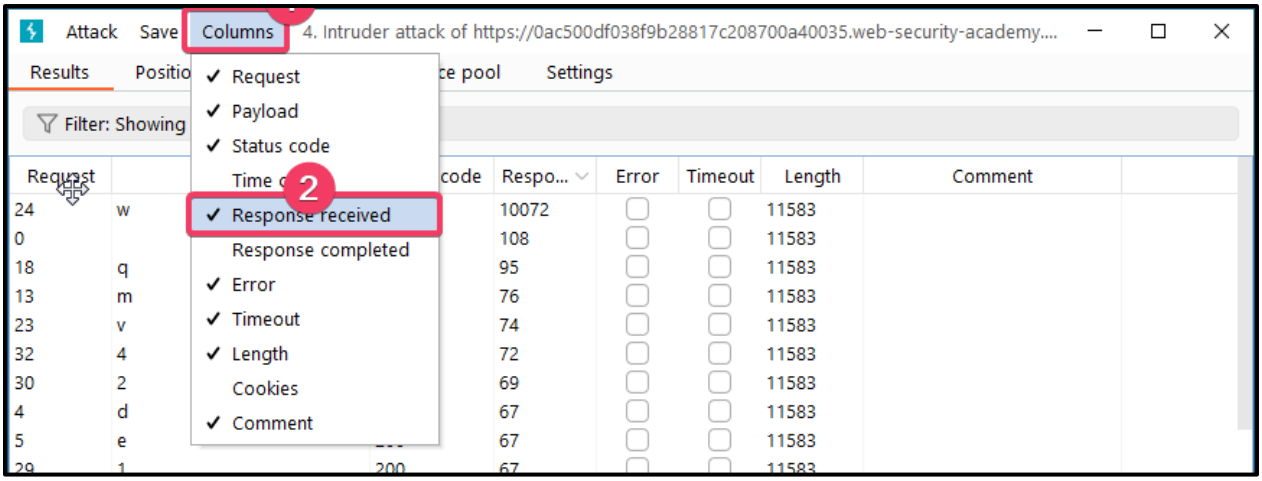


Añadimos de la a-z y del 0-9.

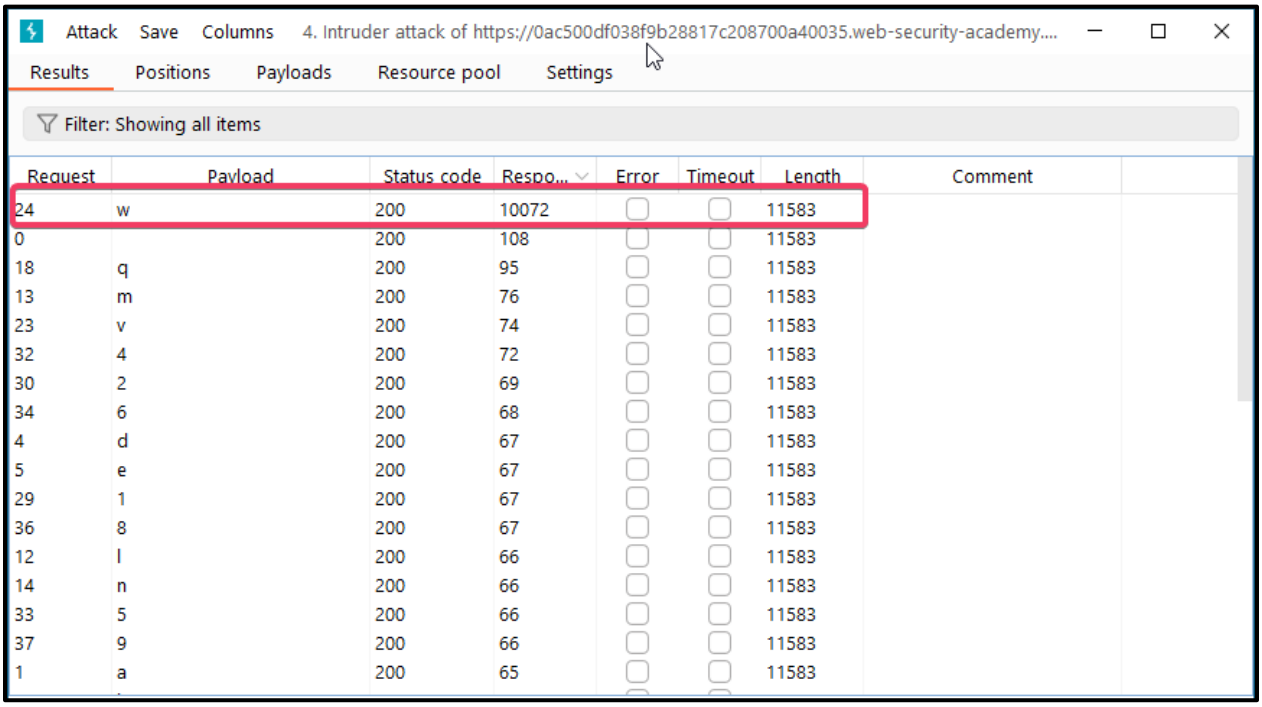




Añadimos la columna Request received para ver cual de los caracteres tarda más, para saber cual es el número correcto en la primera posición.

