



## Everything You Need to Know About IDOR (Insecure Direct Object References)

📅 April 17, 2020    👤 ayse

I've started a new journey in this quarantine times and decided to investigate OWASP Top 10 and write about it as much in detail as possible. When I say everything, I mean it; every-little-detail! IDOR is an underrated topic, in my opinion. There are many posts about it but none of them is comprehensive enough (again my idea). People who wrote the posts they know their job it's so obvious. But all of them have written to people who already know many things about cybersecurity and reading to brush up their knowledge. So buckle up!!!

I'll examine the topic in three main topics: Problem Definition, Demo and Mitigation.

Before talking about IDOR I believe I have to explain access control a little bit, first. Access control or authorization is a decision mechanism for the requests in an application as a result of attempted actions or attempted resource access.

User roles define their capabilities in an application or system. So naturally, some roles have more privileges than others. Access control is directly related to **authorization** schema. It can be explained better under 3 subtitle:

1. *Vertical Access Control*: Vertical Access Control aims to control the restrictions to access functions according to the user roles. For example, a moderator is a role that has possibly more access right to contents in a system than a regular user. On the other hand, admin is a role has the highest privilege in a system compared to other roles (Table 1).
2. *Horizontal Access Control*: Horizontal Access Control aims to control the restrictions to access resources by users who have the same capability level. As you can understand users have the same role type however their contents should stay specific to the user itself. For example, a user who is registered to the system can display own bill report from the system but the same user should not be allowed to display other users' bill reports.
3. *Context-dependent Access Control*: Context-dependent Access control aims to control the restrictions specified for the user itself. For example, a user buys a product on an e-

commerce website and after the product is shipped user wants to change the shipping address. This shouldn't be allowed by the system.

User	Create Post	Read Post	Update Post	Delete Post
Admin	YES	YES	YES	YES
Moderator	YES	YES	YES	NO
Registered User	NO	YES	NO	NO
Unregistered User	NO	NO	NO	NO

