IDOR (Insecure Direct Object Reference)

What is Insecure Direct Object References Attack?

An insecure direct object reference (IDOR) is an access control vulnerability where invalidated user input can be used for unauthorized access to resources or operations. It occurs when an attacker gains direct access by using user-supplied input to an object without authorization. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability. Every resource instance can be called an object and often, represented with an ID. And if these IDs are easy enough to guess or an attacker can use an object to bypass the access check somehow, we can talk about an IDOR at this point. Referring to the above image, an attacker by ethical means can only get the document numbered 101 which is legally his. But what if the web application does not validate the number and an attacker puts the number of its victim? This can cause the attacker to get hold of the sensitive document to which he should not have access.

Severity:

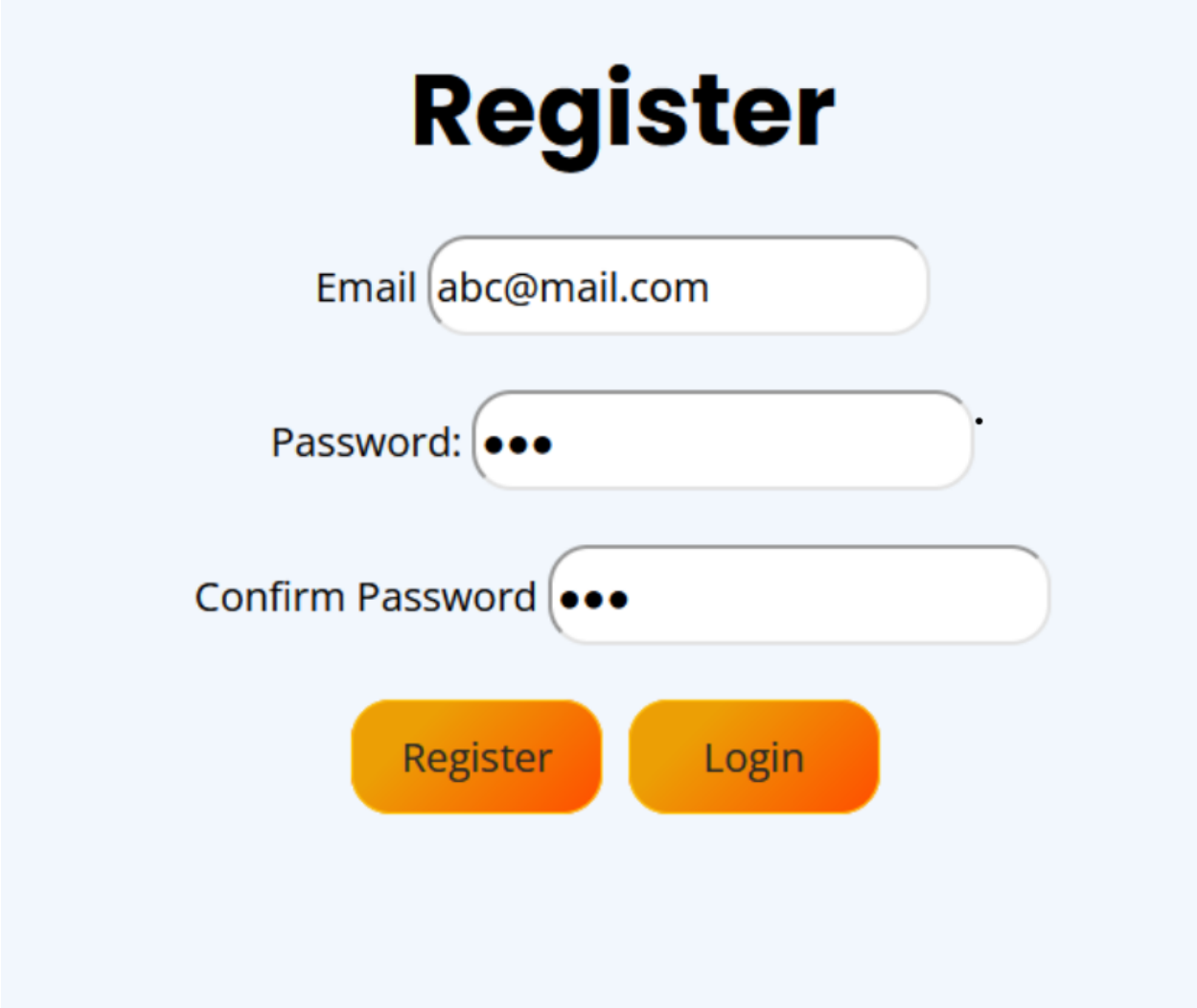
The severity of IDOR varies from P3 to P2 depending on what data is being exposed.

