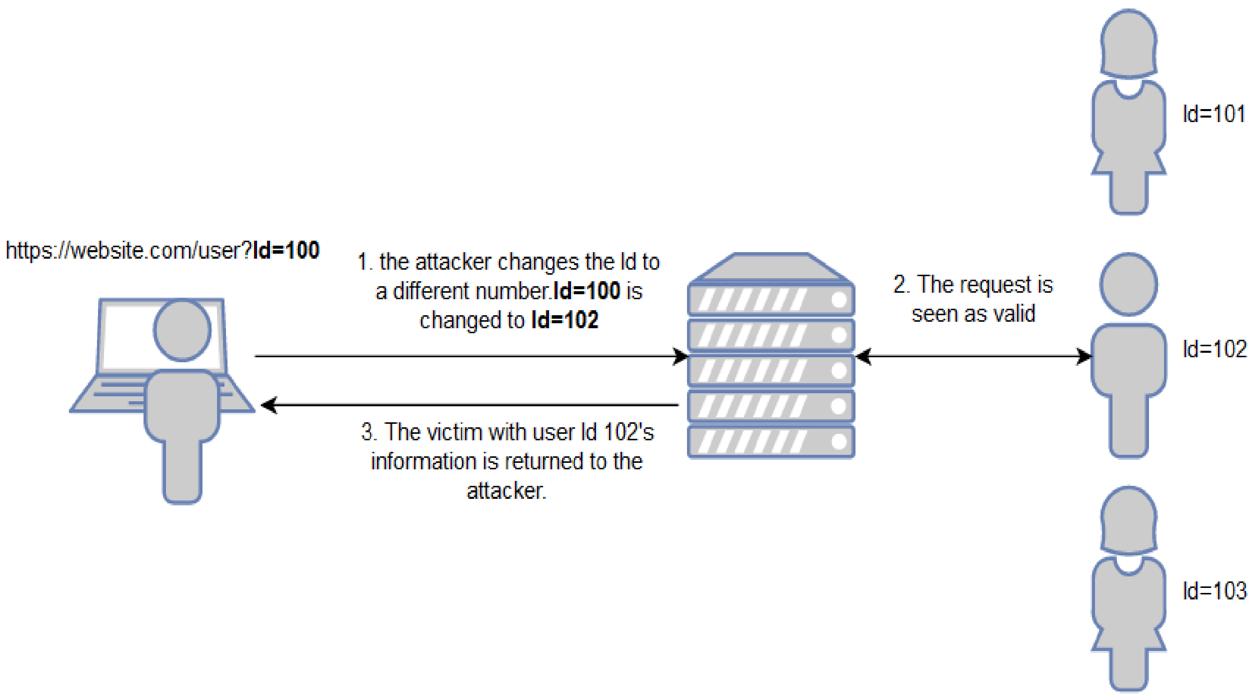
IDOR

☐ ***What is IDOR?***

It stands for Insecure Direct Object Reference. A vuln gives attackers an unauthorized access to retrieve objects such as files , data or docs. It was listed 2021 OSWAP top 10 under broken access control.
In order to find IDORs, the method that can be used is to create 2 accounts for an application and capture the HTTP requests with Burp Suite, or another proxy. These requests can be compared and the string that is different could be an IDOR, as this is a unique value to the user. The test to verify whether this object is vulnerable is to repeat the request from one user in the context of the other user and see if data is returned.


https://website.com/user?**Id=100**
1. the attacker changes the Id to a different number.**Id=100** is changed to **Id=102**
2. The request is seen as valid
3. The victim with user Id 102's information is returned to the attacker.
Id=101
Id=102
Id=103

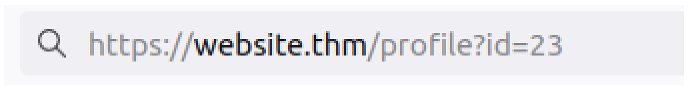
☐ ***How to find and exploit IDOR vulnerabilities?***

As previously mentioned, an IDOR vulnerability relies on changing user-supplied data. This user-supplied data can be found mainly in the following three places:

- Query components
- Post variables
- Cookies

1. Query Components:

Component data is pass5ed by the URL when making a request in a website.


https://website.thm/profile?id=23

The URL can be breakdown into:

Protocol: https://

Domain: website.thm

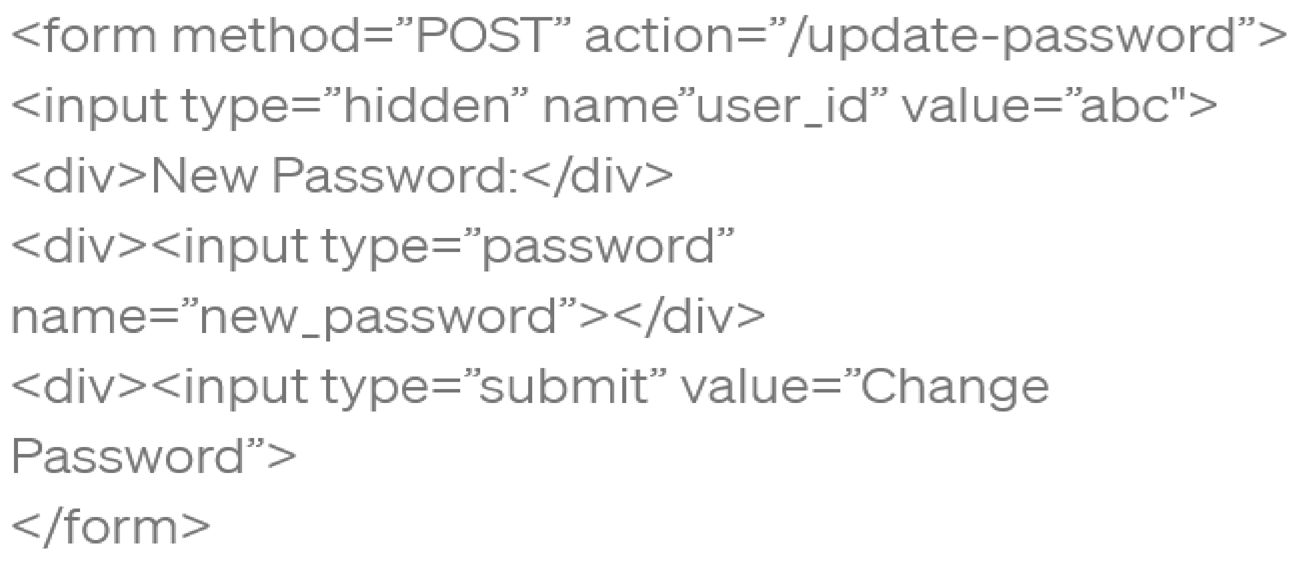
Page: /profile

Query component: id=23

Here we can se that /profile page is being requested and the parameter **id** of value **23** is being passed in the query component. This page could potentially be showing us personal user information. If we change the id parameter to some other value, then we could view other users data.

2. Post Variables:

Sometimes examining the contents of forms on a website can reveal fields that could be vulnerable to IDOR exploitation. For example, take the following HTML code for a form that updates a user's password.


```
<form method=”POST” action=”/update-password”>
<input type=”hidden” name”user_id” value=”abc">
<div>New Password:</div>
<div><input type=”password”
name=”new_password”></div>
<div><input type=”submit” value=”Change
Password”>
</form>
```

You can see in the **<input>** tag that the user's id is being passed to the webserver in a hidden field. By changing the value parameter to another user_id may result in changing the password for another user's account.

3. Cookies:

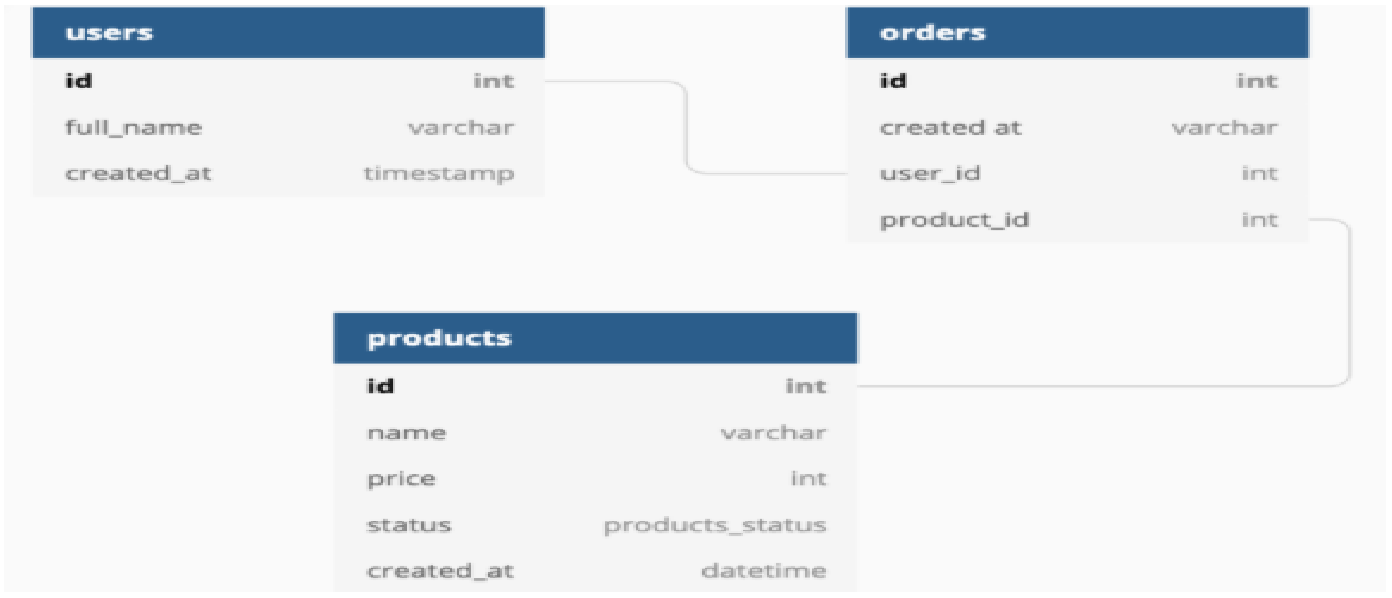
Cookies are used to remember your session to stay logged into a website. Usually, this sends the session is which is a long string of random text which is hard to guess such as **8dghlsjh9k4dfkmzakfi3hr0k3n,** which the webserver securely uses to retrieve your user information and validate your session. Sometimes, less experienced developers may store user information in the cookie itself, such as user's id. Hence, by changing the value of this cookie could result in displaying another user's information. You may refer the below example of how they might look.

```
GET /user-information HTTP/1.1
Host: website.thm
Cookie: user_id=9
User-Agent: Mozilla/5.0 (Ubuntu;Linux) Firefox/94.0

Hello Jon!
```

