# XSS Injection Vulnerability in the Online-Food-Ordering-System

## Simple Demonstration

The vulnerability exists in the `register.php` file, where the following code snippet is present:

```php

$name = $\_POST['name'];

$mobile = $\_POST['mobile'];

$address = $\_POST['address'];

$email = $\_POST['email'];

$password = $\_POST['password'];

$cpassword = $\_POST['cpassword'];

if ($password != $cpassword) {

?>

<script type="text/javascript">

alert("Password and Confirm Password not match!");

</script>

<?php

die();

}

$esql = "SELECT 1 FROM `user` WHERE `email` = '$email'";

$erun = mysqli\_query($conn, $esql);

if (mysqli\_num\_rows($erun) > 0) {

?>

<script type="text/javascript">

alert("Email Already Exist!");

</script>

<?php

die();

}

$sql = "INSERT INTO `user` (`name`, `mobile`, `address`, `email`, `password`, `cpassword`) VALUES ('$name'";

```

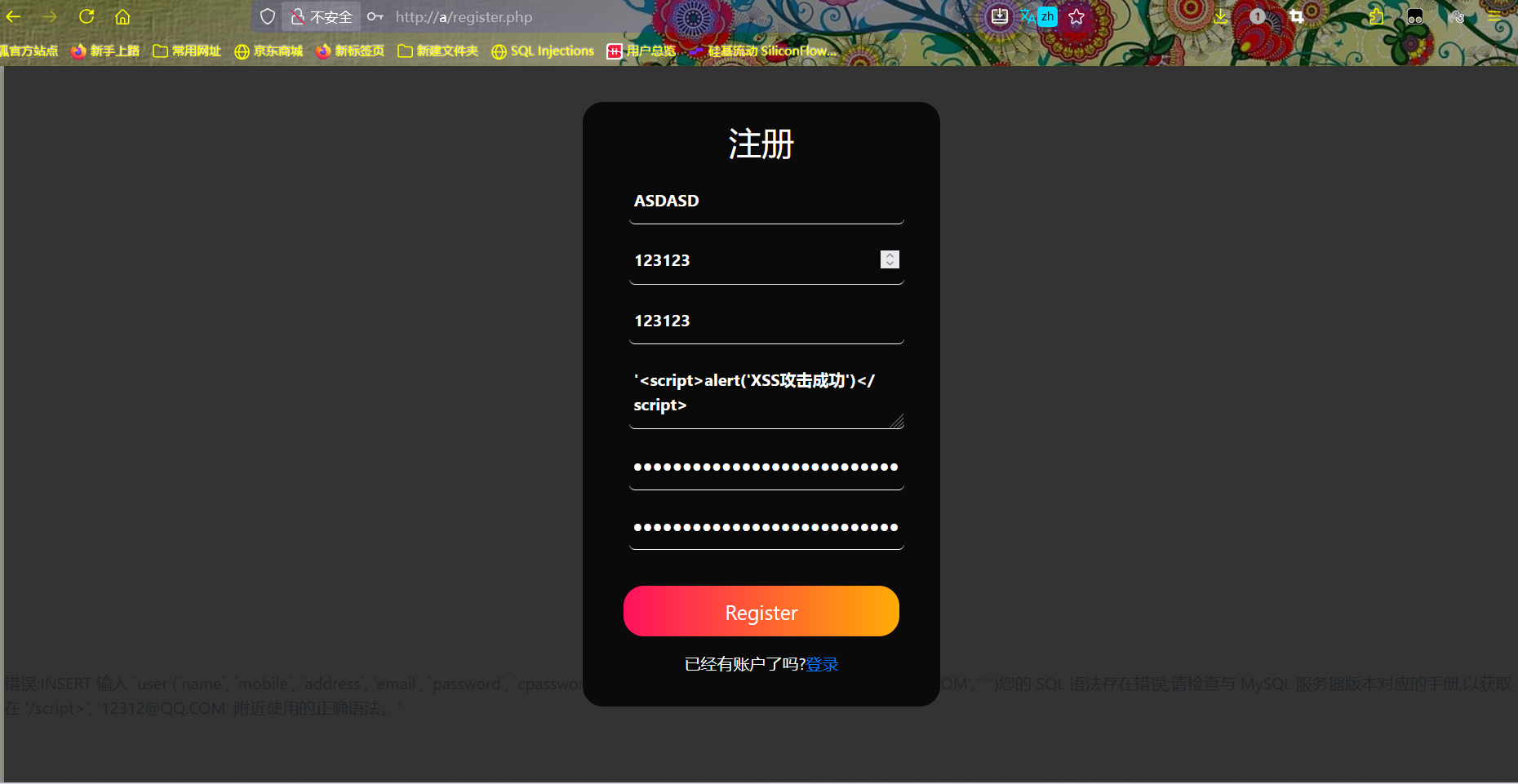
In this code, fields such as \*\*name\*\* (full name) and \*\*address\*\* directly receive user input via `$\_POST` (e.g., `$name = $\_POST['name']`). No XSS-related filtering is implemented—for example, there is no removal of `<script>` tags or escaping of special characters. These unfiltered inputs are directly concatenated into SQL statements and stored in the `user` table.

