

## Actividad | 3 | Cross Site Scripting (XSS).

### Auditoría Informática.

Ingeniería en Desarrollo de  
Software.



academiaglobal

TUTOR: Jessica Hernández Romero

ALUMNO: Carlos Ariel Nicolini

FECHA: 22/10/2025

## Índice

<b>Introducción .....</b>	<b>3</b>
<b>Descripción .....</b>	<b>4</b>
<b>Justificación .....</b>	<b>5</b>
<b>Etapas 1: .....</b>	<b>6</b>
• <b>Descripción del sitio web .....</b>	<b>6</b>
• <b>Ataque al sitio web .....</b>	<b>8</b>
<b>Etapas 2: .....</b>	<b>11</b>
• <b>Ataque al sitio .....</b>	<b>11</b>
<b>Etapas 3: .....</b>	<b>18</b>
• <b>Ataque al sitio .....</b>	<b>18</b>
<b>Conclusión.....</b>	<b>27</b>
<b>Referencias.....</b>	<b>28</b>

# Introducción

El scripting entre sitios (XSS) es un problema de seguridad web que ve a los cibercriminales ejecutar scripts maliciosos entre sitios web legítimos o confiables.

En un ataque XSS, un atacante utiliza paginas web o aplicaciones web para enviar códigos maliciosos y comprometer las interacciones de los usuarios con una aplicación vulnerable. Estos tipos de ataques generalmente ocurren como resultado de fallas comunes dentro de una aplicación web y permiten a un actor malicioso tomar la identidad del usuario, llevar a cabo cualquier acción que el usuario realiza normalmente y acceder a todos sus datos. Por ejemplo, si un usuario tiene acceso privilegiado a la aplicación de una organización, el atacante puede tomar el control total de sus datos y funcionalidad.

El script malicioso que explota una vulnerabilidad dentro de una aplicación garantiza que el navegador del usuario no pueda identificar que proviene de una fuente no confiable. Como resultado, el atacante puede acceder a cookies, token de sesión y cualesquiera otros datos sensibles que recopile el navegador, o incluso reescribir el contenido del lenguaje de marcado de hipertexto (HTML) en la página. El sitio Web o la aplicación que entrega el script al navegador de un usuario es efectivamente un vehículo para el atacante. Los objetivos populares para los ataques XSS incluyen cualquier sitio que permita comentarios de usuarios, como foros en línea y paneles de mensajes.

# Descripción

## **Contextualización:**

Una empresa de software solicita realizar varias pruebas de seguridad en páginas web que no cuentan con los candados de seguridad.

Para esta tercera etapa se solicita realizar una prueba de vulnerabilidad de Cross Site Scripting (XSS). En ella se debe obtener las credenciales que se ingresen para iniciar sesión. Después, desde Burp Suite, modificar la información para comprobar si se puede iniciar sesión o no.

## **Actividad:**

Utilizando el sitio web que se subió a internet en la primera actividad, y el programa utilizado en la Actividad 2, trabajar con la vulnerabilidad Cross Site Scripting (XSS). Así, con la ayuda de Burp Suite, captar las credenciales que se ingresen cuando se inicie sesión, y comprobar si se puede modificar.

Para este trabajo no pude utilizar el sitio que se utilizó en la primera actividad, en su lugar en la máquina virtual que utilice para la segunda actividad, instale un ambiente local con xampp y configure un DVWA para pruebas que está publicada en Github, la cual sirve como plataforma de aprendizaje y realizar este tipo de pruebas.

# Justificación

En esta actividad realizaremos un ataque que nos permitirá entender de primera mano y de una manera muy practica como funciona el proceso de autenticación de un sitio web y como viajan los datos cuando iniciamos sesión en este caso en una página web. Con ayuda de la herramienta Burp Suite y su opción de interceptor, nos podemos dar cuenta a simple vista que ocurre con la información ingresada en el proceso de inicio de sesión. Además, se nos muestra y enseña que pasa cuando se modifican esos datos, lo que nos da una valiosa prueba de lo importante que es tener una buena seguridad en los sistemas y paginas web. Este tipo de pruebas es de muchísima ayuda para detectar posibles fallas o vulnerabilidades en sistemas o páginas web.

Me deja sorprendido de que manera tan sencilla se pueden capturar los datos de una persona y en esta materia e visto algunas cosas de las que se pueden hacer. Al mirar videos para documentarme e notado que me falta mucho aprender y lo sumamente importante que es tener nuestra información bien protegida y cuidada, por que con este ejercicio se deja ver lo fácil que es que un tercero las obtenga.