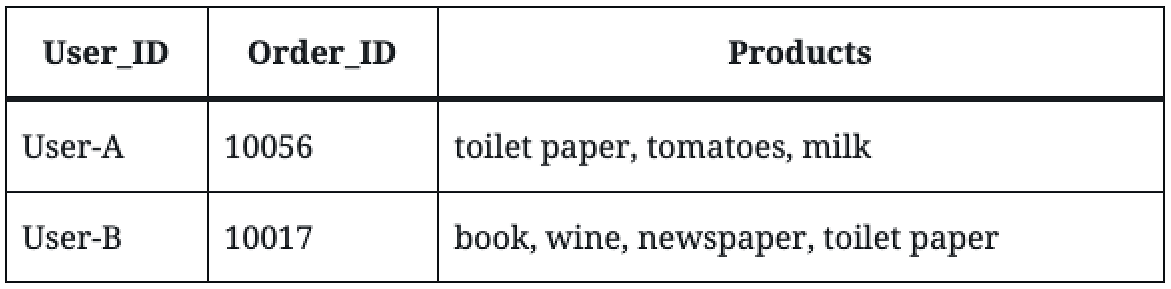
```
GET /user-information HTTP/1.1
Host: website.thm
Cookie: user_id=5
User-Agent: Mozilla/5.0 (Ubuntu;Linux) Firefox/94.0

Hello Martin!
```

Problem

Assume , we have an e-commerce website that has products and users who can give orders to buy products on the page. Let's say we have 2 users; **user-A** and **user-B**. These two users have the same level of authorization access on an e-commerce website. The orders that have given by our users are listed on a page called **order_details.html** according to the **order_id** on e-commerce's website. Naturally, when users want to see their **own order details** they're browsing **order_details.html** page. The following database design is representing the relation between users, orders, and products.



This is the database structure.



User_ID	Order_ID	Products
User-A	10056	toilet paper, tomatoes, milk
User-B	10017	book, wine, newspaper, toilet paper

Table 2. Sample Order Details table for 2 users.

Here there are 2 users A and B with there unique ID.The products they are buying aren't same . But if we look the request for them they are :

USER-A

http://vulnerableecommerce.local/order_details?order_id=10056

USER-B

http://vulnerableecommerce.local/order_details?order_id=10017

Attackers can easily change the id in burpsuit intercept and get the file.

possible vulnerable URL to perform IDOR in below:

- http://aysebilge.com/somepage?invoice=001
- http://aysebilge.com/changepassword?user=abg
- http://aysebilge.com/downloadFile?fileName=1.txt
- http://aysebilge.com/accessPage?item=i-14

References

1. Access Control, https://portswigger.net/web-security/access-control
2. Testing for Insecure Direct Object References, https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/04-Testing_for_Insecure_Direct_Object_References
3. IDOR Tutorials Hands-on OWASP Top 10 Training, https://thehackerish.com/idor-tutorial-hands-on-owasp-top-10-training/
4. How to find IDOR, https://www.bugcrowd.com/blog/how-to-find-idor-insecure-direct-object-reference-vulnerabilities-for-large-bounty-rewards/
5. IDOR, https://portswigger.net/web-security/access-control/idor
6. Web Hacking 101, Peter Yaworski, https://leanpub.com/web-hacking-101
7. AuthMatrix, https://github.com/SecurityInnovation/AuthMatrix
8. AutoChrome, https://github.com/nccgroup/autochrome
9. OWASP Cheat Sheet, https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html