If the user inputs the following payload:

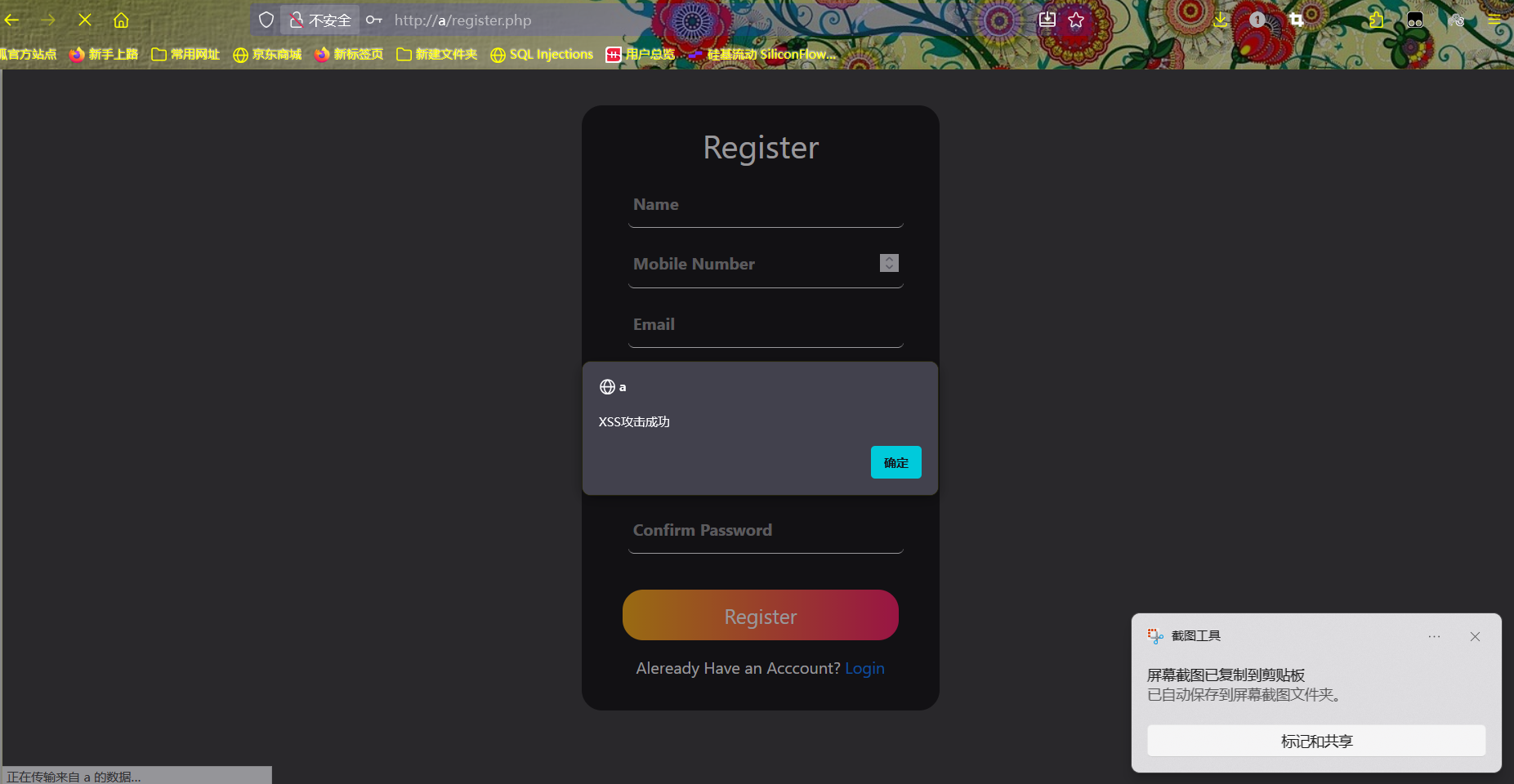
`<script>alert('XSS Attack Succeeded')</script>`