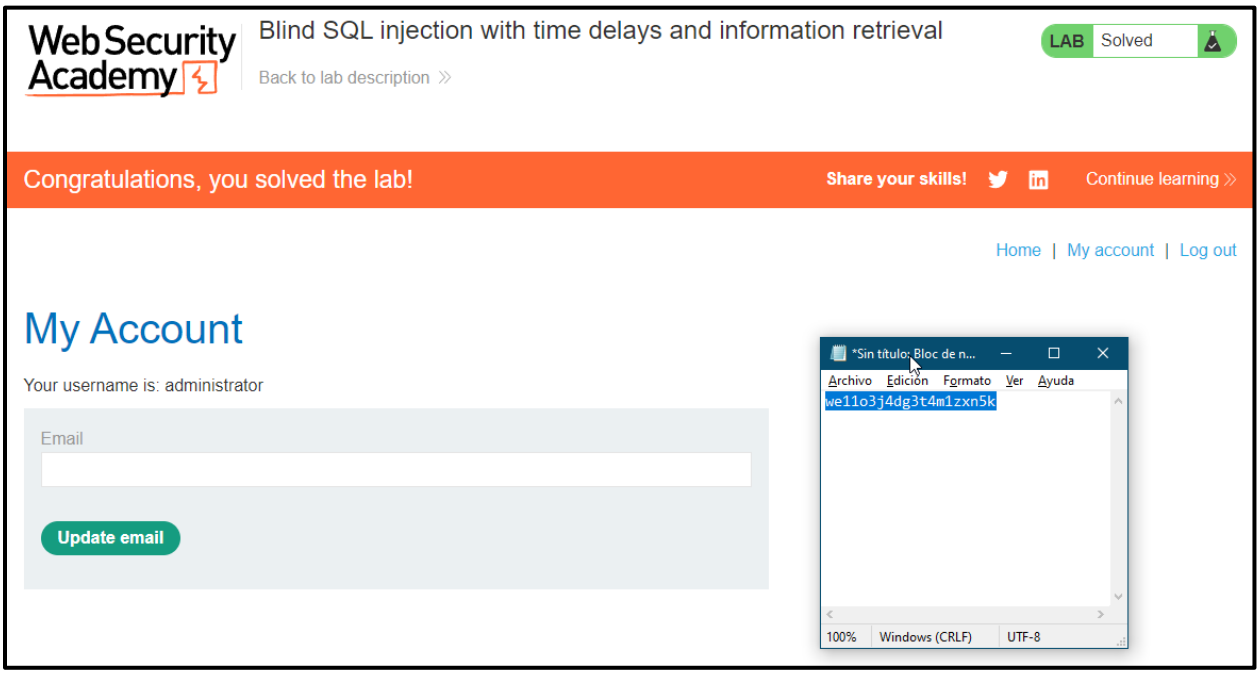


Primer carácter de la contraseña es w.



8. Posición 20 de la contraseña.

https://i.imgur.com/nZvCTri.gif



Exploiting blind SQL injection using out-of-band (OAST) techniques (5 of 5)

Lab: Blind SQL injection with out-of-band data exfiltration

This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie.

The SQL query is executed asynchronously and has no effect on the application's response. However, you can trigger out-of-band interactions with an external domain.

The database contains a different table called users, with columns called username and password. You need to exploit the blind SQL injection vulnerability to find out the password of the administrator user.

To solve the lab, log in as the administrator user.

Para hacer este ejercicio no hace falta el Burp Suite Professional, cosa que no tengo, asi que lo dejaré sin hacer.

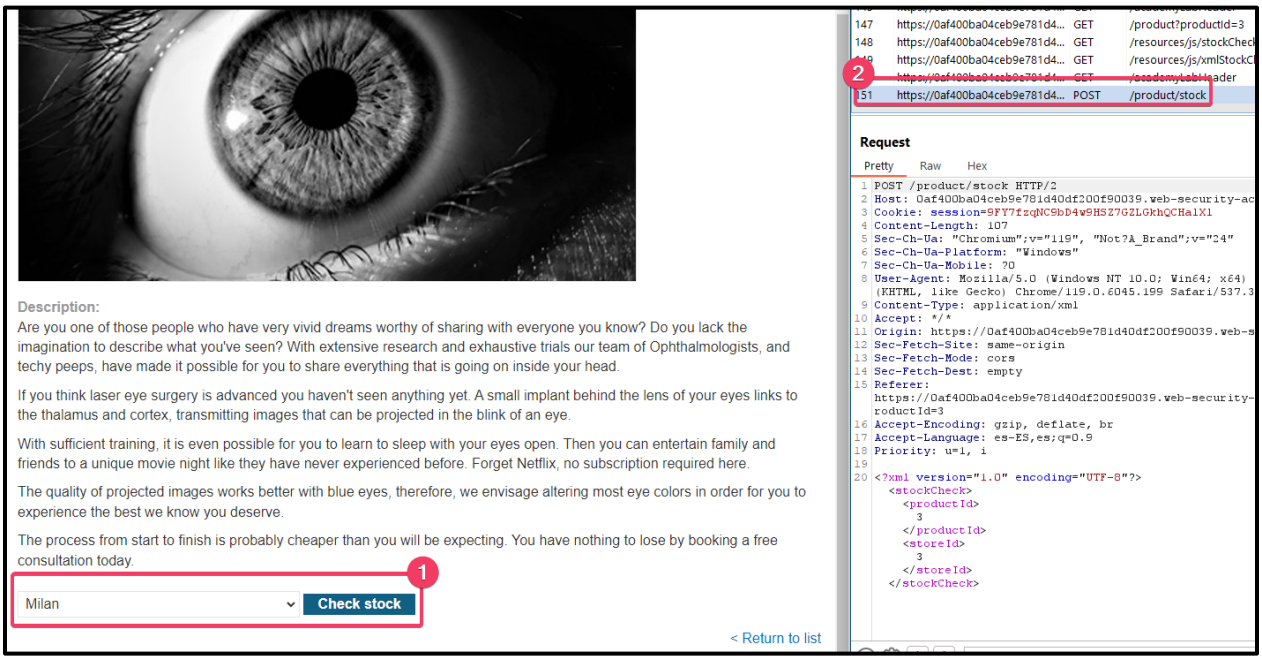
SQL injection in different contexts (2 of 2)