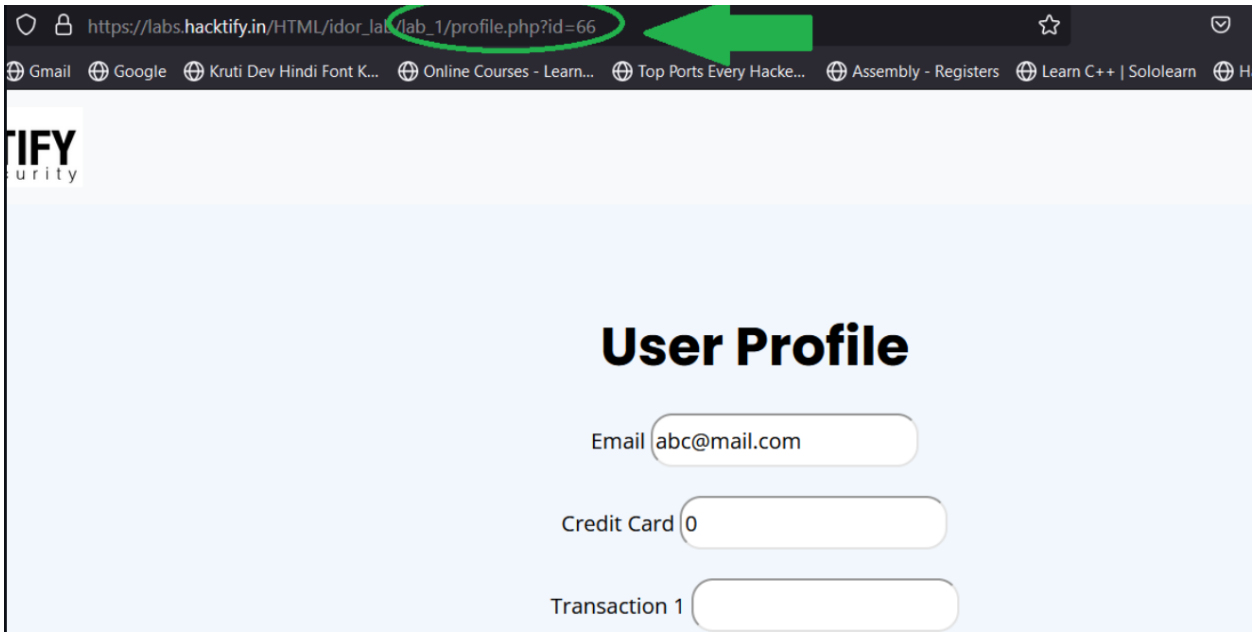
LAB 1: Give me my amount!!

Observation: On accessing the first lab of this module we can see the view hasn't changed much from that of previous labs, Now we can observe that we are given a login page with the name as, 'Bank Login Page', we can also see it has two identical buttons first for Login and later for Register as shown in the first image On Clicking the 'Register' button we can observe the following interface to add a new user,

Solution: IDORs are fun and have a good bounty! Now first things first let's go to the register page and make a user account. (email: abc@mail.com, Password: 123) Let's log in to our user account, using the set credentials, and we can see, The first thing we should know about coding a login page with multiple users is parameters should not be displayed and if displayed there should be checks such that these can't be tampered with. Now as in this URL as marked, we can see the "/profile.php?id=" id parameter, and that too has a numerical value telling us a lot about the backend architecture such as the database at the backend where user profiles are created would have following fields (id, Email, Password, Transaction 1, Transaction 2, Transaction 3), and my user is in the 66th row or is the 66th user. IDORs are particularly simple we just need to test over these open parameters for the unauthorized information disclosure of other users. Let's try changing the id value from "id=66" to "id=65", and we can observe And by this, we know our current lab site is vulnerable to IDOR vulnerability. Before we move on to the next lab let's see if can we get the details of all users. Let's turn up our Swiss army knife and use it for our testing. First like in the past we'll intercept this request when the browser is requesting for the user with

some ‘?id=’, Now send the following request to burp intruder, and click on the clear all button then select the value ‘65’ and click the add button, Now select attack type as ‘Sniper’ and in the payloads, option select ‘Numbers’, as shown, Hit the ‘Start Attack’ button, and analyze each response you got