The payload will directly close the preceding HTML/SQL structure and execute the embedded JavaScript code.

### Example of Input



### Successful Alert Pop-up



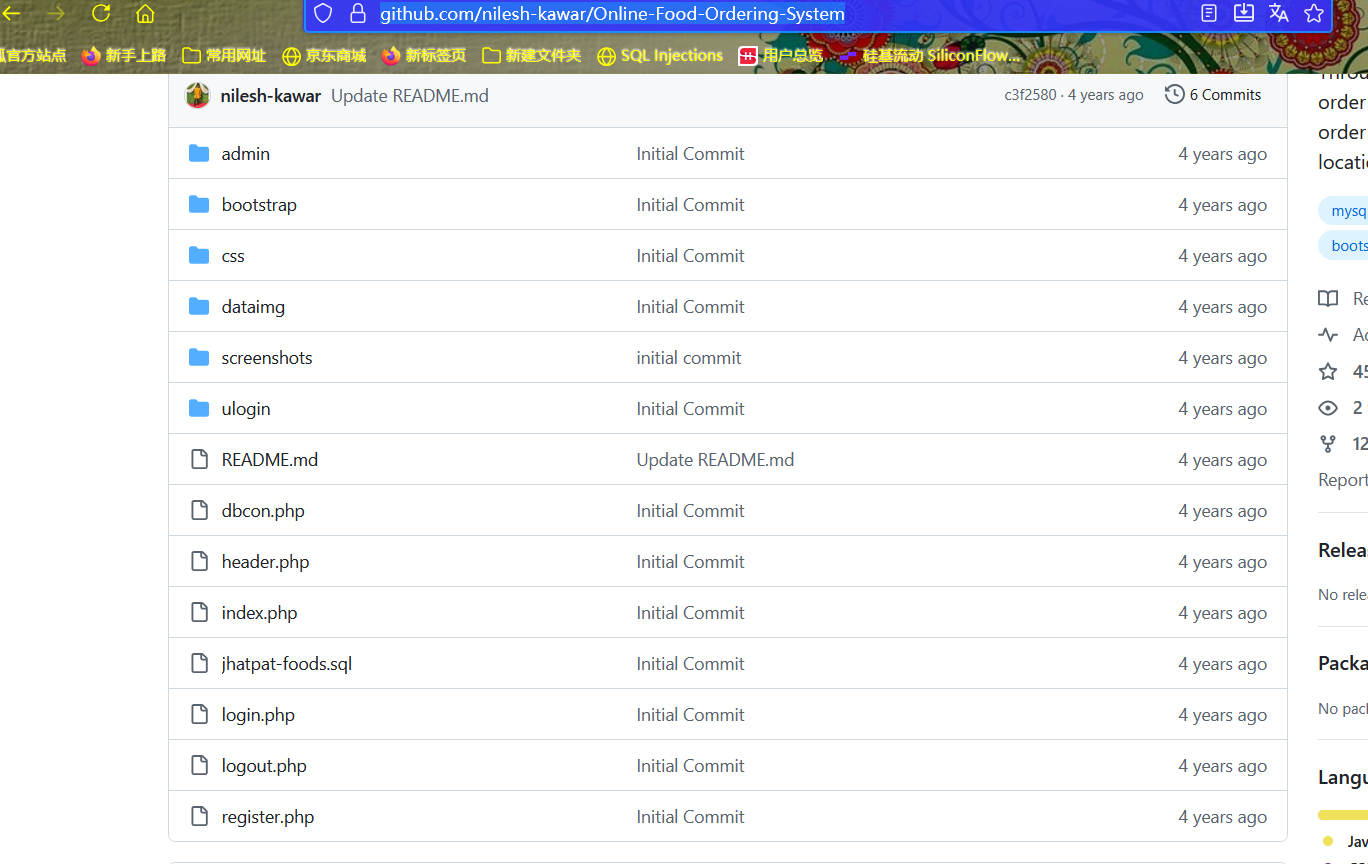
```

# Project Setup and Vulnerability Reproduction

## Download Link

https://github.com/nilesh-kawar/Online-Food-Ordering-System

