

Topic	IDOR Attack	
Class Description	Students will learn about IDOR attack, what it means and they will be performing a brute force attack to fetch sensitive data.	
Class	C-235	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Learn about IDOR attack in ethical hacking • Perform a brute force attack to fetch sensitive data from the website 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code 	
Class structure	Warm-Up Teacher-led Activity 1 Student-led Activity 1 Wrap-Up	10 mins 15 mins 15 mins 5 mins
WARM-UP SESSION - 10mins		
Teacher Action		Student Action

<p><i>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</i></p> <p>In the last session, we learned how to perform complex queries, with multiple UNION and JOIN statements. We also performed a SQL injection that fetched other user's login credentials for us.</p> <p>Any doubts from the last session?</p> <p><i>The teacher clarifies doubts (if any)</i></p> <p><i>We have covered a lot of things in SQL, and we also saw how helpful it can be, when we want to fetch other data from a website by performing SQL injection, but it is not the only technique out there for ethical hacking. In today's class, we will be learning about IDOR attacks.</i></p> <p><i>Let's get started</i></p>	<p>ESR: Hi, thanks, yes, I am excited about it!</p> <p>ESR: Varied!</p>
<p>Q&A Session</p>	
Question	Answer
<p>Which SQL operator is used to give variable names to tables/columns -</p> <ol style="list-style-type: none"> 1. SELECT 2. SET 3. WITH - AS 4. AS - WITH 	<p>C</p>
<p>Which of the following is True?</p> <ol style="list-style-type: none"> 1. SET operators need equal number of columns selected from the tables 2. UNION operator does not remove duplicates 	<p>A</p>

3. UNION ALL keeps all the duplicates 4. SET operator works regardless of data type of the columns	
TEACHER-LED ACTIVITY - 15mins	
Teacher Initiates Screen Share	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Breaking down complex SQL statements • Joining multiple join statements 	
Teacher Action	Student Action
<p>In today's class, we will be learning about one of the very famous attacks called the IDOR attack. It is relatively a very simple hacking technique that is used to fetch unauthorised data. Before diving deep into it, let's see an example -</p> <p>Please refer to Student Activity 1</p> <p><i>Teacher opens the Teacher Activity 1</i></p>	<p><i>Student opens the Student Activity 1</i></p>

Toy-e-Wagon



Login

Login into your account to view our products and access your profile to track orders

E-mail address

Password

Login

Now let's login into the website!

For simplicity, we can use the following credentials -


Email - john.doe@gmail.com





Password - hello_john

Teacher logs into the website

Student observes

Toy-e-Wagon[Dashboard](#)[Profile](#)[Logout](#)[Help](#)





Okay, now let's scroll down a little, and click on the **“Buy Now!”** button on any of the products!

Toy-e-Wagon

Dashboard
 Profile
 Logout
 Help

address

Addresses

521 Aviation Way, Burbank, California, USA - 91504

New Shipping Details

House Number

City

State

Country

PIN Code

Product Details

Sergeant Rodog AI
 Delivered within 5-7 business days

Subtotal

\$94.99

Delivery Charges

\$0.49

Total Amount

\$95.48

Place Order

Great, now let's look at the URL closely. Do you notice anything?

We could see that the end of the URL has a URL parameter, **id** with some value -

ESR:
Varied!

```
ec2-3-13-85-11.us-east-2.compute.amazonaws.com/order?id=1
```

Note - In our example, the id is 1.

As you can see, we do have an ID here in the URL. What could this mean?

Let's try and change the value of this field!