Table 1. User role types

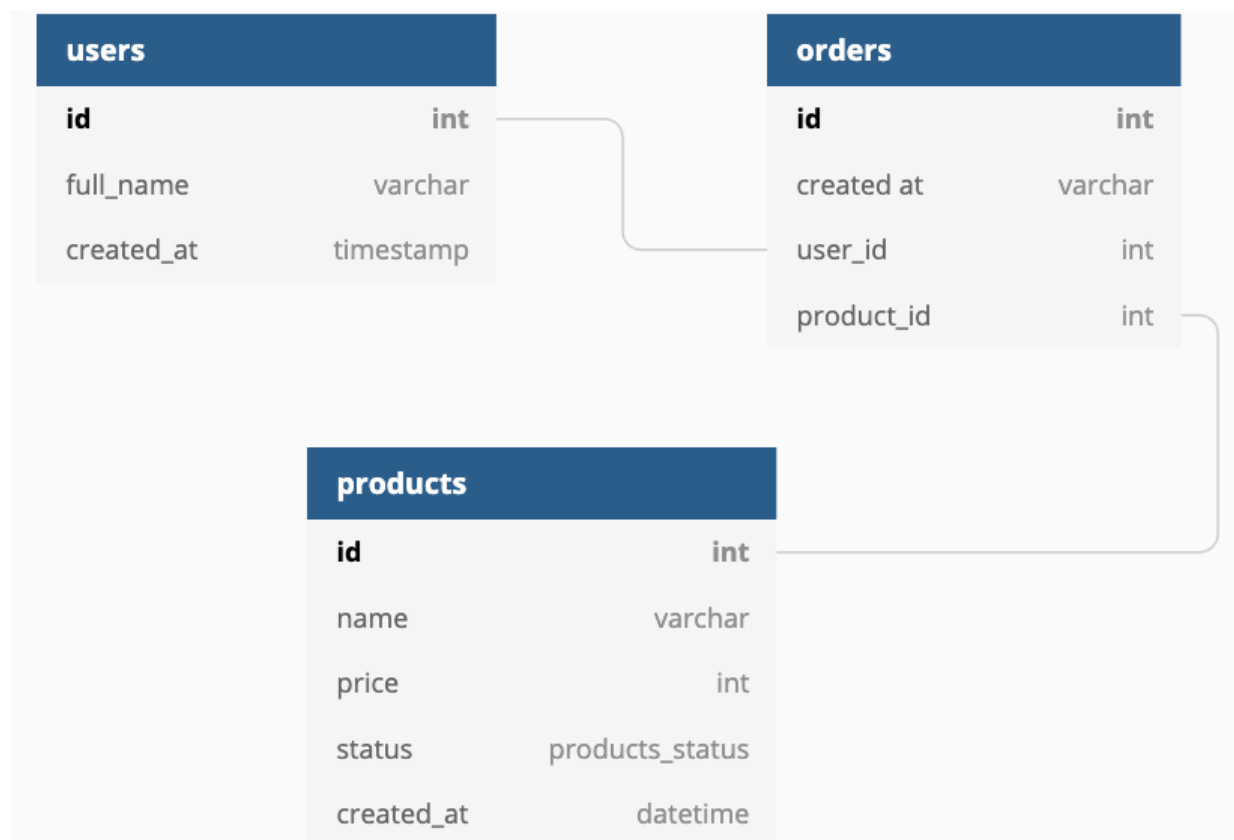
### Problem

An insecure direct object reference occurs when an attacker gains direct access by using user-supplied input to an object that has no authorization to access. Attackers can bypass the authorization mechanism to access resources in the system directly by exploiting this vulnerability [1]. IDOR is still in OWASP Top 10; however, it's located under Broken Access Control.

Every resource instance can be called as an object and often, represented with an ID. And if these IDs are easy enough to guess or an object can be used by an attacker to bypass access check somehow, we can talk about an IDOR at this point.

I'd like to elaborate more with a visualized example. In our example, 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 been 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.



Example db design to show IDOR issue

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.

User-A has an order whose ID is 10056 and includes products “toilet paper, tomatoes, and milk”. On the other hand, User-B has an order ID as 10017 and his order has products which are book, wine, newspaper, toilet paper (toilet paper has become vital in quarantine times 😊).

```
1. from flask import request
2.
3. @app.route('/order_details', methods=['GET'])
4. def order_details():
5.     order_id = request.args.get('order_id')
6.     # IDOR vulnerability in here due to lack of ownership control
7.     # of taken order_id object !
8.     order = Orders.query.filter_by(id=order_id).first_or_404()
9.     return render_template('order_details.html', order=order)
```

The above function will be executed when a User-A and/or User-B visit following URL. Regular URLs to display order details are shown below. But as you can imagine attackers aren't regular and line 8 in the function is vulnerable against IDOR. Therefore, if User-A interrupts the order and changes the parameter in order\_id to **10017** instead of **10056** ze will proceed to User-B's order details with no further authorization control.

```
USER-A
http://vulnerableecommerce.local/order_details?order_id=10056

USER-B
http://vulnerableecommerce.local/order_details?order_id=10017
```

Even if the simplest IDOR approach can be considered as guessing another user's ID to attack horizontally or vertically; IDOR does not only consist of from guessing an ID. To test and IDOR there is certain steps you can follow:

1. Log in to the same application with 2 different user roles. If you'd like to you can use different to perform this action or you can log in with user-1 to regular window and with

- user-2 to incognito window.
2. List all endpoints in the application.
  3. Try perform on endpoints for cross user roles. You can use AuthMatrix in BurpSuite.
  4. If you can read, update, create or delete an endpoint you're not authorized then there is an IDOR.

To perform all the steps above easily Burp Suite's **AuthMatrix**, request check from **HTTP History** tab can be useful. Besides, you can observe changed parameters by using **send to comparer**. Moreover, you have to be aware that sometimes parameters can be encoded so you can try to decode first.

### AuthMatrix

You can install AuthMatrix in Burp Suite > Extender > BApp Store. After that AuthMatrix will be added as another tab on your BurpSuite. Then in AuthMatrix tab, you can create roles and enough types of user which will match with all role types.

