


Challenge 1 – Drobots

Very easy – Web Category

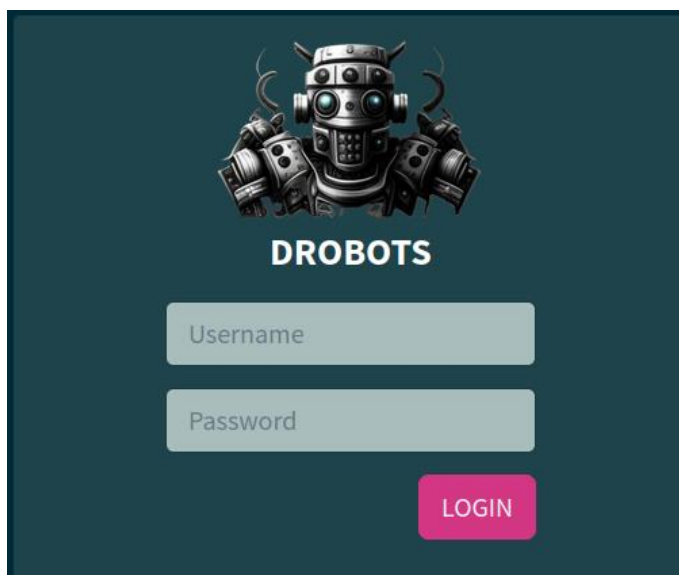
CHALLENGE NAME

Drobots



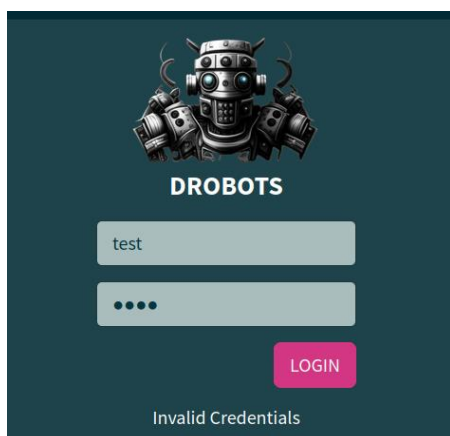
Pandora's latest mission as part of her reconnaissance training is to infiltrate the Drobots firm that was suspected of engaging in illegal activities. Can you help pandora with this task?

In this challenge we are presented with an IP address and a port number. This challenge being a web challenge, it is hosting a website with a login page.



The image shows the login page of the 'Drobots' challenge. At the top is a robot head icon. Below it is the word 'DROBOTS' in white capital letters. There are two input fields: 'Username' and 'Password'. Below the password field is a pink 'LOGIN' button.

The first and easiest thing to check for is an SQLi. In order to do this, we first need to check what the normal error message of incorrect credentials is. So, I used test: test to try it out.



The image shows the login page after an attempt with the credentials 'test:test'. The 'Username' field contains 'test' and the 'Password' field is masked with dots. The 'LOGIN' button is still visible. Below the button, the text 'Invalid Credentials' is displayed in white.

Now that we know what the normal output is we can try and close a bracket that might be there and create the rest of the SQL query. The injection we use is ...