Lab: SQL injection with filter bypass via XML encoding

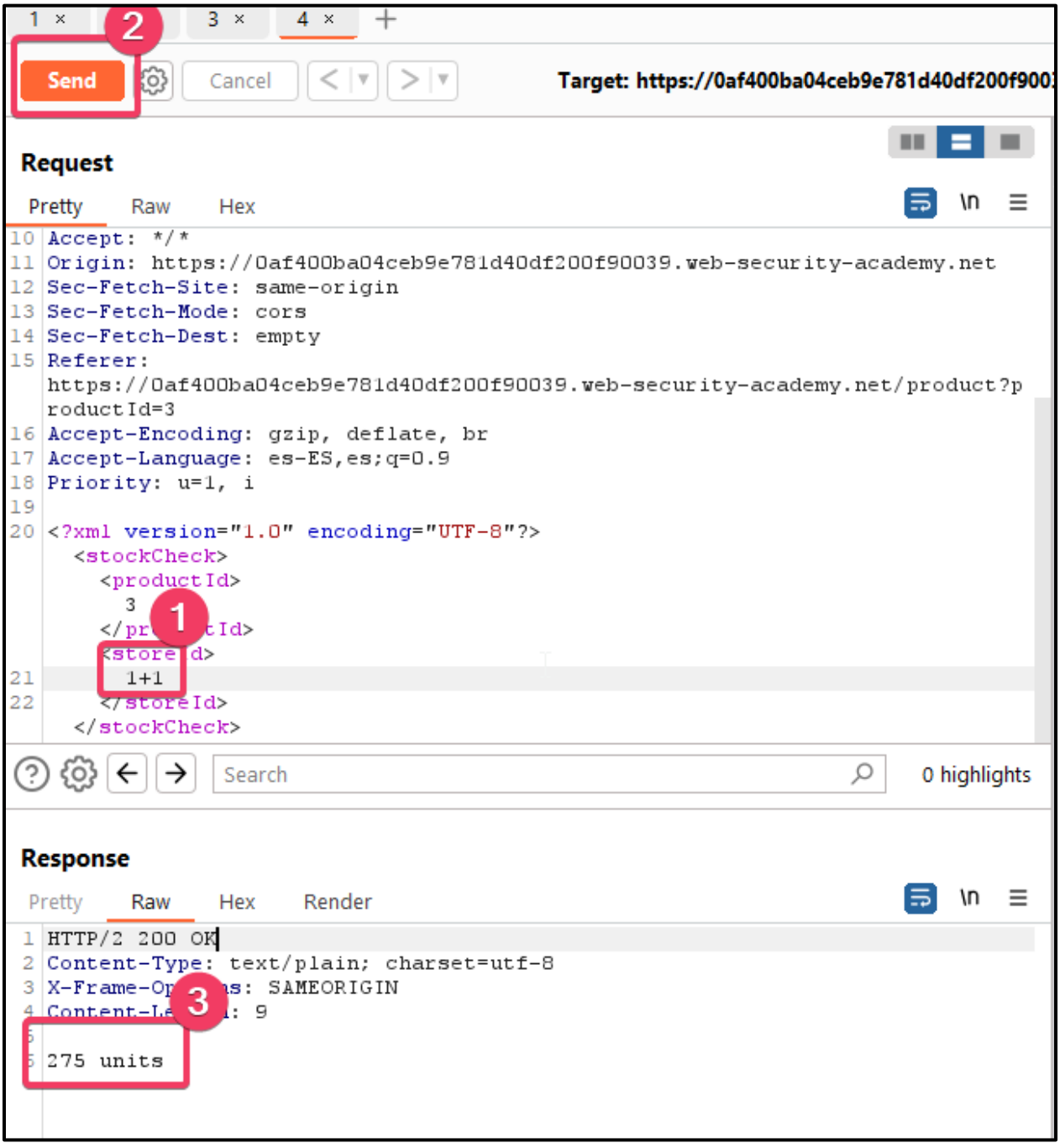
This lab contains a SQL injection vulnerability in its stock check feature. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables.

The database contains a user's table, which contains the usernames and passwords of registered users. To solve the lab, perform a SQL injection attack to retrieve the admin user's credentials, then log in to their account.