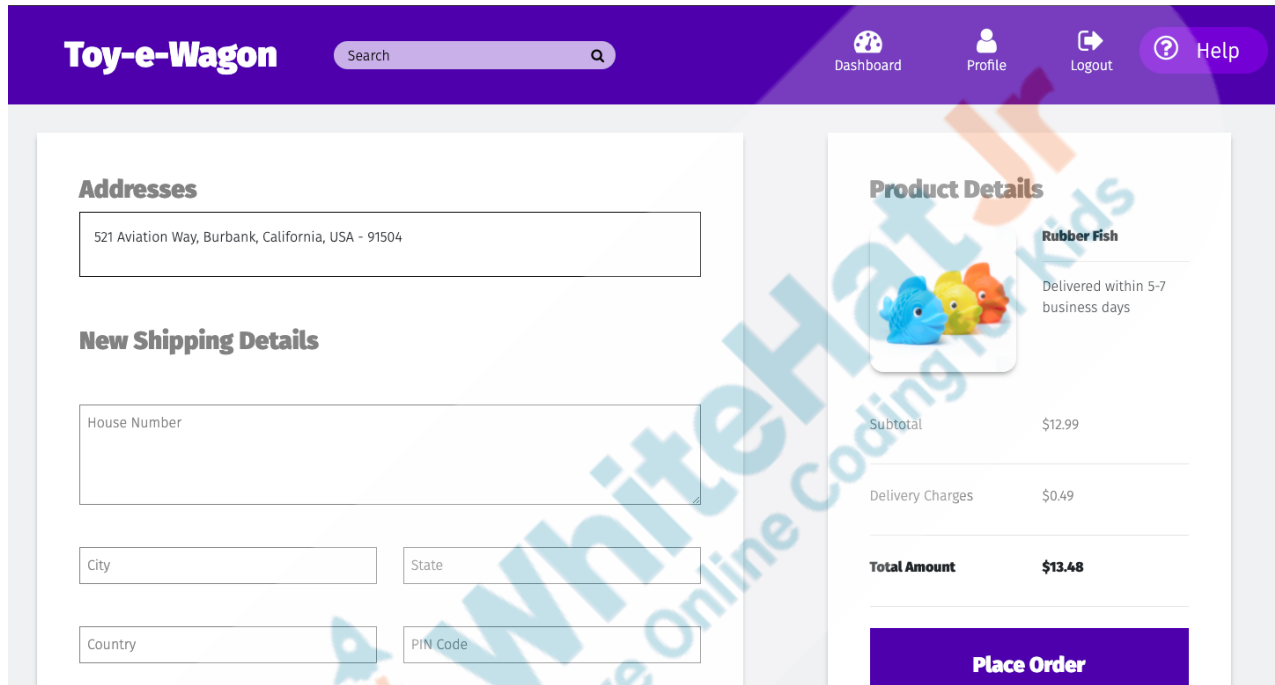
Teacher changes the value of it. Make sure to enter it somewhere b/w 1-11 or you will see an error.

Student observes

With URL -

`ec2-3-13-85-11.us-east-2.compute.amazonaws.com/order?id=10`

We get the output as -



The screenshot shows the 'Toy-e-Wagon' website. The header is purple with the site name, a search bar, and links for Dashboard, Profile, Logout, and Help. The main content area is divided into two columns. The left column contains a section for 'Addresses' with a text box showing '521 Aviation Way, Burbank, California, USA - 91504', and a section for 'New Shipping Details' with input fields for House Number, City, State, Country, and PIN Code. The right column contains a 'Product Details' section for 'Rubber Fish' with a delivery time of '5-7 business days'. Below this is a table showing 'Subtotal' as \$12.99, 'Delivery Charges' as \$0.49, and a 'Total Amount' of \$13.48. At the bottom of the right column is a purple 'Place Order' button.

Did you notice what happened?

That's right! This page gets to know which product the user wants to buy through the URL. It simply takes the product based on the ID that is mentioned.

Do you think this is a good design?

This is known as an Insecure Direct Object Reference

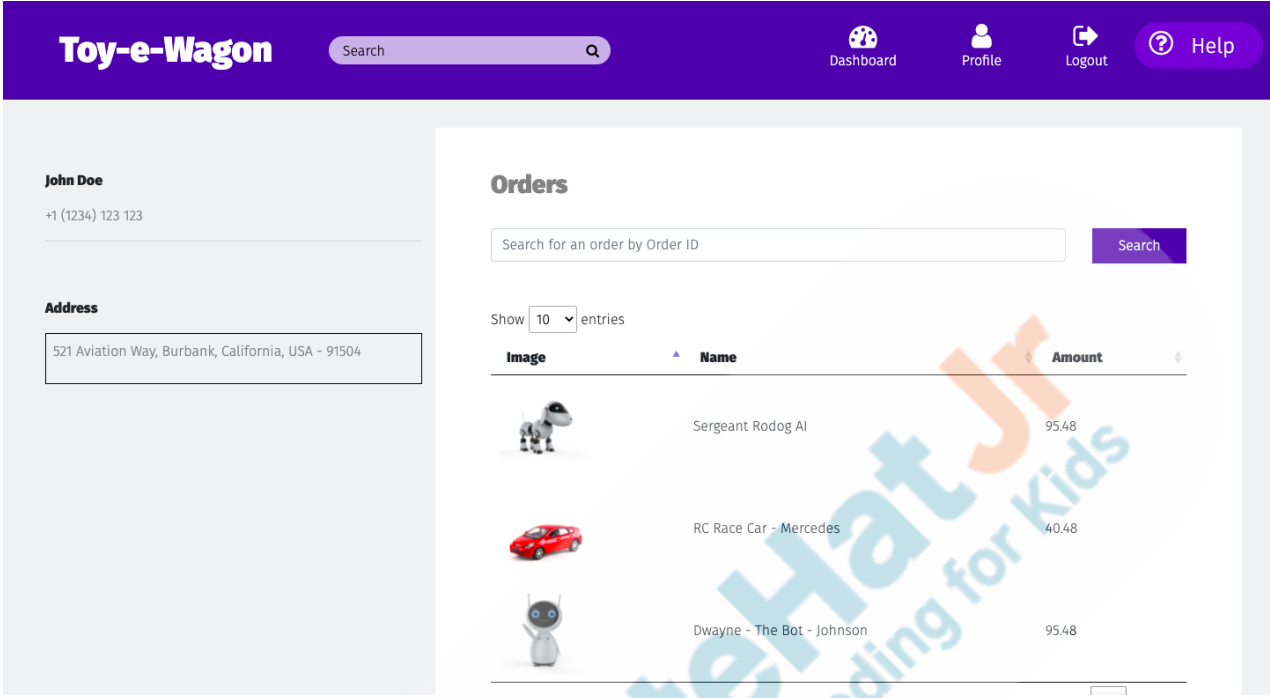
ESR:

Yes! On changing the id value, the product changed!

ESR:

Varied!

<p>(IDOR). This vulnerability occurs when any application uses user-supplied information to access objects directly, without validating it.</p> <p>For example, here, we wanted to buy a different product, but just by changing the URL, the product displayed to us becomes different, and if we place an order now, we can actually place the order on the new product instead of the one we wanted to buy earlier.</p> <p>Nowadays, many websites by design, use this design where the page knows what exactly to display based on URL, such as posts on a social media platform, etc. but they are designed in a way where the access is still controlled from the backend.</p> <p>This means that, if I were trying to open a post that I don't have access to through a URL, it would not open for me.</p>	
<p>Here, in the example that we just saw, we were able to access a different product through a URL, but no harm was done and we did not access any data that we shouldn't have access to in the first place, but let's look at a different example now.</p> <p>Let's go to the profile page now from Navbar and observe the URL there.</p> <p><i>Teacher moves to the profile page</i></p>	<p>ESR: Yes!</p> <p><i>Student observes</i></p>

	
Now let's observe the URL.	
<pre>ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=1</pre>	
<p>Here, you would notice that even in the profile page, we have id as 1.</p> <p>Let's try to change the value of this id and try 2!</p> <p><i>Teacher changes the ID in the URL</i></p>	<i>Student observes</i>
<pre>ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=2</pre> <p>Gives -</p>	

Toy-e-Wagon

Dashboard
 Profile
 Logout
 Help

James Bond
 +37 (1234) 567 890

Address

Orders

Show 10 entries

Image	Name	Amount
	None	95.48
	None	40.48
	None	95.48

Here, we can notice that our data in the profile page changed.

We can see some of the data is now updated. For example, the name and the address in this case is different -

Teacher observes

Student observes

Name and Address for id=1

John Doe

+1 (1234) 123 123

Address

521 Aviation Way, Burbank, California, USA - 91504

Name and Address with id=2

James Bond

+37 (1234) 567 890

Address

Here, we can see that the name and the contact information for both the IDs are different, and one of them has an address while the other one doesn't.

This means that the profile page is fetching data based on the ID that we are providing on the URL.

<p>By design, do you think it is a good design?</p> <p>That's right. There could be sensitive data in the profile page, such as passwords, or payment details, etc.</p> <p>By this design, anyone can access other user's data through tweaking the URL, and this is a demonstration of an IDOR vulnerability.</p>	<p>ESR: No!</p>
<p align="center">STUDENT-LED ACTIVITY - 15 mins</p>	
<ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Full Screen. 	
<p align="center"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Fetch URLs containing sensitive data from the profile page 	
Teacher Action	Student Action
<p>Okay, so now we have seen how an IDOR vulnerability can be accessed to fetch unauthorised data, just by tweaking the URL.</p> <p>We have seen what happens when the user with the given ID exists. Let's also see an example on what happens when the user does not exist.</p> <p>Let's try an ID of 100</p> <p><i>Teacher guides the student</i></p>	<p><i>Student tweaks the ID</i></p>

```
{"message":"'NoneType' object is not subscriptable","status":"error"}
```

Here, we can observe that this particular id gives us an error, which says that the **“object is not subscriptable”**

This somewhat means that it was not able to find the user object with the help of ID, and couldn't fetch the user's data in the process.