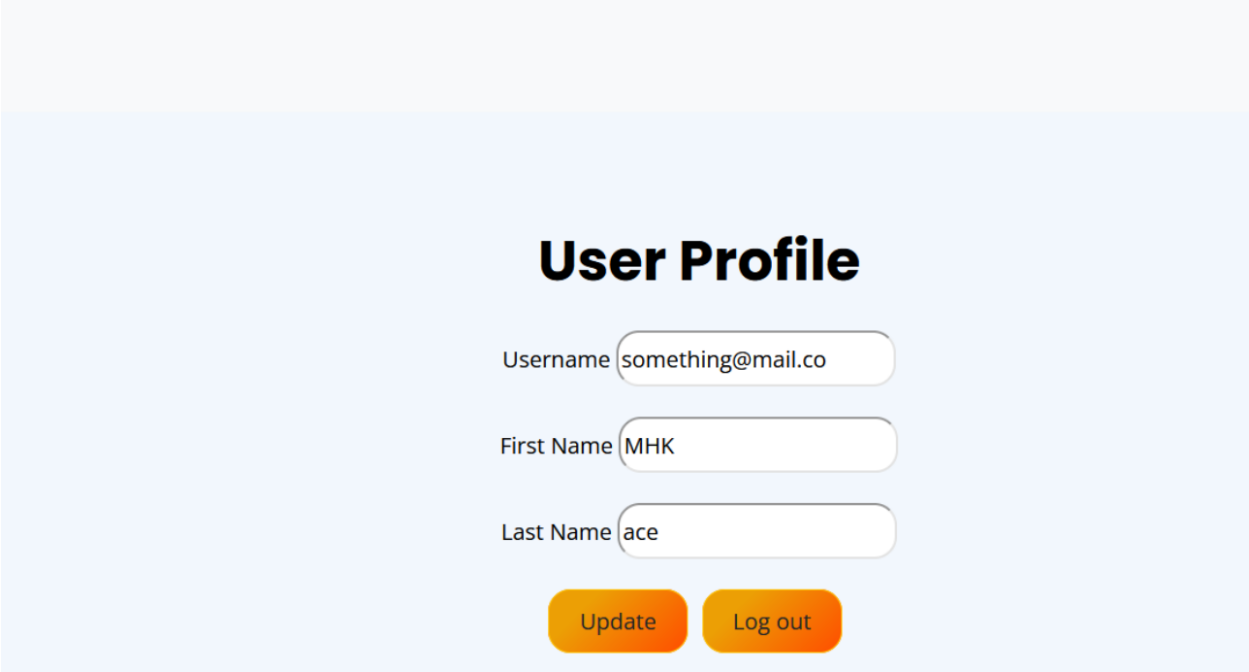
Now as we can see when we log in to our account we can also update our details, by far we know that we have the read access privilege to read the contents of the profiles of others, which is serious enough when we take into account that it's a bank application. To increase the impact of the existing vulnerability let's also test whether we can update other accounts or not. This is just the profile of 'id=33'

Now let's play with the input fields, like, Hit Update! Log out! Log in again! Navigate to the same id (in my case id=33), and check for yourself. After this we can suspect that this application has a common database for all so try and share a screenshot of the user with id = 33 of what you see (over Whatsapp or Discord).