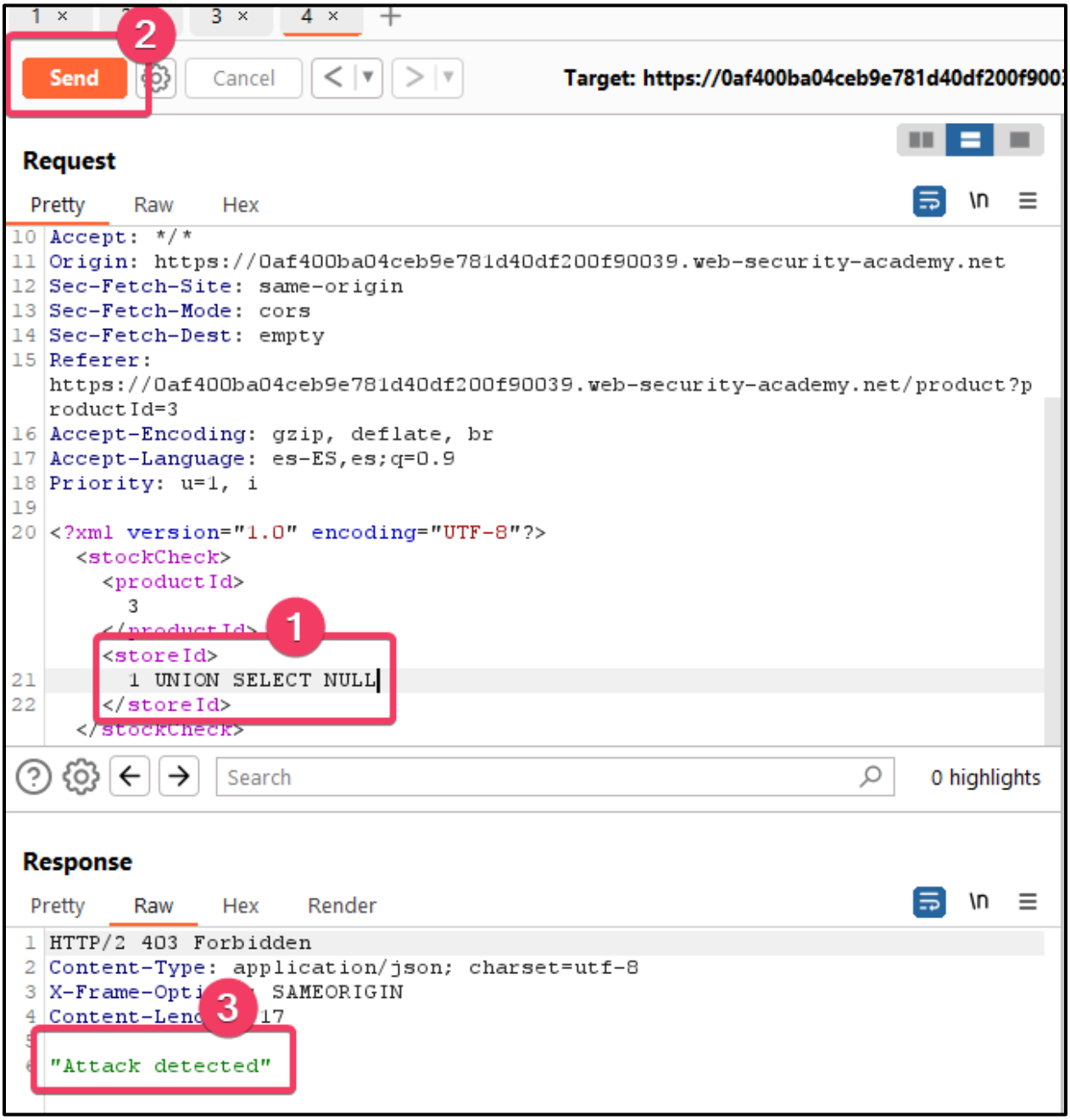
1. Vamos a mandar a burp repeater el post /product/stock.



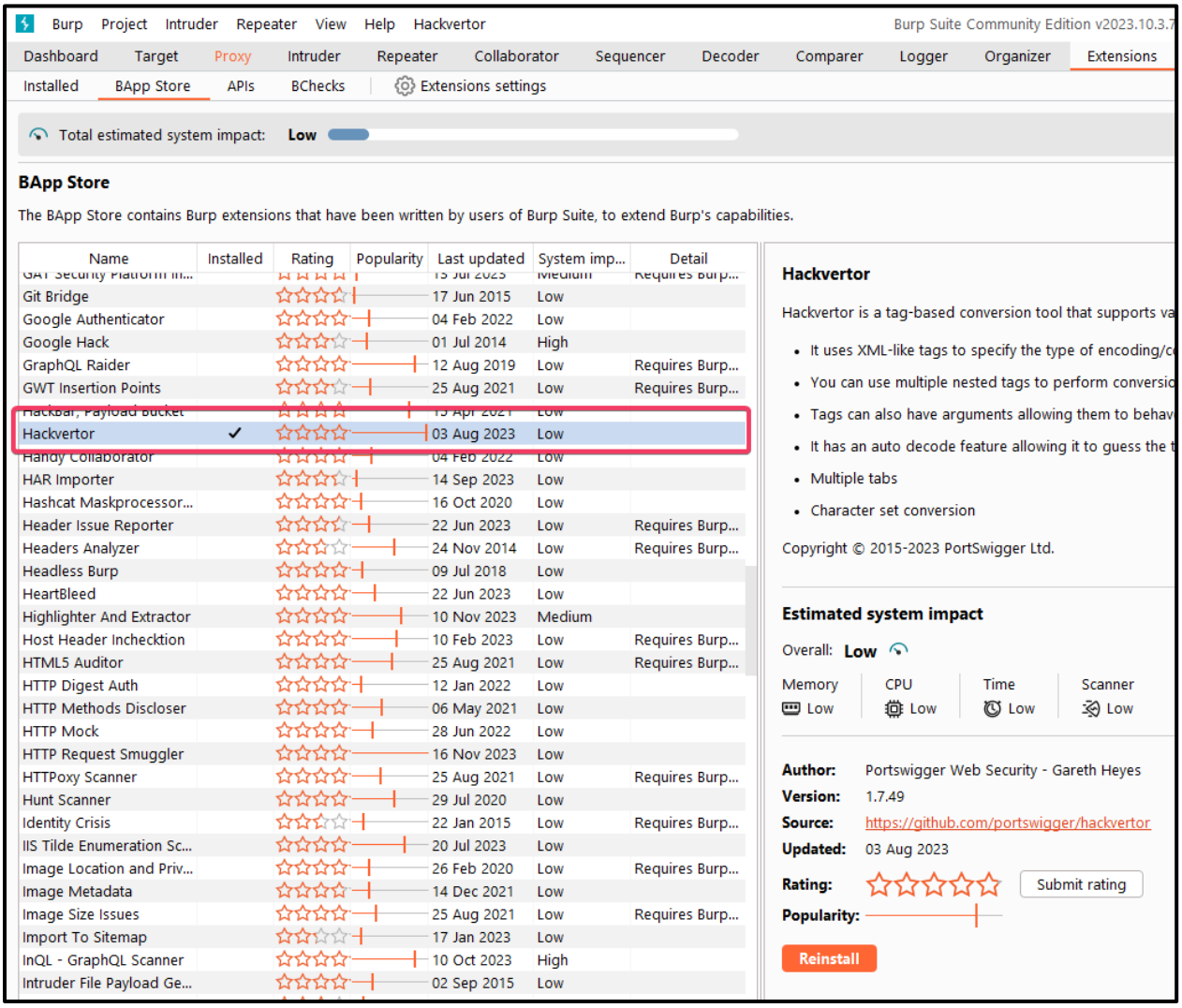
2. Vamos a comprobar si nuestro input esta siendo evaluado cambiando el storeId (si está evaluando).