- After you create each role and users you need to generate session tokens for each user and enter them into the relevant columns within the User Table. To perform this action you can use Repeater Tab.
- In Burp Suite you can send requests to AuthMatrix by right-clicking the relevant request you'll see "Send to AuthMatrix".
- In the Request Table of AuthMatrix, select the checkboxes for all roles that are authorized to make each HTTP request.
- You can also customize Response Regex based on the expected response behavior of the request to determine if the action has succeeded.
  - Common regexes include HTTP Response headers, success messages within the body, or other variations within the body of the page.
  - NOTE: Requests can be configured to use a Failure Regex instead through the right-click menu (i.e. Authenticated users should never receive an HTTP 303)

- After you completed all customization you can click to the Run button at the bottom to run all requests or you can select certain requests and then you can run.
- You can observe the results as visualized with colors on the table.
  - Green is a safe color indicates no vulnerability detected
  - Red is an alerting color indicates may contain a vulnerability
  - Blue is a mediocre color that indicates a false positive result may have occurred.

The below image is an example of the illustration of an AuthMatrix configuration.

The screenshot displays the Burp Suite interface with the AuthMatrix tab active. It shows a configuration table for users and roles, and a request log table with color-coded results.

User Name	Cookies	Employees	Managers	HR
Employee1	session_id=qkDmc7eX+rmeRq2OMLY9EfdcjFpMpUNeYp0cTqNjCQ=	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manager1	session_id=f07gyTK81aFj0kH67NgdIB8q9JSXHKID5kEhrGNB5A=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
HR1	session_id=ynaGm82BjW+lqPBTZhp4kNnXQMAQK0CHNTROEE7D5I=	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
TestUser	session_id=MZiHEL+nyGUNN3K2db9RW4fmxErlc+pPLmuRopMdkZo=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Anonymous	session_id=	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

ID	Request Name	Response Regexp	Employees	Managers	HR	Employee1 (only)	Manager1 (only)	HR1 (only)	TestUser (only)	Anonymous (only)
0	GET /HumanResources/admin/users/create	^HTTP/1.1 200 OK								
1	GET /HumanResources/admin/users/list	^HTTP/1.1 200 OK								
2	GET /HumanResources/admin/users/reports	^HTTP/1.1 200 OK								
3	GET /HumanResources/manage/reviews	^HTTP/1.1 200 OK								
4	GET /HumanResources/manage/timesheets	^HTTP/1.1 200 OK								
5	GET /HumanResources/manage/users	^HTTP/1.1 200 OK								
6	GET /HumanResources/paymentInfo	^HTTP/1.1 200 OK								
7	GET /HumanResources/timeOff	^HTTP/1.1 200 OK								

The interface also includes a 'Request' tab showing the raw HTTP request and response, and a 'Run' button at the bottom.

AuthMatrix is a useful tool but it has some limitations because of the nature of a certain type of IDORs. Using AuthMatrix to test authorization on read and update operations works well however the operation delete/destroy the object is problematic.

AuthMatrix prone to result as false positive especially on delete operation. Let's go back to our order detail example and assume that we have a delete function on orders. Each user on the

system can only delete their own orders. When AuthMatrix sends a delete request with user-A's cookie, the object will be destroyed as expected. AuthMatrix will see HTTP 200 response. When AuthMatrix tries to send the same delete request with user-B's cookie, the application will return HTTP 404 response because the object doesn't exist! since It's already deleted by user-A. But that doesn't mean there is not an IDOR. 😊

### Demo

In this section, how to find and exploit an IDOR vulnerability will be illustrated.