## Directory Structure



## Setup Steps

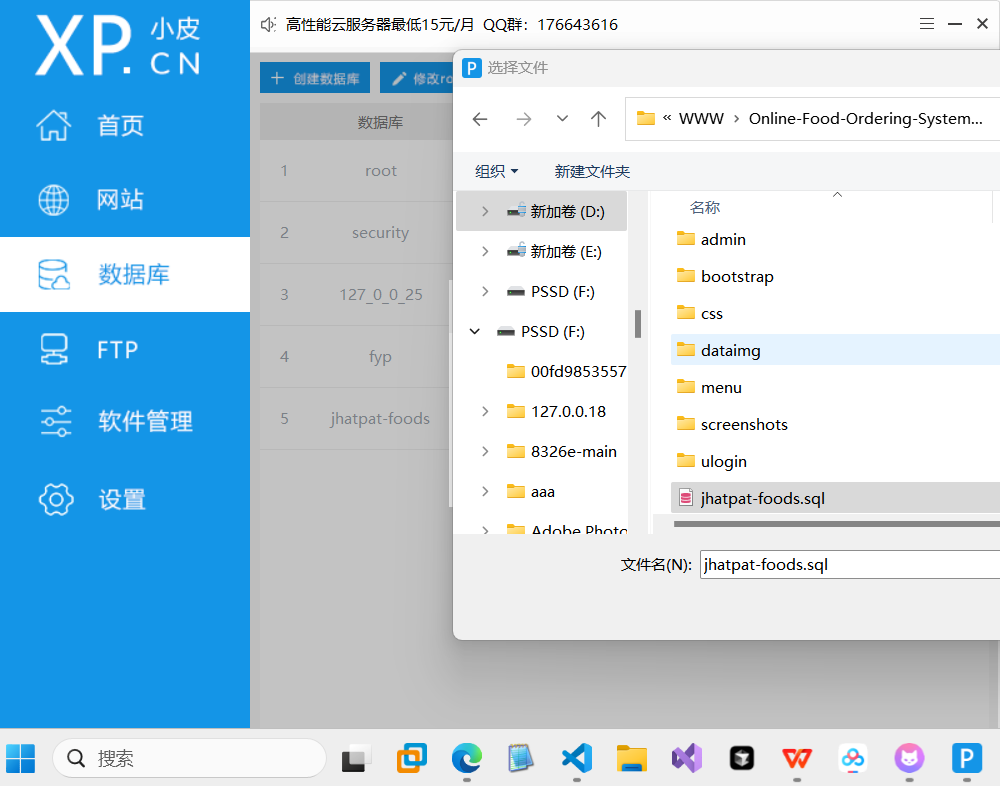
1. Launch phpstudy (with Apache + MySQL).

Place the project files into the \*\*Web root directory\*\* (`www`) of phpstudy.



3. Import the database

Create a new database named `jhatpat-food` and import the downloaded `jhatpat-foods.sql` file.



4. Modify the database configuration file (`dbcon.php`) to match your database settings.