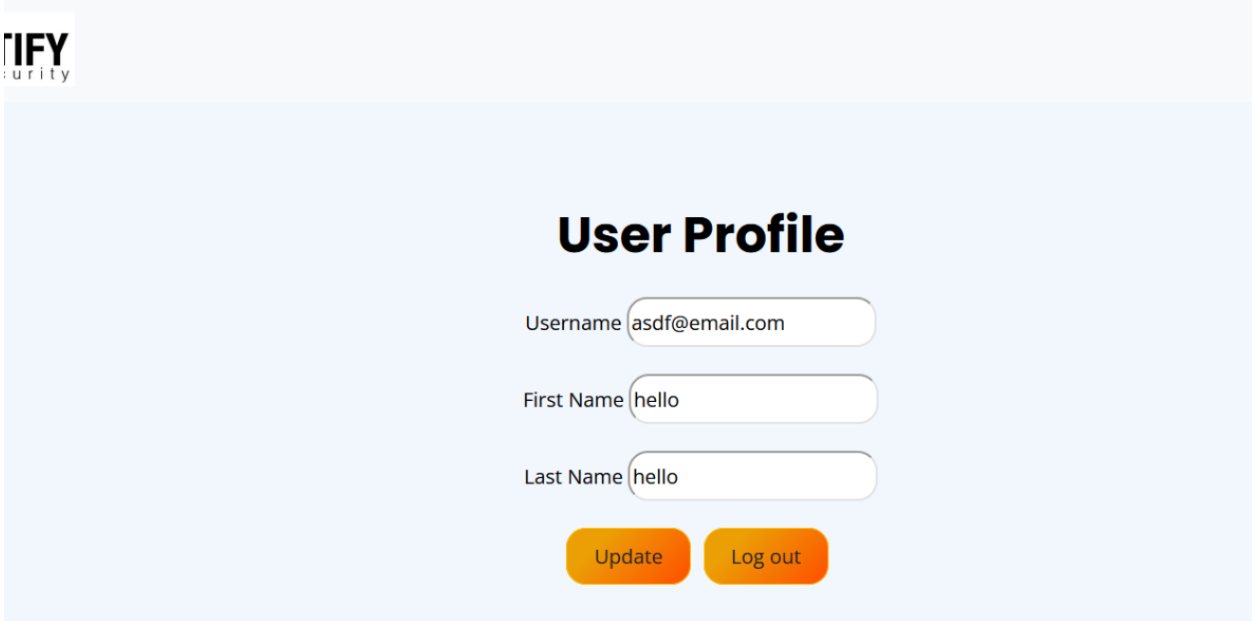
LAB 2: Stop polluting my params!

Observation: Upon accessing the lab we are thrown a login page with the fields asking for Email and Password combination, we also have two buttons each for login and Register.

