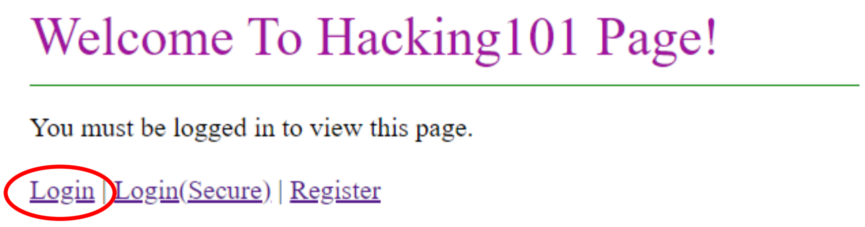
Vulnerable web application challenges

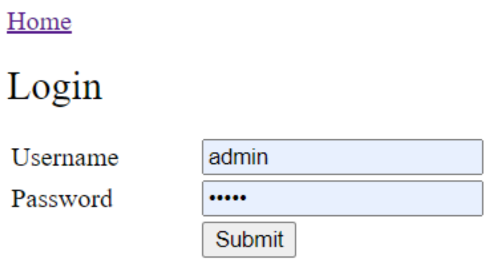
1) SQLi challenges (Bypassing authentication)

Participant required to login to the webapp as user **admin** without knowing the password.


Welcome To Hacking101 Page!

You must be logged in to view this page.

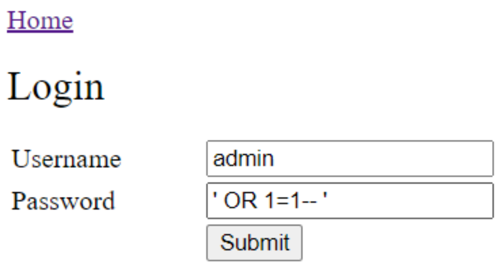
Participant can access the challenge via the Login function.



Participant may use the following payload as an injection input in the password field:

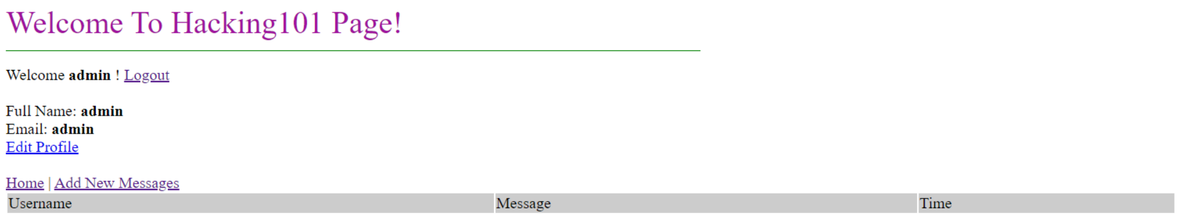
' OR 1=1-- '

The solution for the *Login* function is to input **admin** in username field, and the payload in password field (It would not work the other way around).



Username: admin

Password: ' OR 1=1-- '

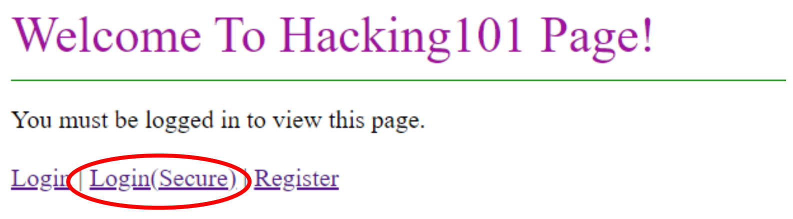

Welcome To Hacking101 Page!

Welcome **admin** ! Logout

Full Name: **admin**
Email: **admin**
Edit Profile

Username	Message	Time

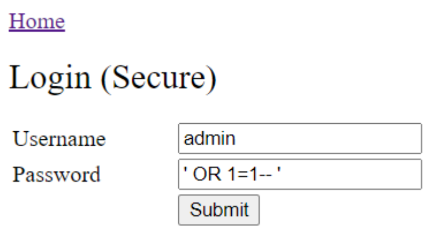
For the next SQLi scenario, participant required to do the same but the password field now is hashed before being processed.