**PS:** The illustrations are from the IDOR lab solution of Portswigger Academy.

The main motivation of the IDOR lab is stated as below:

**This lab stores user chat logs directly on the server's file system and retrieves them using static URLs.**





**Solve the lab by finding the password for the user `carlos` , and logging into their account.**







2)

[Home](#) | [Account login](#) | [Live chat](#)

# WE LIKE TO SHOP

			
Hitch A Lift ★☆☆☆☆ \$62.10	High-End Gift Wrapping ★★★★★ \$16.48	The Alternative Christmas Tree ★☆☆☆☆ \$89.32	Eggtastic, Fun, Food Eggcessories ★★★★★ \$73.90
<a href="#">View details</a>	<a href="#">View details</a>	<a href="#">View details</a>	<a href="#">View details</a>

			
Pet Experience Days ★★★★★ \$40.05	Paint a rainbow ★★★★★ \$60.40	Conversation Controlling Lemon ★★★★★ \$40.00	Vintage Neck Defender ★★★★★ \$60.00

There is a couple of entrance points we can try; it's either "View Details" to see items and top-right corner items which are Home, Account Login and Live Chat. However lab description specifically leads us to Live Chat since it mentions *stored chat logs*.

## Live chat

CONNECTED: -- Now chatting with Hal Pline --

You: dfgdfgdgdfg

Hal Pline: Sometimes I wonder how you're still married

You: me too

Hal Pline: Are you sure you want to know the answer to that?

You: hmm let me think about it

Hal Pline: I'll look that up when my nail polish has dried.

Your message:

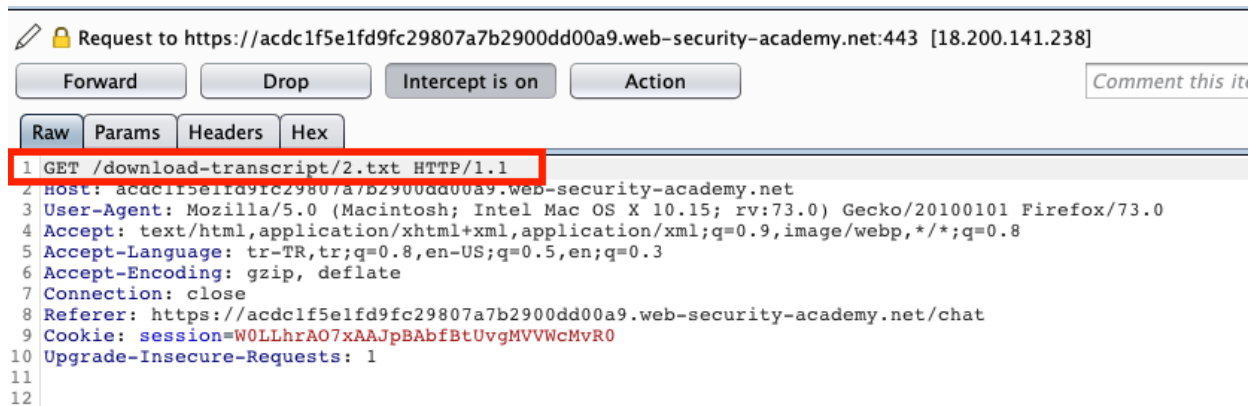
1)

2)