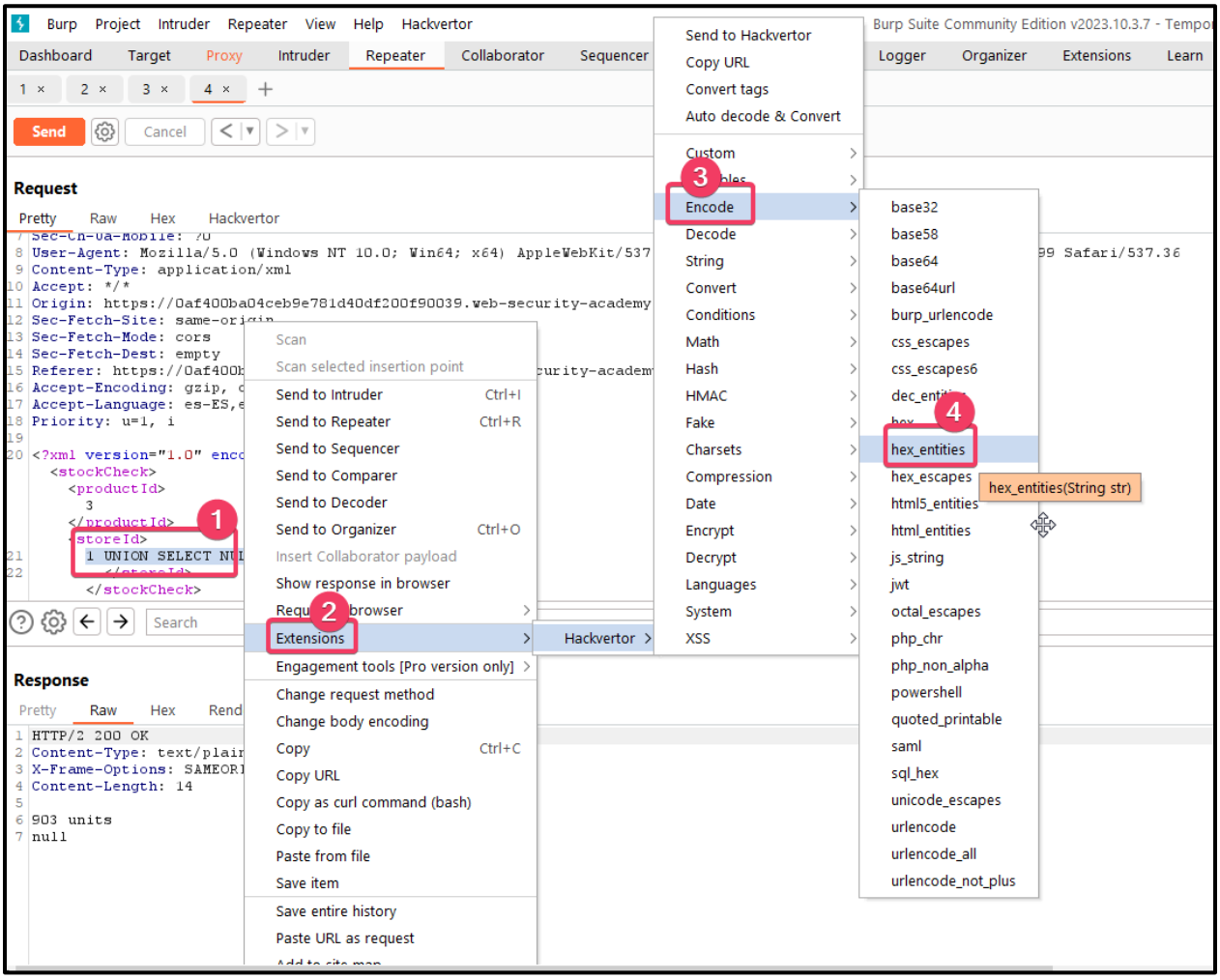


3. Vamos a intentar determinar el numero de columnas usando UNION SELECT.

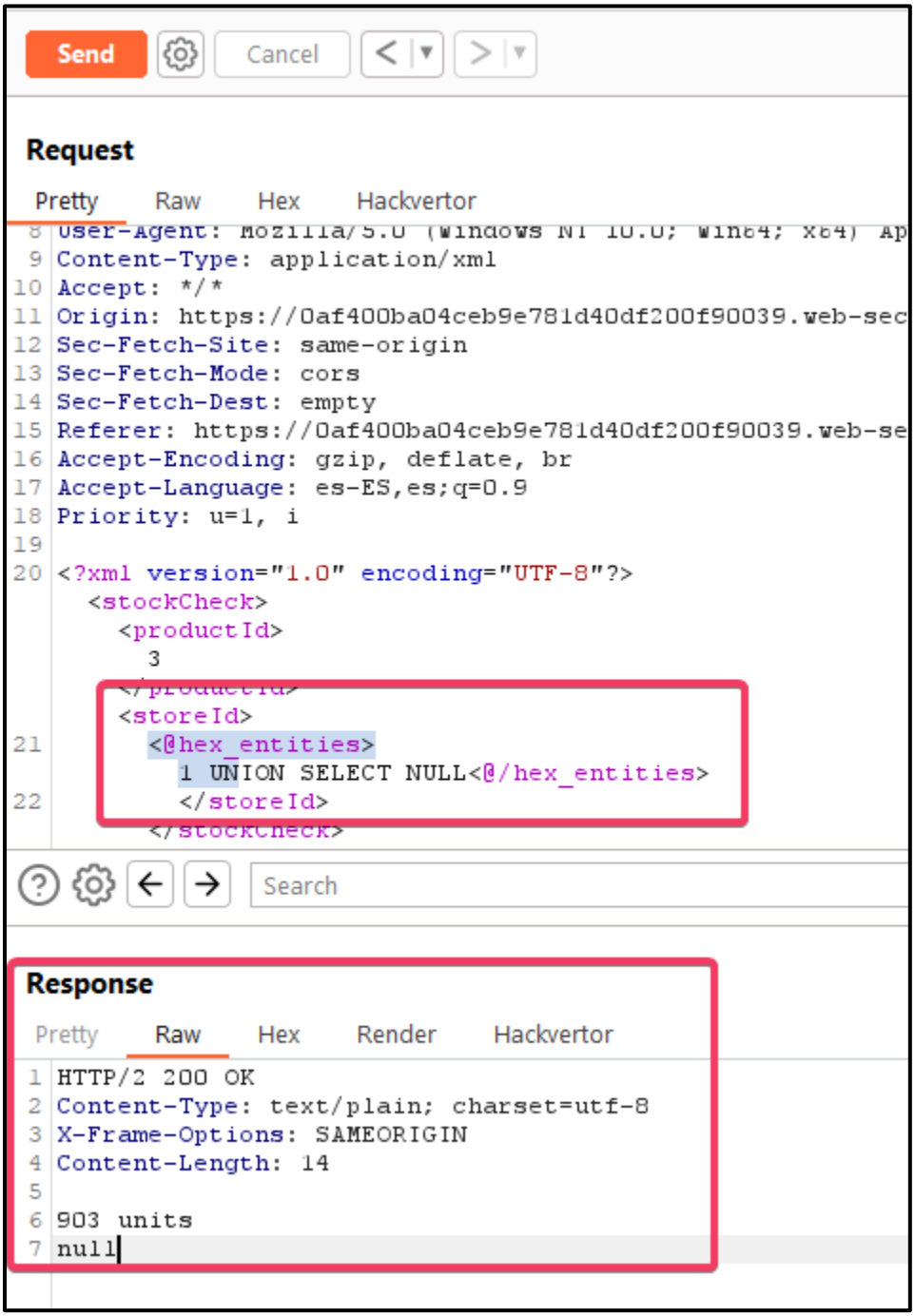


4. Seleccionamos todo el input, descargamos hackerton y lo usamos.

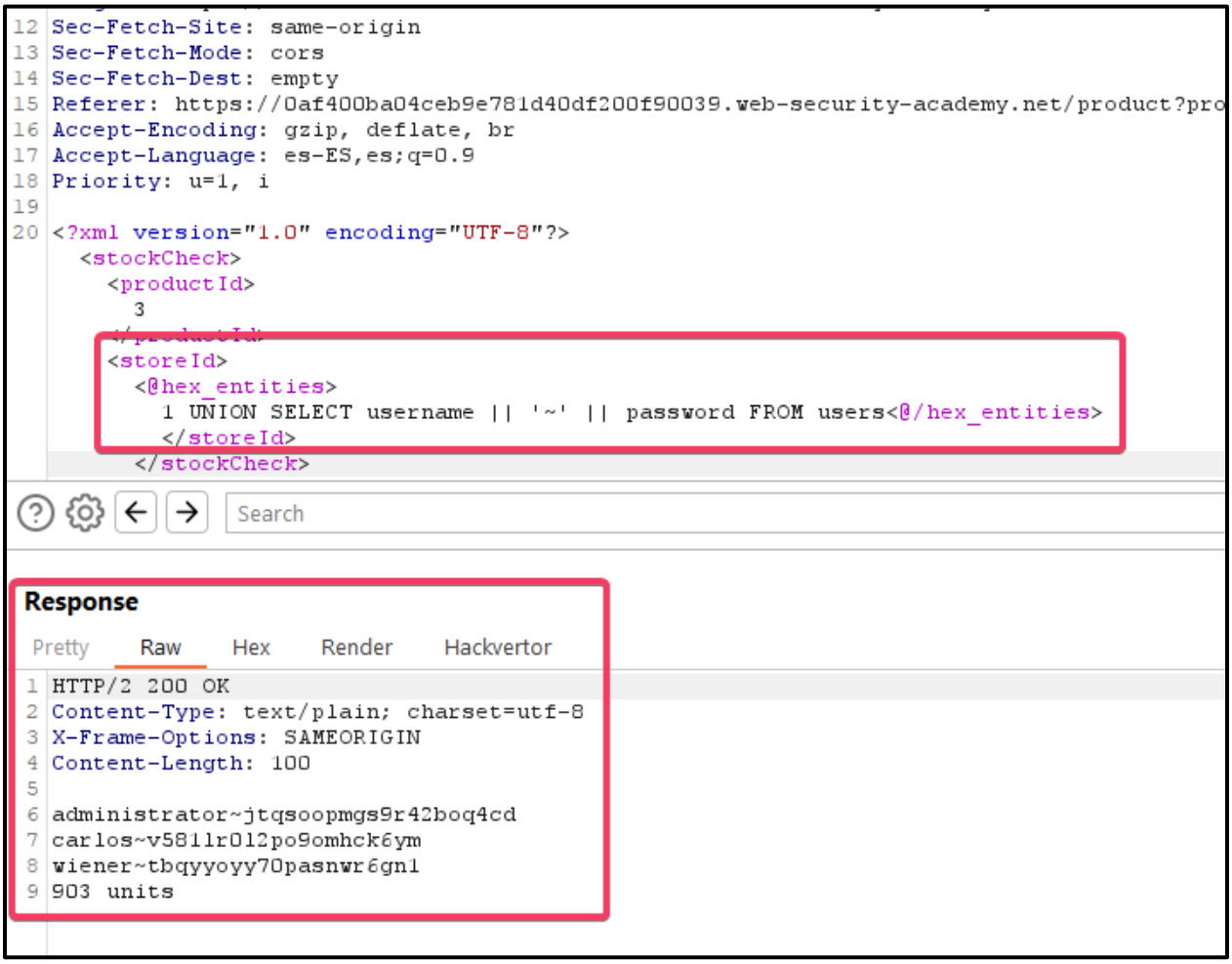


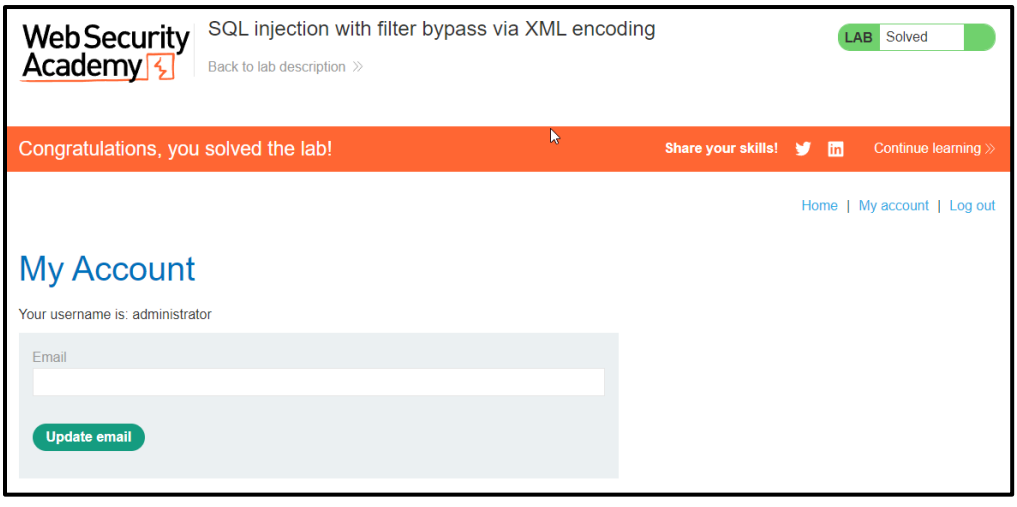

Burp Project Intruder Repeater View Help Hackvertor
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Logger Organizer Extensions Learn
Send Cancel
Request
Pretty Raw Hex Hackvertor
Response
Pretty Raw Hex
Scan
Scan selected insertion point
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Send to Organizer Ctrl+O
Insert Collaborator payload
Show response in browser
Extensions
Engagement tools [Pro version only]
Change request method
Change body encoding
Copy Ctrl+C
Copy URL
Copy as curl command (bash)
Copy to file
Paste from file
Save item
Save entire history
Paste URL as request
Send to Hackvertor
Copy URL
Convert tags
Auto decode & Convert
Custom
Encode
Decode
String
Convert
Conditions
Math
Hash
HMAC
Fake
Charsets
Compression
Date
Encrypt
Decrypt
Languages
System
XSS
base32
base58
base64
base64url
burp_urlencode
css_escapes
css_escapes6
hex_entities
hex_escapes
html5_entities
html_entities
js_string
jwt
octal_escapes
php_chr
php_non_alpha
powershell
quoted_printable
saml
sql_hex
unicode_escapes
urlencode
urlencode_all
urlencode_not_plus
hex_entities(String str)
1
2
3
4