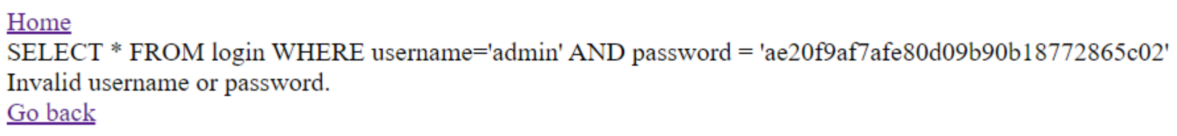

Welcome To Hacking101 Page!

You must be logged in to view this page.

Participant can access the next scenario via the Login (Secure) function.

If participant were to repeat the same technique from previous scenario, it would not work.



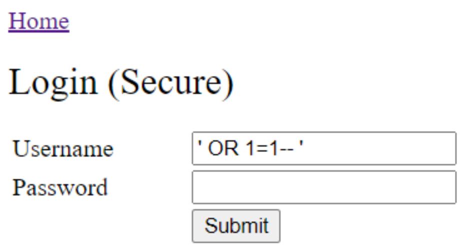


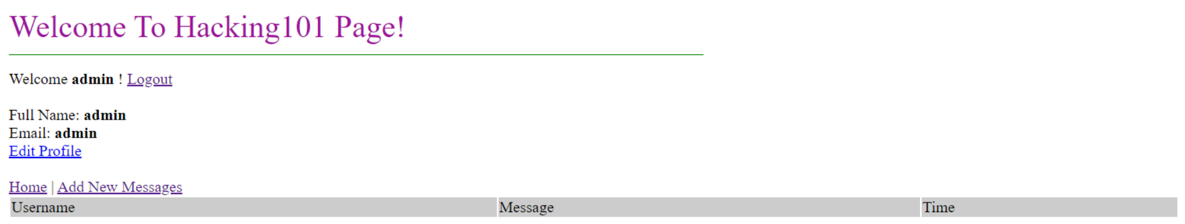
SELECT * FROM login WHERE username='admin' AND password = 'ae20f9af7afe80d09b90b18772865c02'
Invalid username or password.

The input in password is first hashed, rendered any injection attack useless.

The solution is to apply the injection on the username field this time, instead of the password field. The password field may be anything. Participant may use the same payload:

' OR 1=1-- '




Welcome To Hacking101 Page!

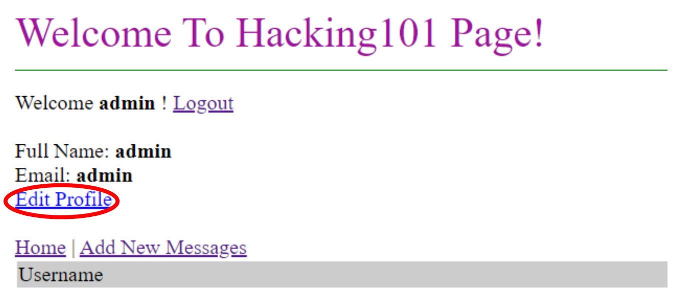
Welcome **admin** ! Logout

Full Name: **admin**
Email: **admin**
Edit Profile

Username	Message	Time

2) Cross-Site Scripting (XSS) challenges (Self-reflected)

Participant are required to execute arbitrary JavaScript code using XSS injection on vulnerable input field in *Edit Profile* page.



Welcome To Hacking101 Page!

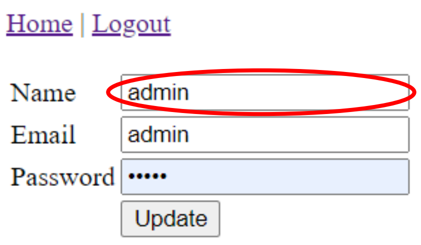
Welcome **admin** ! Logout

Full Name: **admin**
Email: **admin**
Edit Profile

Home | Add New Messages

Username

Participant can access the challenge via the Edit Profile function.