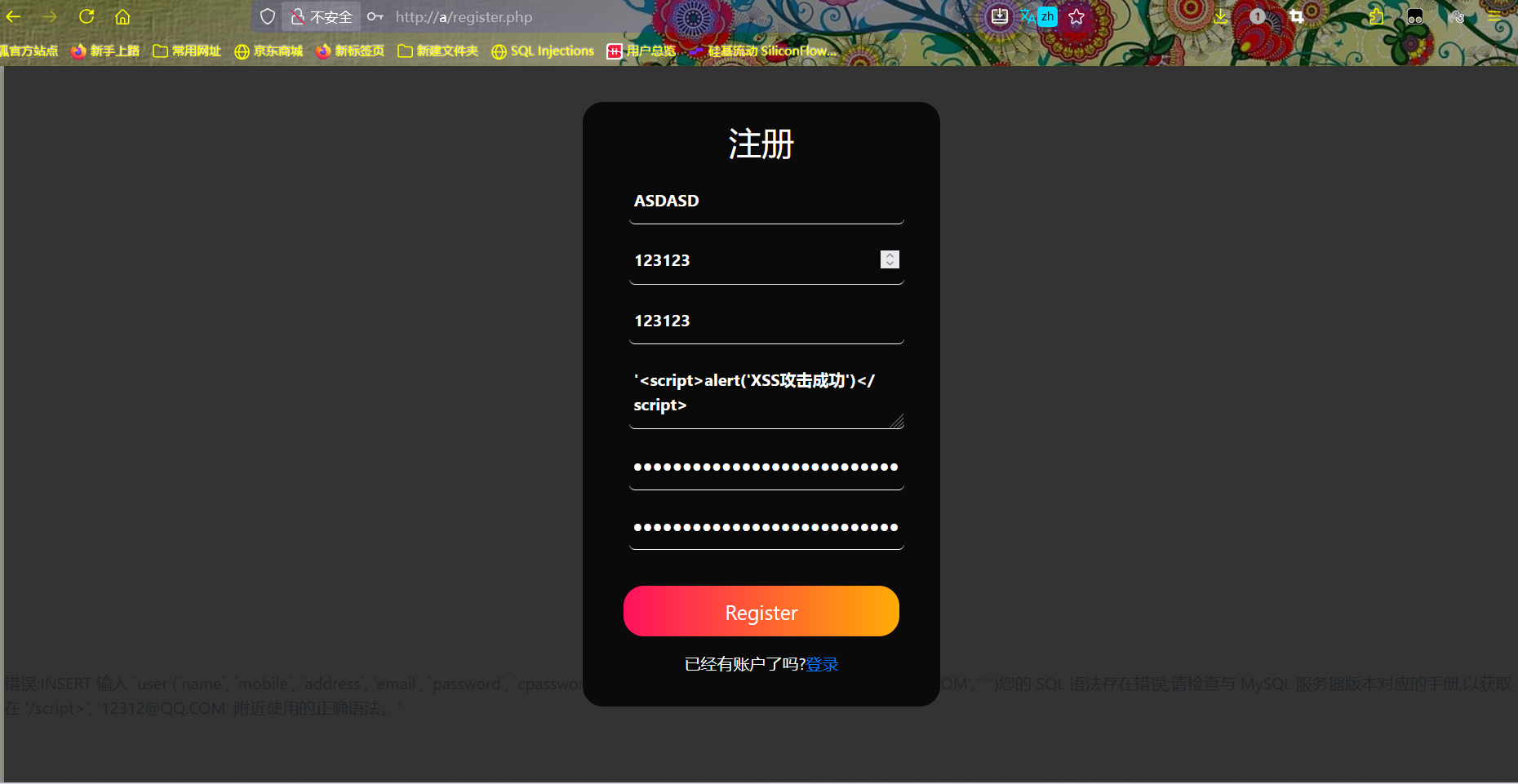
5. Access the website successfully via a browser:

Example URL: http://a/register.php

### Verify the Vulnerability

Input the following payload in the \*\*Address\*\* field (or other vulnerable fields like \*\*Name\*\*):

`<script>alert('XSS Attack Succeeded')</script>`



### Successful Alert Pop-up