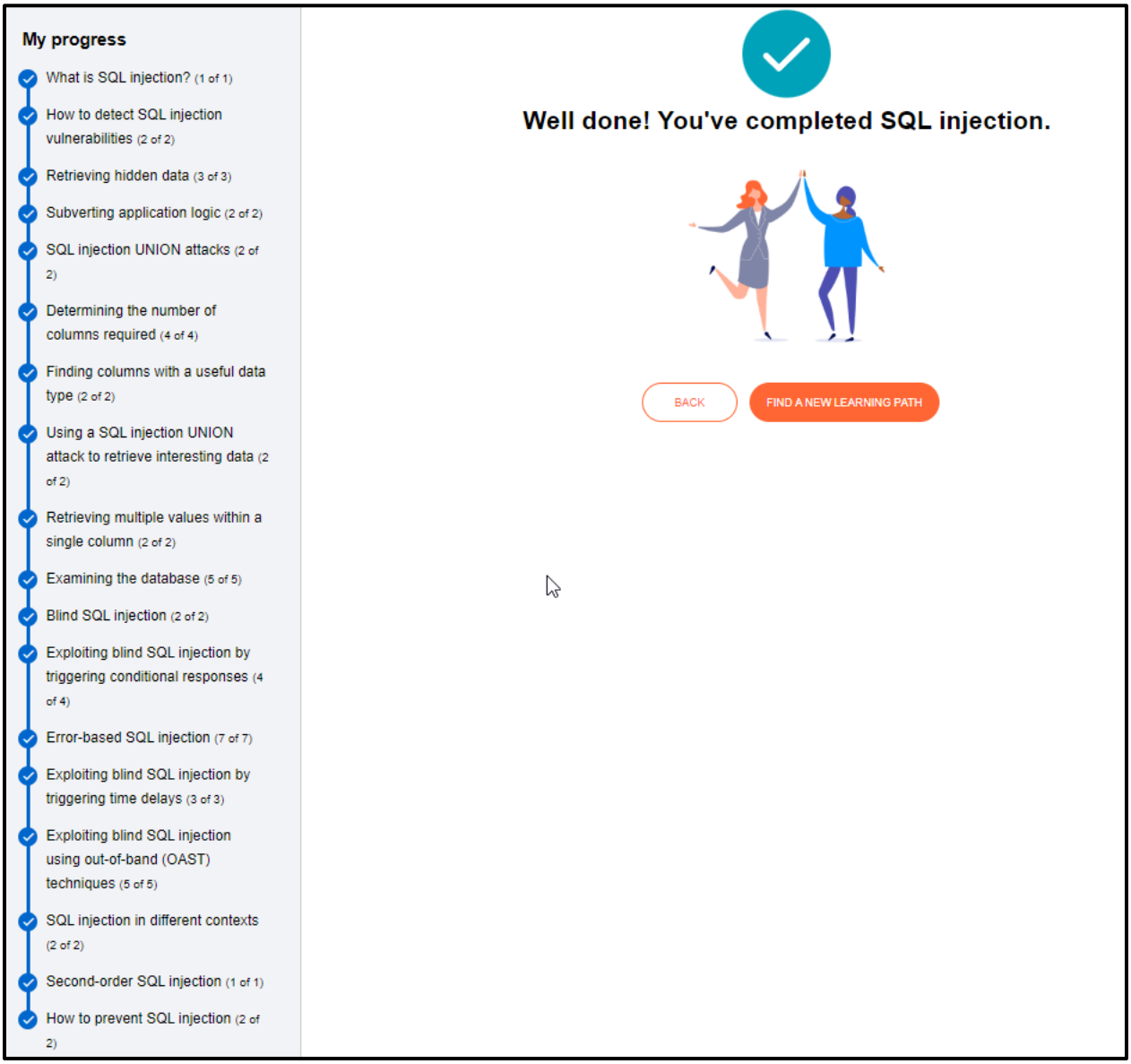
Podemos observar como hemos pasado el WAF.



5. Como solo nos puede dar 1 columna, necesitamos concatenar los usuarios y las contraseñas.






My progress
What is SQL injection? (1 of 1)
How to detect SQL injection vulnerabilities (2 of 2)
Retrieving hidden data (3 of 3)
Subverting application logic (2 of 2)
SQL injection UNION attacks (2 of 2)
Determining the number of columns required (4 of 4)
Finding columns with a useful data type (2 of 2)
Using a SQL injection UNION attack to retrieve interesting data (2 of 2)
Retrieving multiple values within a single column (2 of 2)
Examining the database (5 of 5)
Blind SQL injection (2 of 2)
Exploiting blind SQL injection by triggering conditional responses (4 of 4)
Error-based SQL injection (7 of 7)
Exploiting blind SQL injection by triggering time delays (3 of 3)
Exploiting blind SQL injection using out-of-band (OAST) techniques (5 of 5)
SQL injection in different contexts (2 of 2)
Second-order SQL injection (1 of 1)
How to prevent SQL injection (2 of 2)
Well done! You've completed SQL injection.
BACK
FIND A NEW LEARNING PATH