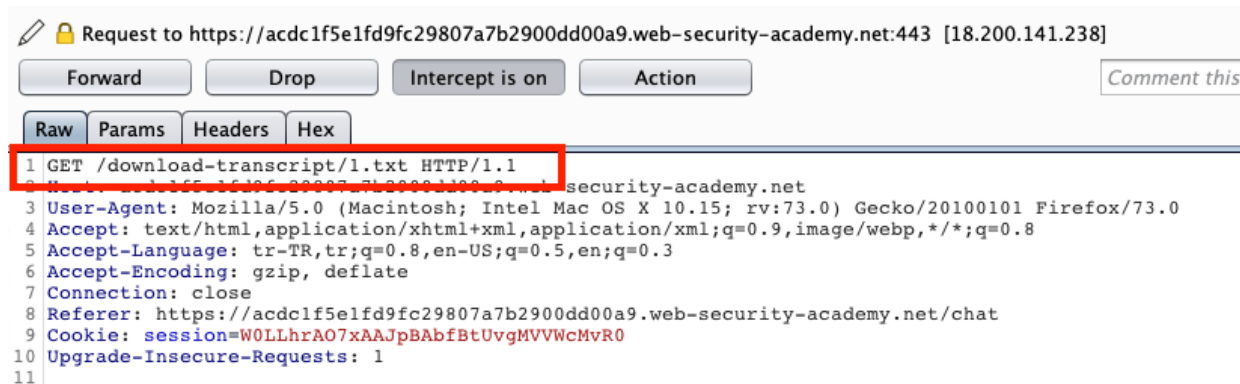
Send

View transcript

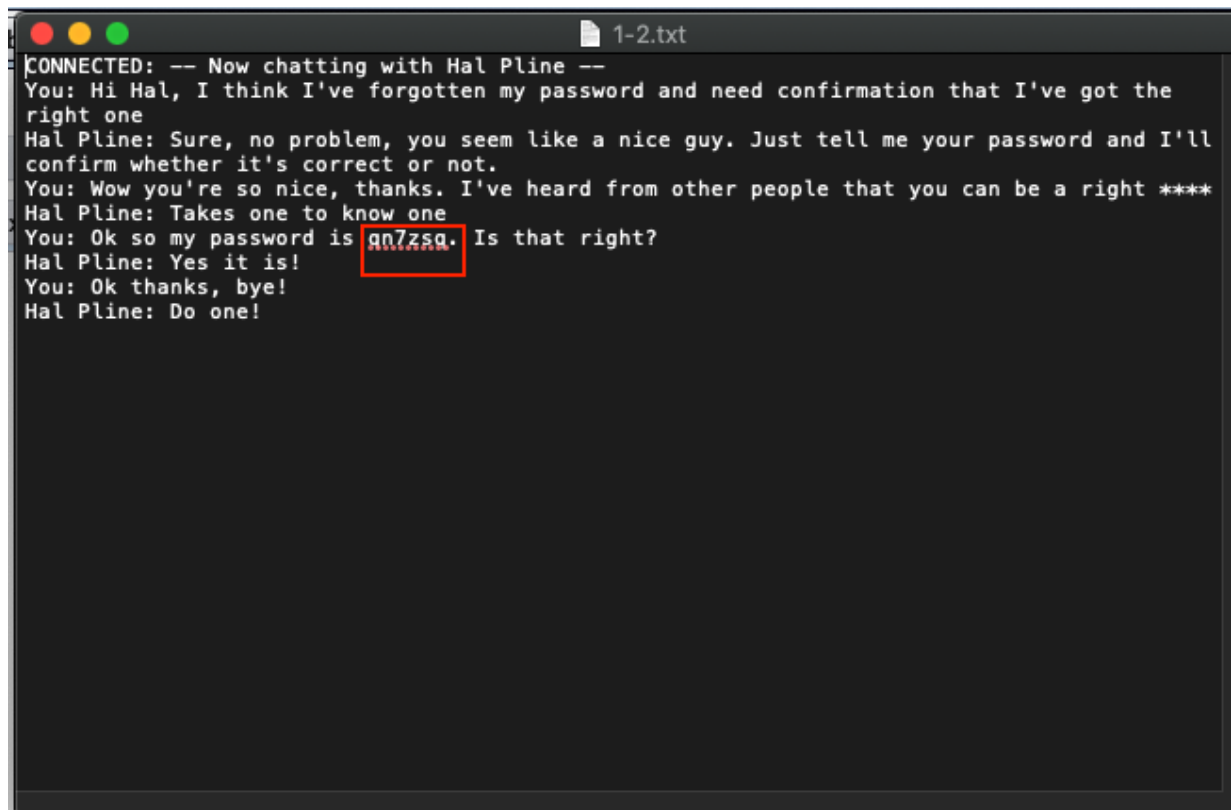
In Live chat, you're chatting with Hal. As long as you send a text to Hal, she is texting you back. Hal is confused and trying to distract you but don't mind her. 😊 Now when we click to View transcript button, we're getting texting transcript we had with Hal. But right at this point, if you're not distracted there is something you can pay attention to. Wait wait! We'll come in there.



The GET request which is the area inside the red rectangular is a request for download a file. According to the request I can download 2.txt which consists of my texting transactions. 2.txt is a parameter changes with each request and incrementing by one with every new request. So, the value of a parameter is used directly to retrieve a file. The beauty of this fact is I can catch this request and change the parameter using Burp Suit. At this point, Access Control mechanism shouldn't let me download an altered file. But if it let me this is an exploitation of IDOR vulnerability.