Now, there are going to be many users, each one of them with their unique IDs.

If we were to find all the users of this website with IDs ranging from 1 to 1,000 - how are we going to do that?

That's right! We can find out about all the users data, by performing a brute force attack.




ESR:

We can perform a Brute Force Attack!

<p>Now, the way it would work is that we are going to iterate over all the URLs with IDs 1-100 - but how are we going to decide if this page contains the data or not? Which is the easiest way to ensure this?</p> <p>That's right! We know that when we try to make a request to the page and it gives us a status code of 200, it means that the request has been successful.</p> <p>Which status code lets us know that it could not find what we were looking for?</p> <p>Awesome! So then, we now know what we want to code. To make a GET request on the page, you can use the - requests.get() function.</p> <p>Let's write the code to get all the valid URLs now.</p> <p><i>Teacher guides the student in writing the code in a new VS Code Editor</i></p>	<p>ESR: Status Code!</p> <p>ESR: 400</p> <p><i>Student writes the code in a new VS Code Editor</i></p>
<pre>import requests for i in range(1, 100): URL = f"http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id={i}" r = requests.get(url=URL) if r.status_code == 200: print(URL)</pre>	
<p>Here, we can see that we have simply imported the requests library, and then we have created a loop.</p> <p>Inside the loop, we are first defining our URL, with the value of our number as the ID, and we are using the</p>	

<p>requests.get() function to retrieve the data.</p> <p>We are then using the status_code attribute to get the value of the status code of the response, and if it is 200, we are printing the URL.</p>	
<p>On running the code, the output should look like the following -</p>	
<pre> http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=1 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=2 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=3 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=4 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=5 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=17 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=23 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=29 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=33 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=45 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=57 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=64 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=72 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=83 http://ec2-3-13-85-11.us-east-2.compute.amazonaws.com/profile?id=88 </pre>	
<p>Now these URLs can be used to scrape data from the page, or a hacker could use this retrieved data to sell it to other competitors.</p> <p>We can always add some measures in the backend, that check if the user has access to the requested data or not, even if it is getting fetched based on some value on the URL, but the applications nowadays also focus on making it difficult for the hacker to guess the sequence nowadays.</p>	

Here, we have just numbers, but it could also be a unique ID that is completely random and is even hexa-decimal 16 digit strings, etc.	
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
Quiz time - Click on in-class quiz	
Question	Answer
What is the full form of IDOR? 1. Insecure Database Object Relations 2. Insecure Direct Object Relations 3. Insecure Database Object Relations 4. Insecure Direct Object Relations	D
IDOR attack can be performed by - 1. Tweaking the URL 2. Changing the browser 3. Changing the system 4. Changing the internet connection	A
Which of the following would be used to make a post request? 1. requests.get() 2. request.post() 3. requests.post() 4. request.get()	C
End the quiz panel	
<p align="center"><u>FEEDBACK</u></p> <ul style="list-style-type: none"> ● Appreciate the students for their efforts in the class. ● Ask the student to make notes for the reflection journal along with the code they wrote in today's class. 	

Teacher Action	Student Action
<p>You get Hats off for your excellent work!</p> <p>In the next class we will learn about SQL Union</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div>
<p>Project Discussion</p> <p>You have discovered a small API in a company's website that can be used to fetch their user's data without access or authorisation. You want to report about this bug to the company and for that, you need to perform a brute force attack to show them how you accessed and fetched the data without even logging in.</p> <p>Write a brute force attack to show them how you fetched the data.</p>	
<p>Teacher Clicks</p>	<p>✕ End Class</p>
ADDITIONAL ACTIVITIES	
Additional Activities	<p><i>The student uses the markdown editor to write</i></p>

Encourage the student to write reflection notes in their reflection journal using markdown.

her/his reflections in the reflection journal.

Use these as guiding questions:

- What happened today?
 - Describe what happened.
 - The code I wrote.
- How did I feel after the class?
- What have I learned about programming and developing games?
- What aspects of the class helped me? What did I find difficult?

ACTIVITY LINKS		
Activity Name	Description	Link
Teacher Activity 1	Ecommerce Website	http://ec2-3-108-196-161.ap-south-1.compute.amazonaws.com/
Teacher Activity 2	Reference Link	https://github.com/pro-whitehatjr/PRO-C235-Reference-Code
Student Activity 1	Ecommerce Website	http://ec2-3-108-196-161.ap-south-1.compute.amazonaws.com/