🏪🔢 NahamStore: IDOR

Introduction

IDOR (Insecure Direct Object Reference) is a vulnerability that occurs when an application exposes direct references to internal objects—such as numeric identifiers, filenames, account numbers, or any internal resource—without implementing adequate *authorization controls*. This allows an attacker to manipulate these identifiers in **requests to access information** or **perform actions on resources** that do not belong to them.

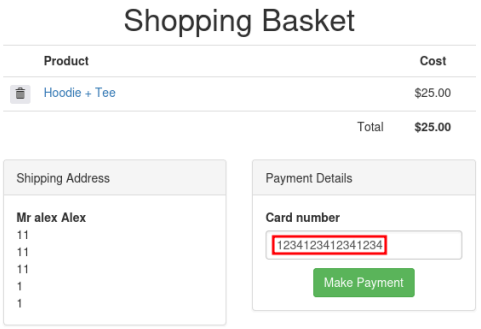
In this section we will find **IDORs** on **nahamstore.**

🕵️🔢 Looking for XSS

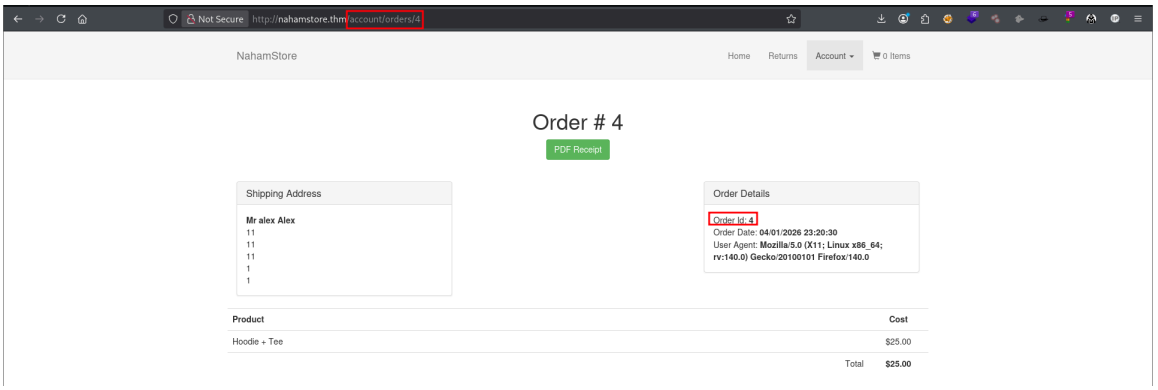
🔢 First IDOR

To find our first IDOR, we'll need to **be logged in**; that will be our context. Once logged in, we'll complete the process of ordering a product from the store. The steps, some of which we've already seen, are as follows:

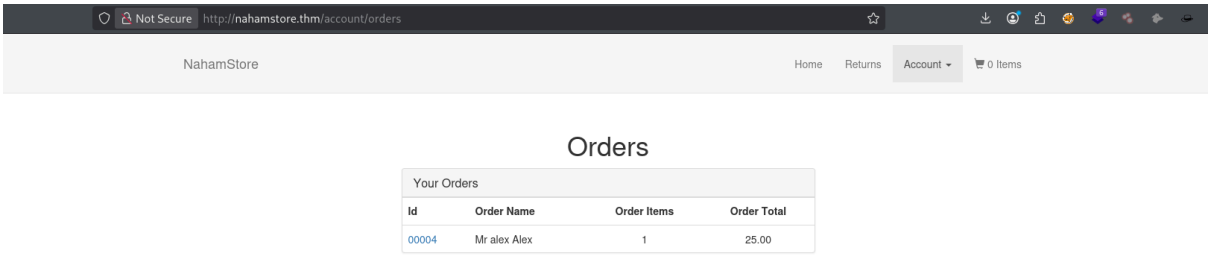
1) Click on a product (either the Sticker Pack or the Hoodie + Tee).
2) Add that product to your basket by clicking "Add To Basket."
3) Go to /basket and select an address. If you don't have one, add one by selecting "Add Another Address."
4) Finally, complete the payment by adding a card number.



Once the payment is completed, a form will be generated where we see some sections that will be very interesting for the search for possible IDORs.