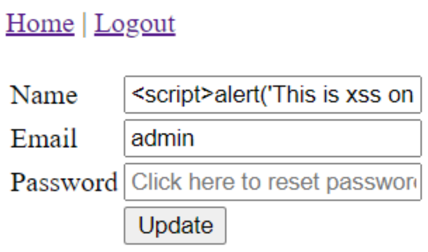
Home | Logout

Name	admin
Email	admin
Password	•••••
	Update

Participant may procced with the first xss challenge by trying out payload in the name field.

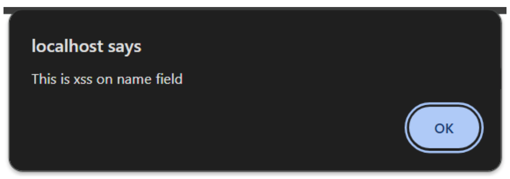
The solution is to test out the <*script*> tag input in the name field. Participant may use the the following payload:

<script>alert('This is xss on name field')</script>



Home | Logout

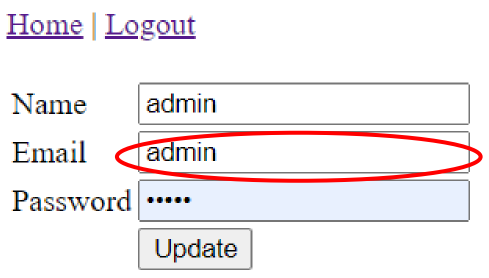
Name	<script>alert('This is xss on
Email	admin
Password	Click here to reset passwor
	Update



localhost says

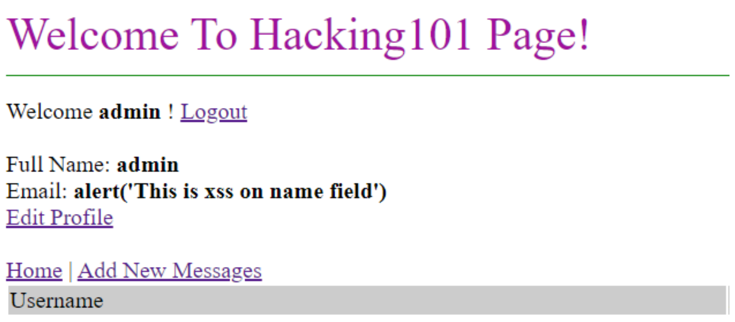
This is xss on name field

For the next XSS scenario, participant required to do the same but on the email field. Participant may notice that <*script*> tag is now being filtered out from the input.


Home | Logout

Name	admin
Email	admin
Password	•••••
	Update

Participant may procced with the next xss scenario by trying out payload in the email field.


Welcome To Hacking101 Page!

Welcome **admin** ! Logout

Full Name: **admin**
Email: **alert('This is xss on name field')**
Edit Profile

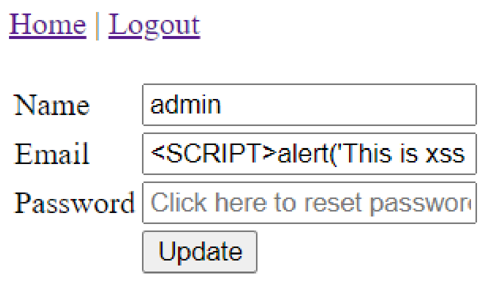
Home | Add New Messages

Username

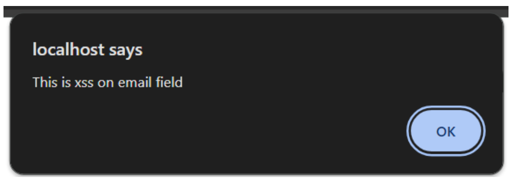
The input in email field is now filtering for <script> tag, rendered any injection attack useless.

The solution for this scenario is to apply the injection in a different input convention, the simplest one being by adjusting *the <script>* tag to be in **capital** case. Participant may use the same payload but with minor adjustment:

<SCRIPT>alert('This is xss on email field')</SCRIPT>


Home | Logout

Name	admin
Email	<SCRIPT>alert('This is xss
Password	Click here to reset passwor
	Update


localhost says

This is xss on email field