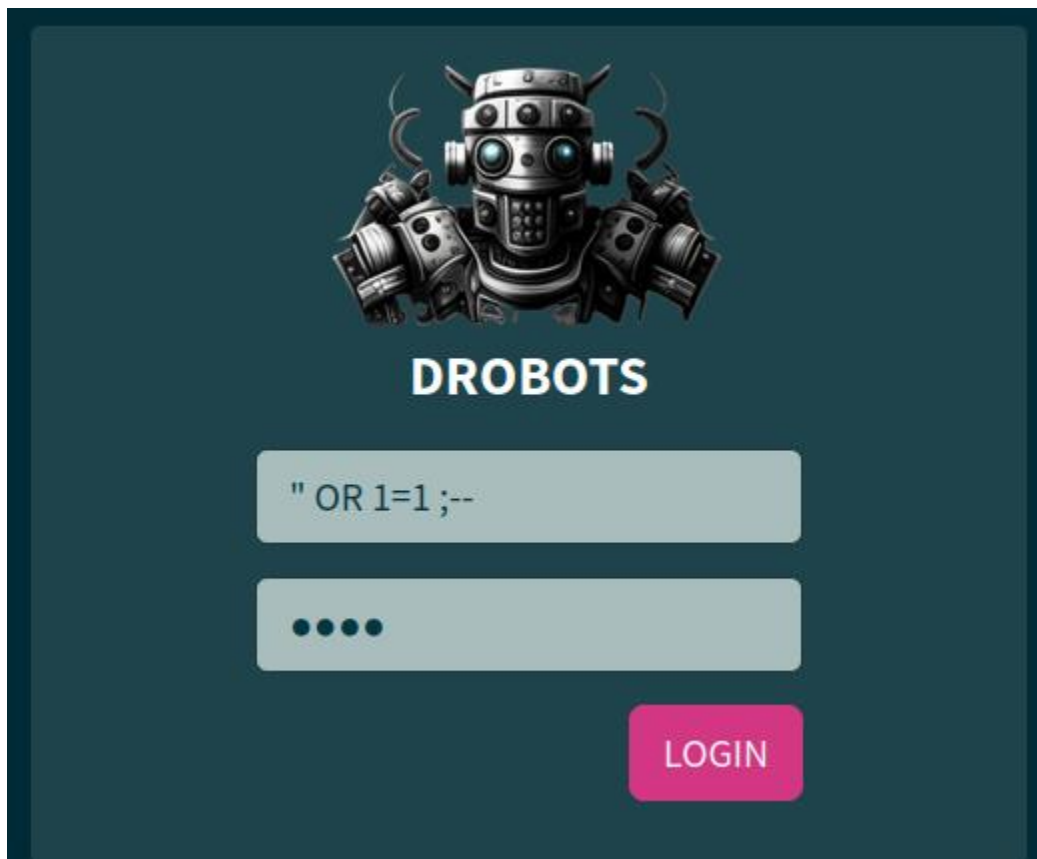
" OR 1=1 ;--

The " is to close the bracket making it equal false

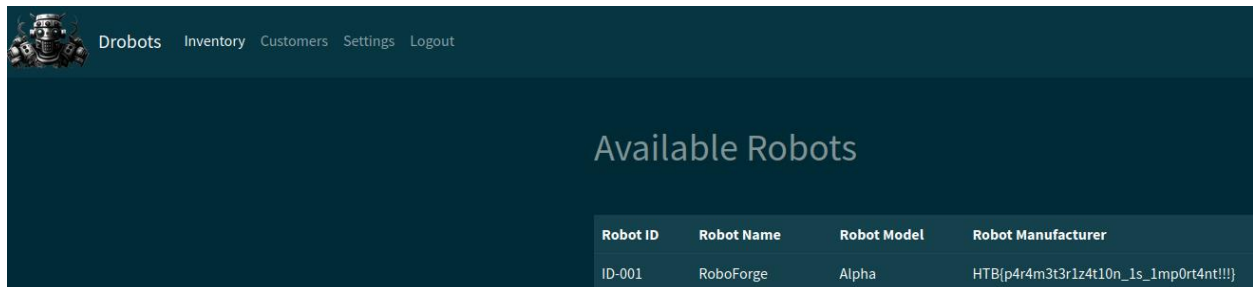
The OR operator is to say, "false or.."

The 1=1 is always true making the statement, so fat is select anything that is false or true.

The ;-- is to finish the statement and comment out anything that may come after the payload.



Now we can press login and see if it works...



Robot ID	Robot Name	Robot Model	Robot Manufacturer
ID-001	RoboForge	Alpha	HTB{p4r4m3t3r1z4t10n_1s_1mp0rt4nt!!!}

And it does!

The reason why this works is because the code behind handling the login query is not sanitized. Because we cannot see the exact code we cannot tell where exactly this can be resolved, but one easy fix for this would be to not allow any special characters to pass through the username field.

Essentially what this payload is doing is turning the database query to “fetch username and password, then login if they both match” into “just pull all user information”.

What was something like...

```
SELECT * FROM users WHERE username = "<username>" AND password = "<Password>"
```

Then becomes...

```
SELECT * FROM users WHERE username = ""(False) OR 1=1(True) ;-- AND Password = "<password>"
```

Where the section in green is commented out and not parsed by the database.