In the URL we can see ***"/account/orders/4"*** as a potential IDOR point and also in that section ***"Order Id: 4"***. First, in the URL we're going to change the order number from ***"/account/orders/4" to "/account/orders/3"*** to see if we have access to another user's order data within the system:

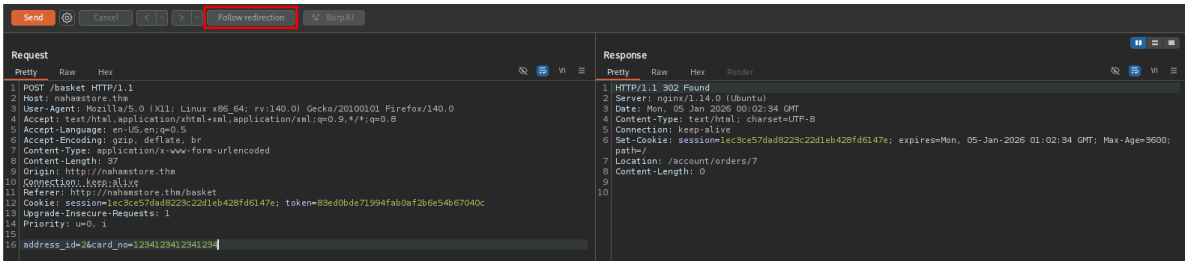


However, when we do this, we see that it does not redirect to the ***"/account/orders"*** *section* and only allows us to view the orders we have created. Therefore, the IDOR by URL is ruled out.

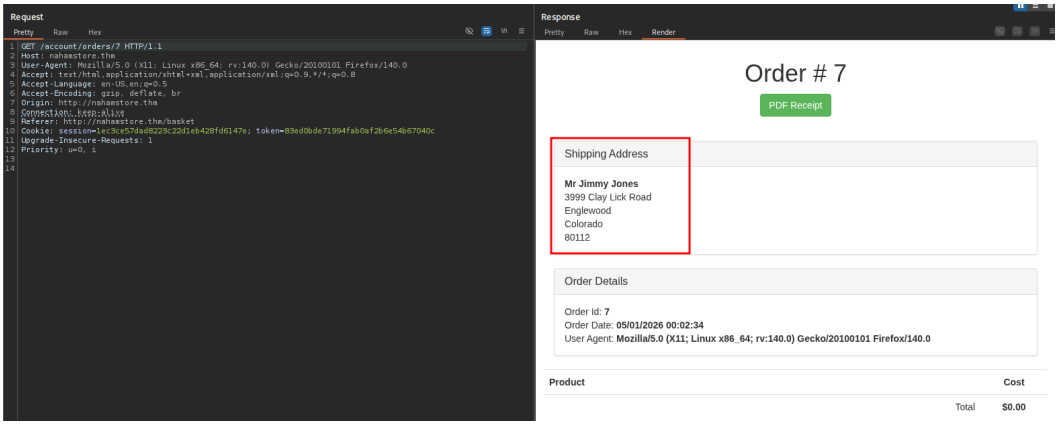
What we will do is focus on that ***"Order Id: 4"*** section that was generated when completing the "Payment". For this we will use ***Burp Suite*** to intercept the request while clicking on "***ID 00004"*** of our order to see if we can identify any interesting parameters.

As we can see, we haven't intercepted the ***"Order Id: 4"*** section. So, what we'll do is *repeat the purchase process and* ***intercept the request at the time of payment****.* This way, we'll see if we have access to the "Order Id: 4" section.

The result has been quite interesting! We've managed to visualize the ***"address_id="***, which could be a very interesting vector. What we'll do now is *send the request to* ***Repeater*** and see if we can view the data related to the ***"address_id="*** of other users.
By doing this, the site will redirect us; what we have to do is click on the repeater to **follow redirection**



And this will be the result:



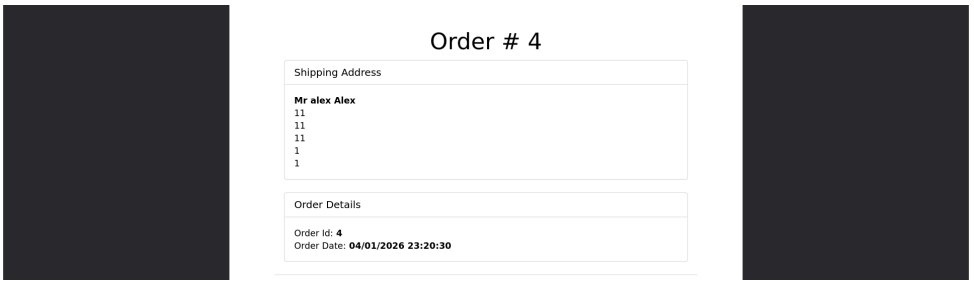
We have generated an order that allows us **to view the private data of another user of the website.**

We have completed our first **IDOR**

🔢 Second IDOR

The second IDOR can be found in the same section /account/orders/4, but this time focusing on **"PDF Receipt"**.