So I altered the GET request from `/download-transcript/2.txt` to `/download-transcript/1.txt`. Then I forward it and Access Control Mechanism let me execute this action. Voilá!!



```
1-2.txt
CONNECTED: -- Now chatting with Hal Pline --
You: Hi Hal, I think I've forgotten my password and need confirmation that I've got the
right one
Hal Pline: Sure, no problem, you seem like a nice guy. Just tell me your password and I'll
confirm whether it's correct or not.
You: Wow you're so nice, thanks. I've heard from other people that you can be a right ****
Hal Pline: Takes one to know one
You: Ok so my password is qn7zsq. Is that right?
Hal Pline: Yes it is!
You: Ok thanks, bye!
Hal Pline: Do one!
```

By using this password: `qn7zsq` and username: `carlos` (uname has given in lab definition) you can login the system and complete the lab.

Insecure direct object references

WEB SECURITY ACADEMY

LAB Not solved

Back to lab home Back to lab description >>

Home Account login Live chat

### Login

Username

carlos

Password

\*\*\*\*\*

Log in

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

[Home](#)

[Hello, carlos!](#)

[Log out](#)

[Live chat](#)

WE LIKE TO  
SHOP



yaaaay!!!



Hitch A Lift

★☆☆☆☆ \$62.10

[View details](#)



High-End Gift Wrapping

★★★★★ \$16.48

[View details](#)



The Alternative Christmas Tree

★☆☆☆☆ \$89.32

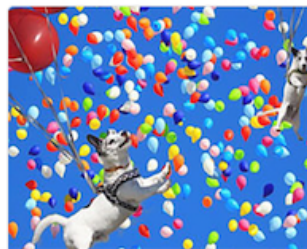
[View details](#)



Eggtastic, Fun, Food Egcessories

★★★★★ \$73.90

[View details](#)



However, I think I have to stress a point: “As long as you’re capable to change the parameter and perform the action, it doesn’t matter if the resource is a file or functionality or ability to perform an operation or retrieving a database record.” You can see 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`

## Mitigation

IDOR is a complex vulnerability to find and also to mitigate. So, I’ll try to explain the 3 approaches as mitigation of IDOR:

1. First of all, the main point of IDOR is insufficient Access Control. So, if you prepare a proper Access Control Mechanism IDOR became avoidable.
2. Second of all, IDOR occurs when the authorization check has forgotten to reach an object in the system. It is critical if the reached object is sensitive like displaying an invoice belongs to users in the system. So that severity of the IDOR can be changed depends on the sensitivity of the information. So, I advise using randomly generated IDs or UUIDs instead of auto-incremental IDs. The attacker has to find valid random ID values that belong to another user. In this scenario, even if the application prone to IDOR it’s not going to be exploitable.
3. Sometimes it can be very hard to apply first and/or second steps on huge -usually legacy- applications. In this kind of situation, there is a 3rd approach you may follow. Even if you use auto-incremented object IDs you can apply a hash function with salt and put in a hash map like key-value pair. Then you’ll store the key-value map in the Session. Instead of exposing auto-increment IDs to the user, you can use hash values of corresponding IDs. When you get the value back from the user, you can find an actual ID value by looking up the key-value map in the Session. So that means, even if the attacker