Upon clicking the Register button we are given a page as follows,



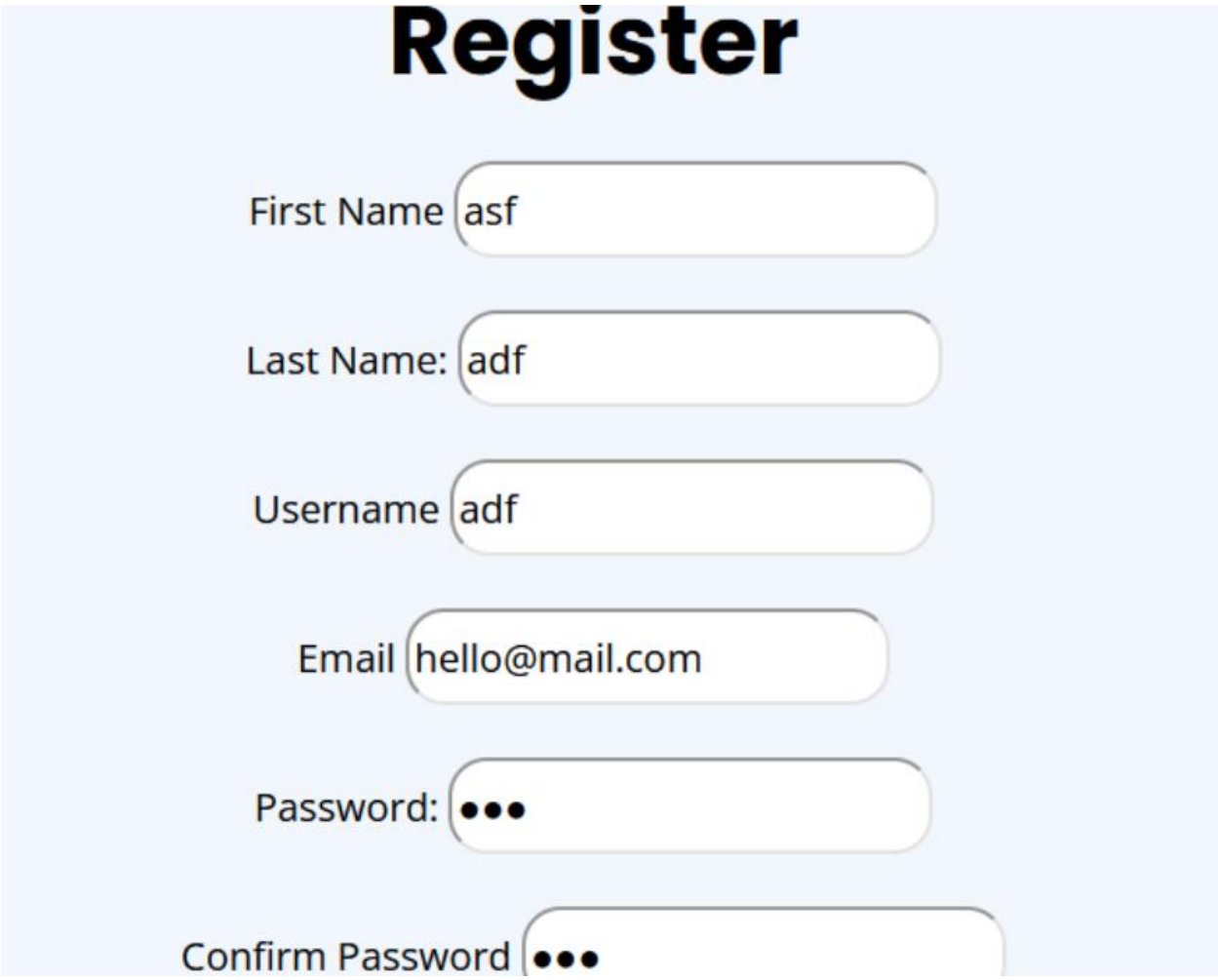
Solution: As from the last lab, we know how an IDOR vulnerability works, let's just create a new user account and see what more can we do with it…:) So using some random details to create the account such as, Here, we have email: password => something@mail.co: 123 Now we'll just register this user and log in with our credentials, When we log in, we get the following response for our 'User Profile', As we did in the previous lab, let's try changing the value of the '?id=" parameter, we get, We changed our value to 20, This is similar to what we did in the last lab and here in this lab we were not allowed to update the accounts of other users too… So same for 2 labs doesn't seem right to me!! Also as we practiced that during challenges, labs, or Ctfs the biggest hint is the name for the one… So after reading the name, this was one of the findings that the majority of the people left, now a question can we launch our IDOR in some other way?? The answer to this is yes, what if this is a real-world scenario and you are not getting the accounts of other users, would you just give up? The lab says, "Don't pollute my parameters." so from here we might have heard or if not we'll know that there is an attack called HTTP parameter pollution, HTTP Parameter Pollution: Basically, if we say in easy terms this is not a bug but is used as one because there is no particular standard for servers to handle the parameters. So different server behaves differently when we give multiple parameters of the same name to them. Let's now do it practically for better understanding so now the payload we are going to use is, Payload: &id=20 Don't just start throwing queries just now, we are still not done with it…. Using this parameter, On requesting the server with the following URL we get the following response, Now let's try to add one more ID parameter and see the result, So by this, we can at least conclude that this lab is also vulnerable to HTTP Parameter Pollution, and the server will give us the response for the parameter placed far towards the right.



