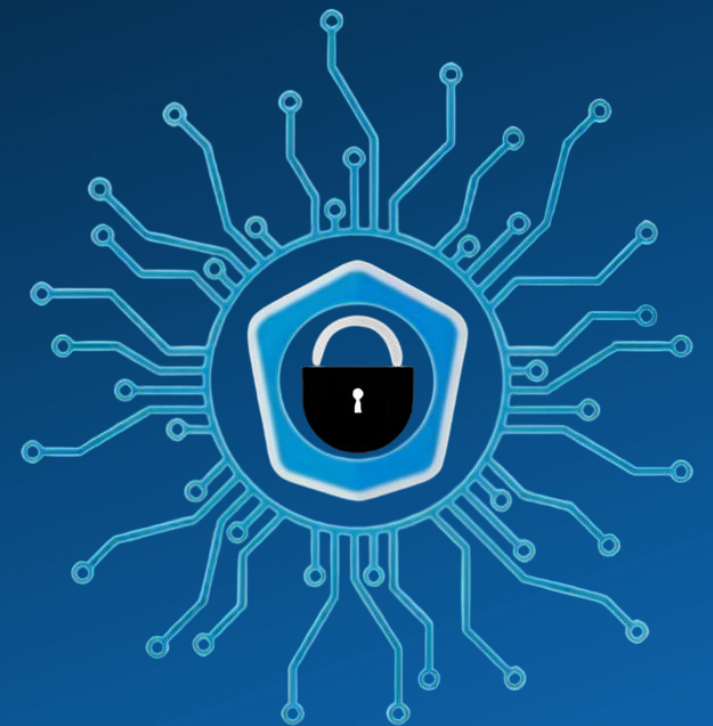


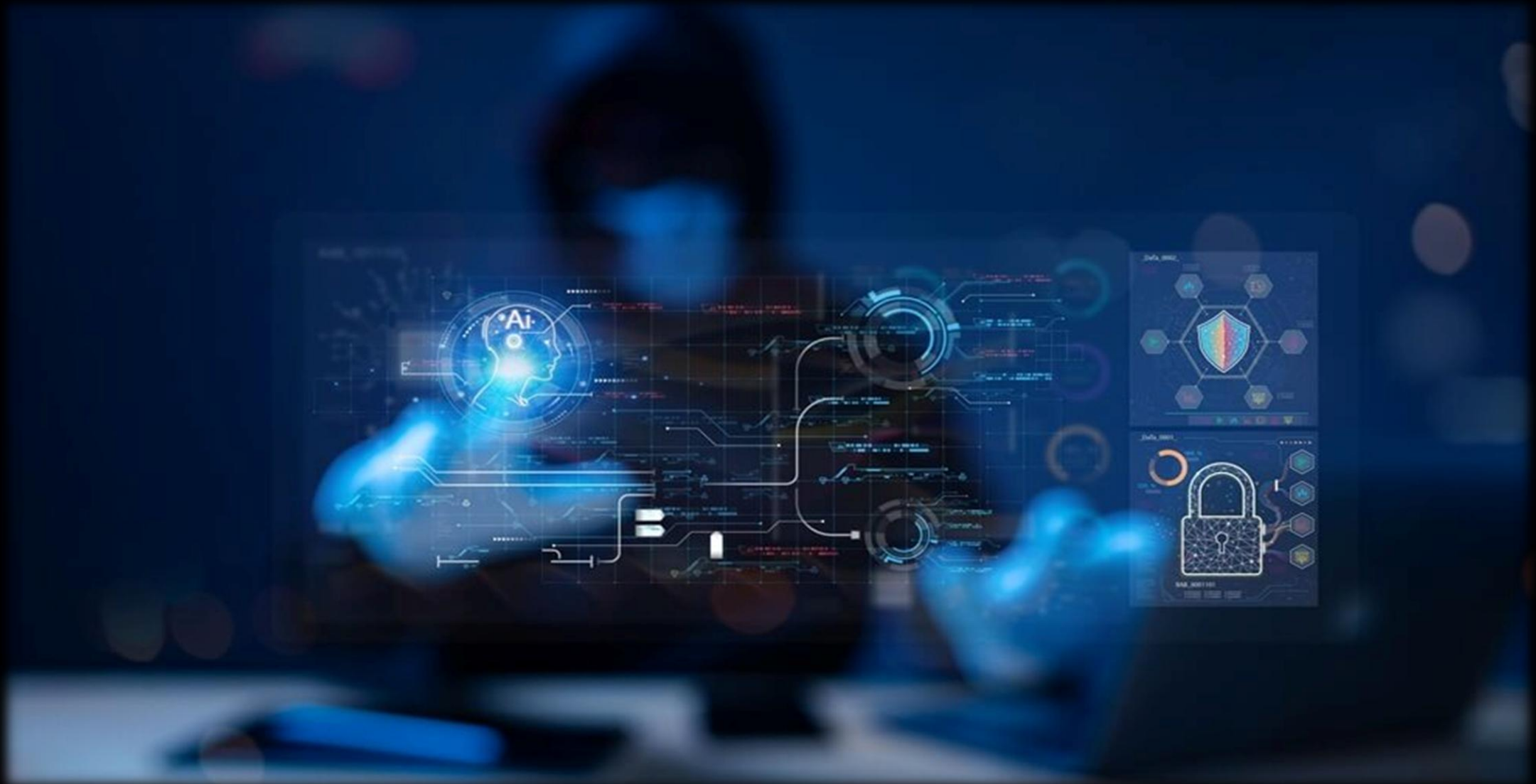
ALWAYSTRUE CHALLENGE WALKTHROUGH

**WRITTEN BY:
TALEEN SKAFI**

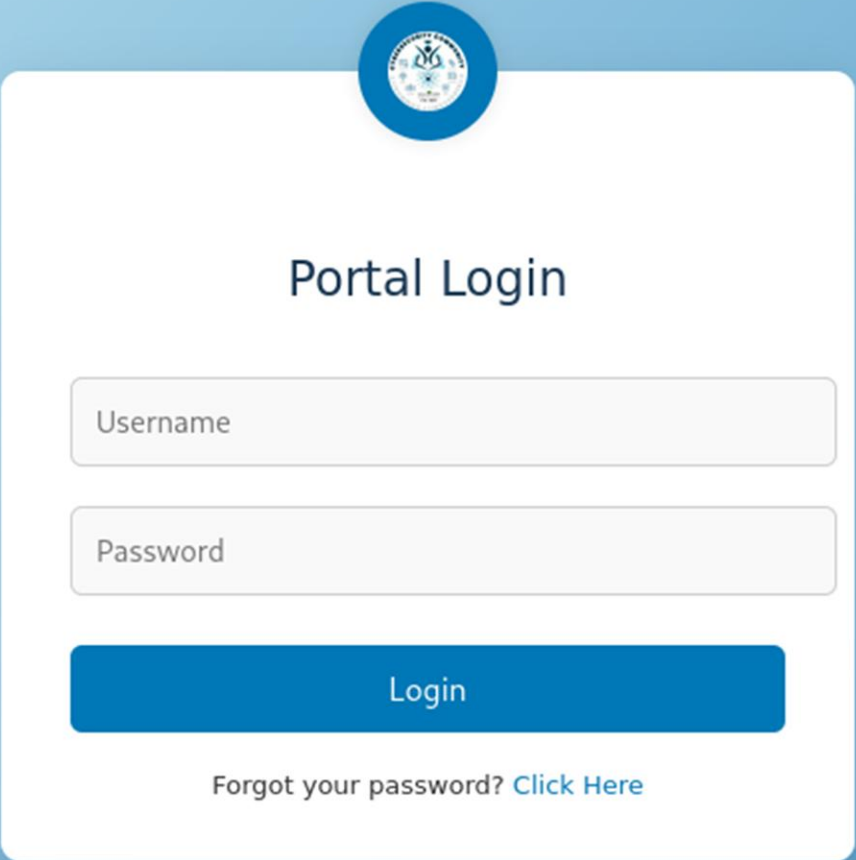


BZU - CSC

Welcome to **AlwaysTrue**. In this challenge, we're looking at a login page that seems pretty standard at first, just a username and password field, nothing special. But we don't have any credentials, so we need to get creative. With a closer look, it turns out the application isn't handling inputs very safely. By using a classic **SQL injection** trick, we might be able to fool the system into thinking we're already logged in. No brute force, no cracking, just a simple line of input that always returns true. Let's see if we can break in.



When we open the challenge, we see a normal login portal:

A screenshot of a web portal login page. The page has a light blue background. In the center, there is a white rectangular box with rounded corners. At the top of this box is a circular logo featuring a shield and a key. Below the logo, the text "Portal Login" is centered. Underneath, there are two input fields: "Username" and "Password". Below these fields is a blue button with the text "Login". At the bottom of the box, there is a link that says "Forgot your password? Click Here".