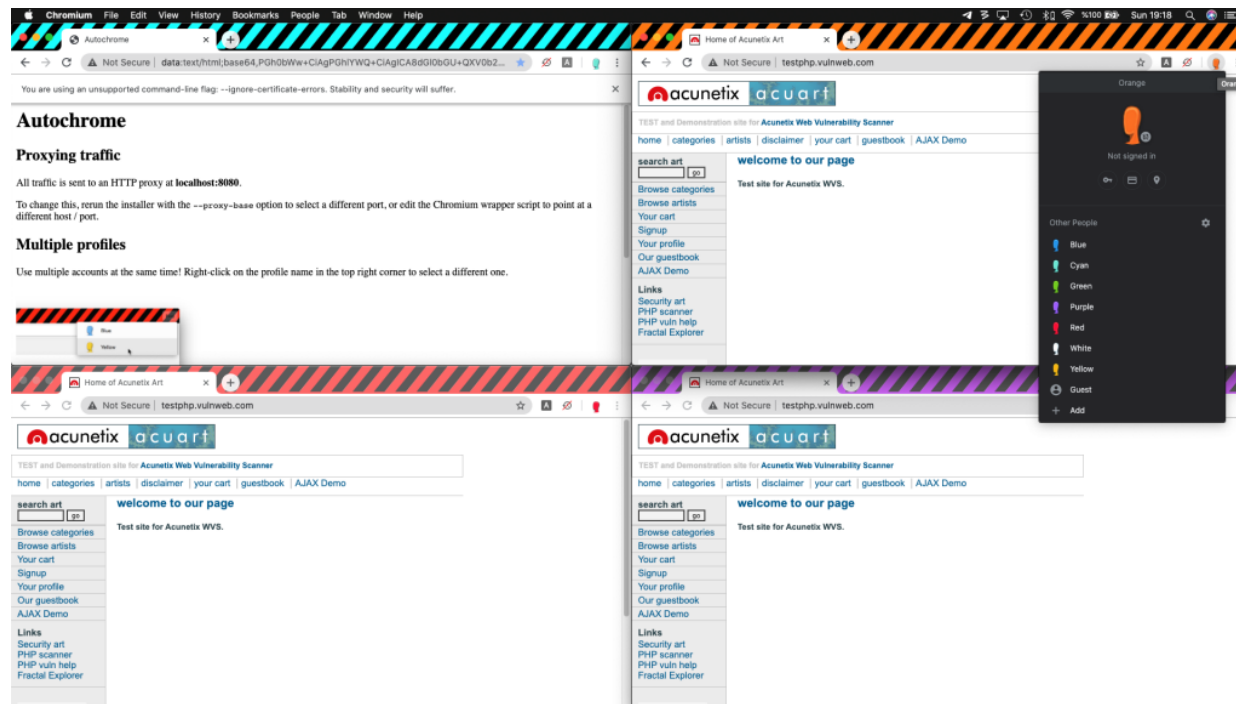
spoof the generated value it's not going to exist on the map. Basically that means IDOR is not going to be exploitable anymore.

```
User1: 434d0bcc0812f382480132e29d42abd02efa7344
User2: 9981b2af821d3dab6a433076408d04e313e83738
```

## Testing with “AutoChrome”

If you have come this far I have another tricky tool for you. We talked about testing IDOR on two different users so far. But sometimes you need to do the same test with many users at once however you're limited with your browser in this sense. And even if you use different browsers for each user and even if it reaches enough number of users, it is messy and confusing for a clean test. So, you can use **AutoChrome** for testing with many users. It is a specialized **Chromium** version and let's 7 consecutive users at the same time each of them has different colors on the bar as you see 4 of them in the below image. All traffic is sent to an HTTP proxy at **localhost:8080** in default. To change this, run the installer with the **-proxy-base** option to select a different port, or edit the Chromium wrapper script to point at a different host/port. You can install it as a Chrome extension as well.





## 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](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>

[authmatrix](#)[autochrome](#)[bug-bounty](#)[bugbounty](#)[burp](#)[idor](#)[pentest](#)[websec](#)

0 Comments

aysebilge

 [Disqus' Privacy Policy](#)

 1 Login ▾

 Recommend

 Tweet

 Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 



Name

Be the first to comment.

 [Subscribe](#)

 [Add Disqus to your site](#)

 [Do Not Sell My Data](#)

**DISQUS**

Search...



**AYŞE BILGE GÜNDÜZ**



## RECENT POSTS

Everything You Need to Know About IDOR (Insecure Direct Object References)

---

## TAG CLOUD

authmatrix

autochrome

bug-bounty

bugbounty

burp

idor

pentest

websec

## CATEGORIES

 Application Security

---

sparkling Theme by Colorlib Powered by WordPress