Upon clicking, a PDF will open allowing us to view the order and the data we have entered there.



The interesting part comes when we perform this process by intercepting the request with Burp Suite.

```
POST /pdf-generator HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: http://nahamstore.thm
Connection: keep-alive
Referer: http://nahamstore.thm/account/orders/4
Cookie: token=83ed0bde71994fab0af2b6e54b67040c; session=bcf1542a24224e254b24073200ee37da
Upgrade-Insecure-Requests: 1
Priority: u=0, i

what=order&id=4
```

As we can see in the intercepted request, we have two parameters: **"what=order"** and, even more interestingly, ***"id=4"*** which identifies the order number, in this case, order #4. This means that if we modify the "id" parameter, *we could see the orders that other users have placed on this website*. And that's what we're going to do:

```
Priority: u=0, i

what=order&id=3
```

However, the PDF that is generated is this one

Order does not belong to this user_id

As we can see, the operation is not going as expected; however, the generated PDF is giving us another parameter that we must consider, which is the ***"user_id"*** parameter.

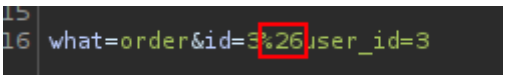
```
Upgrade-Insecure-Requests: 1
Priority: u=0, i

what=order&id=3&user_id=3
```

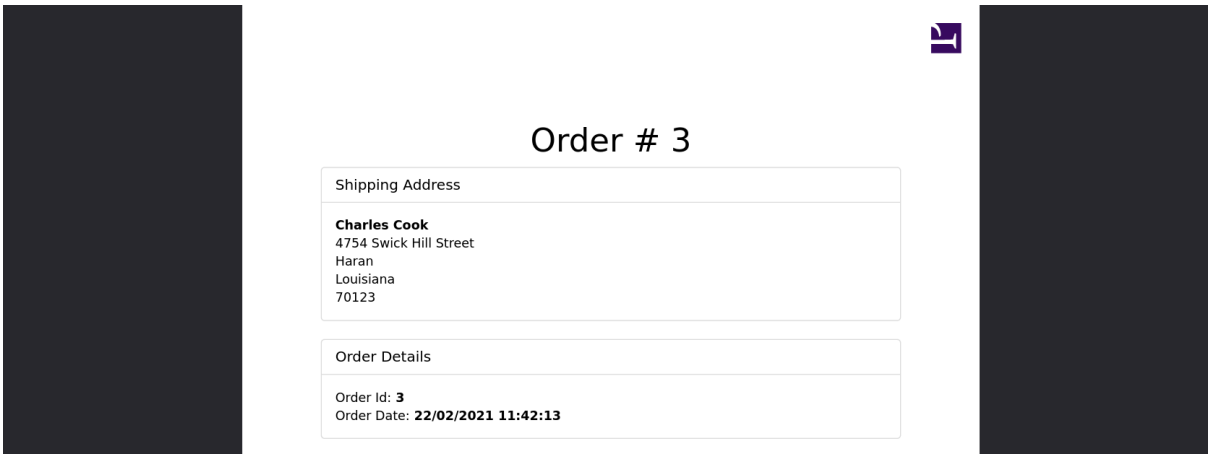
But we get the same result again.

Order does not belong to this user_id

So what I decided to do was URL Encode the "&" that separates the id and user_id parameters (by pressing ctrl+u on it)



And this was the result:



We have managed to access the PDF of another user's order.

This is because by injecting ***user_id*** within the id parameter using URL encoding ***(%26),*** the backend incorrectly parses the parameters (HTTP Parameter Pollution) and relies on manipulable data, allowing access to an order that does not belong to the authenticated user, resulting in an *IDOR*.

Conclusion

This concludes our discussion of IDOR vulnerabilities. The next vulnerability will be Local File Inclusion.