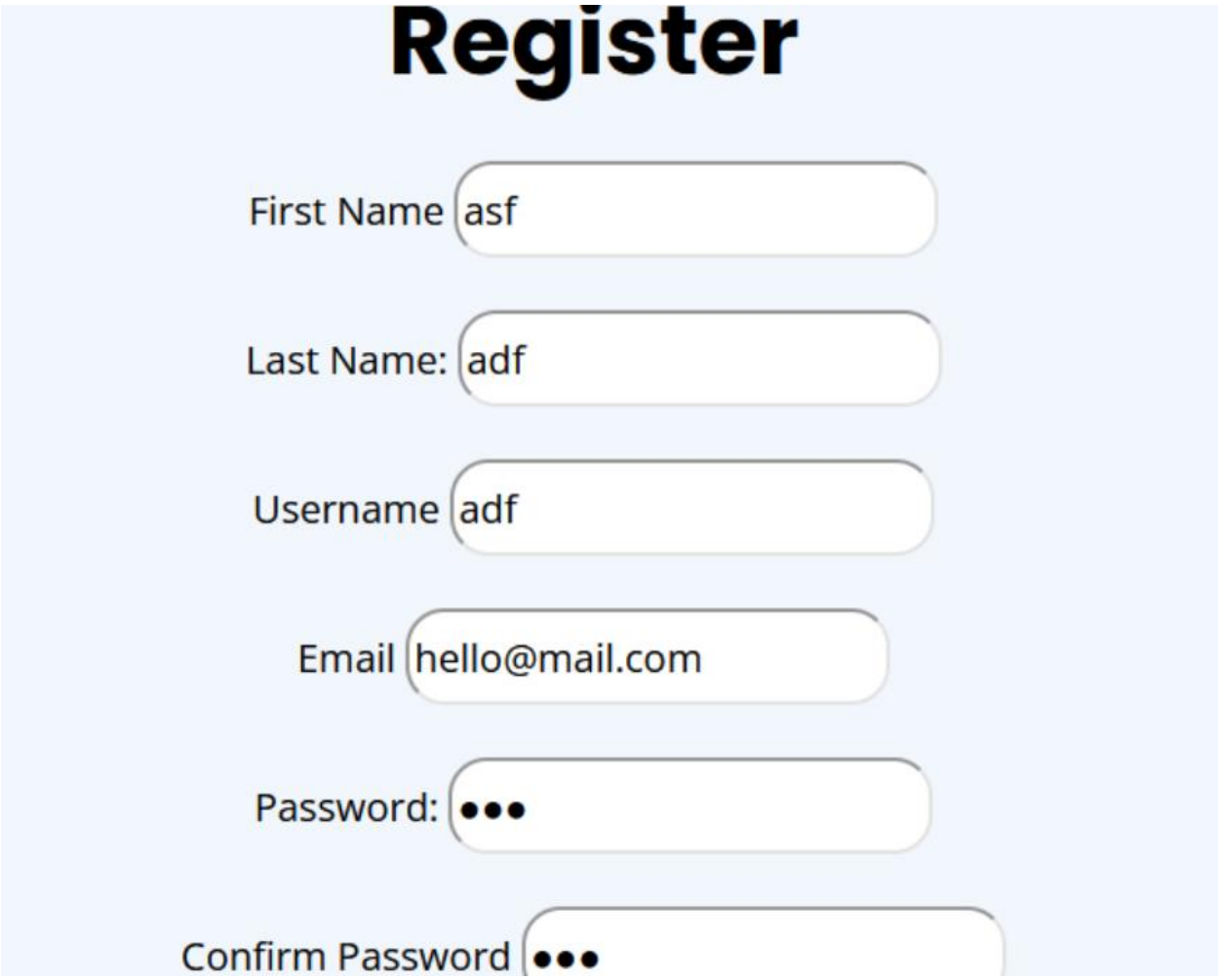
Hence we found that this lab has two different ways to access user accounts, The First is our typical IDOR, Second and the new one is by HTTP Parameter Pollution. :)

LAB 3: Someone changed my Password 🤔!

Observation: Even before we access the lab the name tells us there will be a user login page where we can access the account for other users and change their passwords…. Huh!! Let's access the lab and we observe that we are given a login page, such as, Also, upon clicking the register button we get a user registration page in response which is as follows,



Solution: First, let's create 2 user accounts, and then we can follow the steps as the previous labs and then let's see what plays out, The first account is something like this,