Portal Login

Username

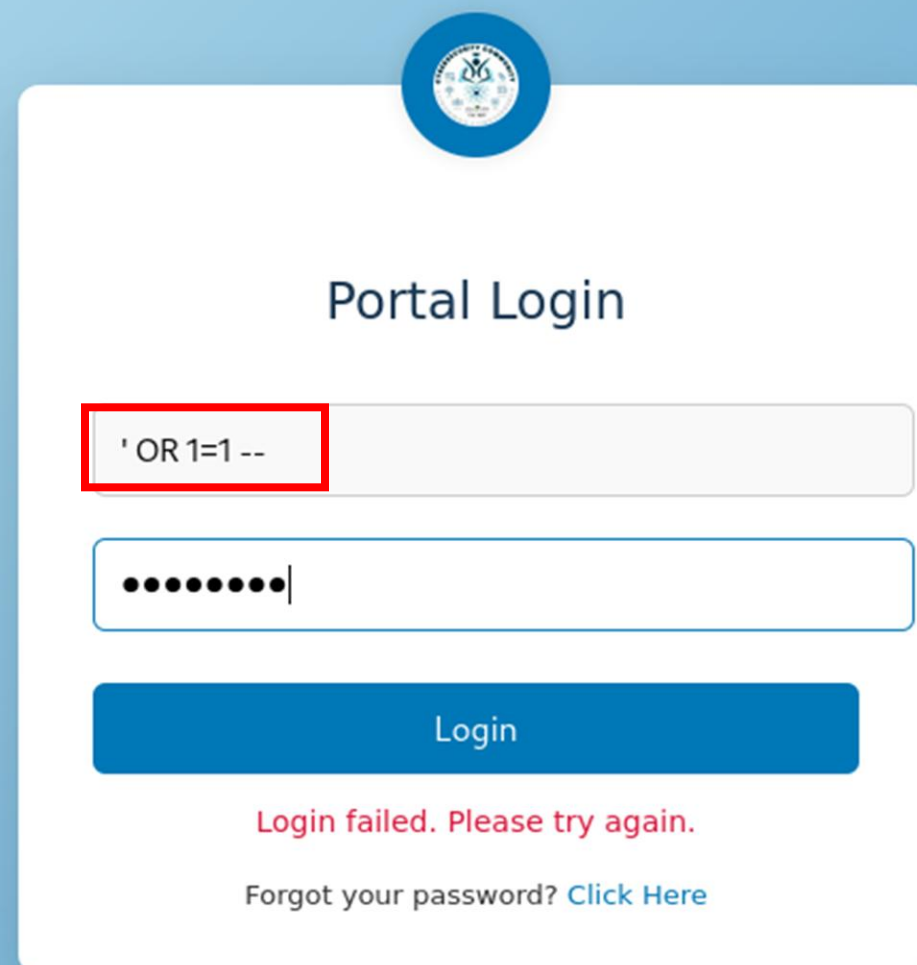
Password

Login

Forgot your password? [Click Here](#)

Sometimes, when a login page isn't properly secured, it takes the user's input and put it directly into the SQL query that interacts with the database. This can lead to a vulnerability known as **SQL injection**, where some payloads are used as an input to change the behavior of the query and let us log in without valid credentials. Let's try using it to bypass this login portal.

Let's start by trying the classic SQL injection payload: '**OR 1=1 --**' as the username, and anything for the password. This input is designed to modify the login query so that it always returns true, potentially letting us bypass the authentication check.

A screenshot of a web application's login page. At the top center is a circular logo. Below it, the text 'Portal Login' is centered. There are two input fields: the first is for the username and contains the text 'OR 1=1 --' (highlighted with a red box), and the second is for the password and contains ten dots. Below the fields is a blue 'Login' button. Underneath the button, a red error message reads 'Login failed. Please try again.' At the bottom, there is a link that says 'Forgot your password? Click Here'.

On the server side, the application takes whatever we type into the login form and plugs it straight into a SQL query. When we enter '**OR 1=1 --**' as the username and anything for the password, the query it builds looks like this:

```
SELECT * FROM users WHERE username = ' OR 1=1 --' AND password = 'anything';
```

Here's what's happening: the **--** is used to comment out the rest of the line in SQL, so everything after it (including the password check) is ignored. That leaves us with a condition like:

username = " OR 1=1

and since **1=1** is always true, the query returns all users in the database. The app might then log us in as the first user it finds, which could be the admin , without needing a real username or password.

If we want to make sure we log in specifically as the **admin** user, we can try entering something like: **admin' --**

This works by closing off the username input with the **'**, and then using **--** to comment out the rest of the SQL query. On the server side, the app might build a query like this:

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

Because **--** tells the database to ignore everything after it, the password check is skipped entirely. So the query just looks for a user with the username **admin**, and if that user exists, the app might let us in without applying any password.



Portal Login

Login

Forgot your password? [Click Here](#)

And just like that, both payloads did the trick. Whether we used ' **OR 1=1 --** to bypass the login or **admin' --** to go straight for the admin account, we were in. The app logged us in as admin, and sure enough , there was the flag waiting for us 😊

Welcome back!

Your dashboard is loading...

Flag: CSC{Sql_1nj3ct10n_1s_3aSy}