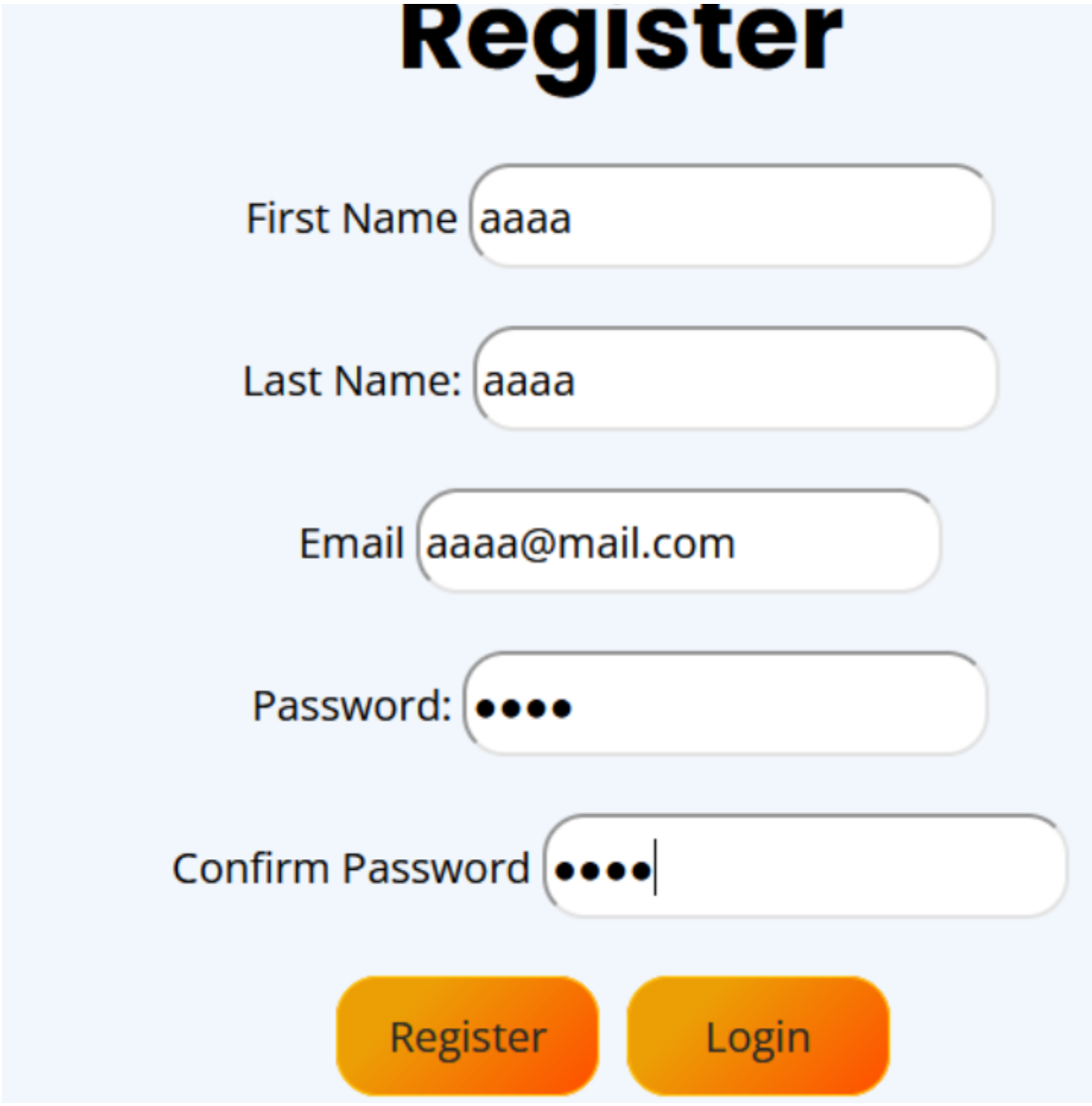


And the second account is something like this,

LAB 4: Change your methods!

Observation: On accessing the lab we are given a login page, which is as,

Solution: Sorry, but the write-up for this one will be concise as we don't have any way to check the result… Also, I am traveling So we'll create two accounts as follows, Account 1: (aaaa@mail.com:aaaa) Account 2: (cccc@mail.com:cccc)



Initially, when we try to change the 'id' parameter we have a successful response, but we cannot see any data so it is impossible to verify the success of the methodology and this is not the intended process. Now, we logged in with our Account 2 and changed the parameter value to that of Account 1, let's try updating the fields, By looking at the intercepted request we can notice quite a few things, First, the username field is linked to

the email parameter, which means we could try updating the email of Account 1 if we can log in with the same, and the id parameter remains that of Account 1, that means our attack was successful, also we could notice that the request method is set to 'GET' and the id field goes empty, quite a few interesting things we can play along with, Let's send this request to the repeater and now we made a few changes as, And send it, now we don't know what happened at the backend!! Updating the 'username' field was a great idea but that does not change the login email id of the user!! That's for sure!! Because you won't be able to login with the updated email-id. Personal opinion they should have made it vulnerable to SSRF too, so that at the very least we could chain IDOR with SSRF to atleast get a pingback for confirmation that the updates are being made!! But nonetheless I am also lost in this one, as we don't know anything until we can check backend.