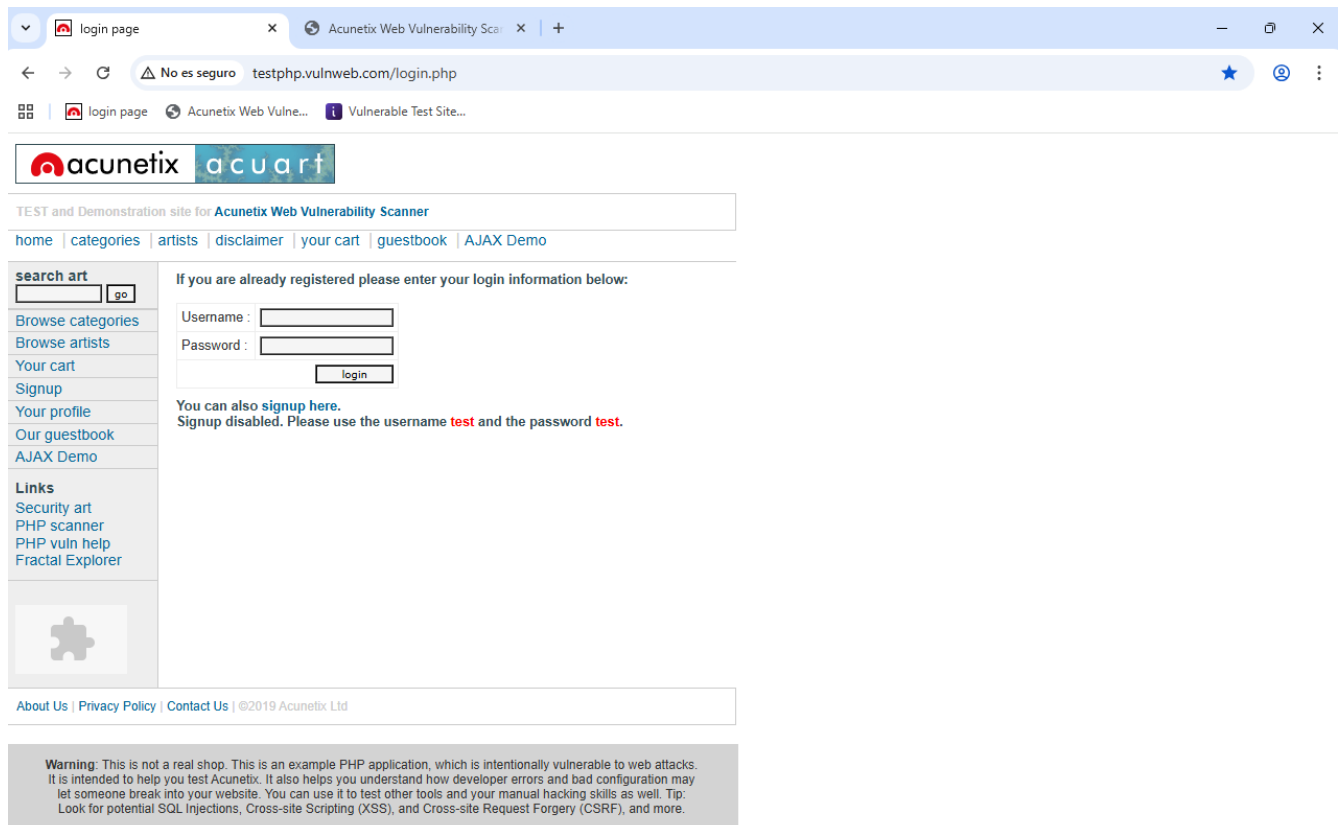
Este trabajo fue subido al siguiente enlace de GitHub

<https://github.com/CarlosNico/Auditor-a-Inform-tica>

# Etapa 1

## Descripción del sitio Web

Para este ejercicio se utilizó un sitio especial para realizar este tipo de pruebas de seguridad en una máquina virtual de Hyper-V. La página de prueba es <http://testphp.vulnweb.com>



login page x Acunetix Web Vulnerability Scanner x +

No es seguro testphp.vulnweb.com/login.php

login page Acunetix Web Vulnerability Scanner Vulnerable Test Site...

**acunetix** **acuart**

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#)

**search art**

[Browse categories](#)  
[Browse artists](#)  
[Your cart](#)  
[Signup](#)  
[Your profile](#)  
[Our guestbook](#)  
[AJAX Demo](#)

**Links**  
[Security art](#)  
[PHP scanner](#)  
[PHP vuln help](#)  
[Fractal Explorer](#)

If you are already registered please enter your login information below:

Username :   
Password :

You can also [signup here](#).  
Signup disabled. Please use the username **test** and the password **test**.

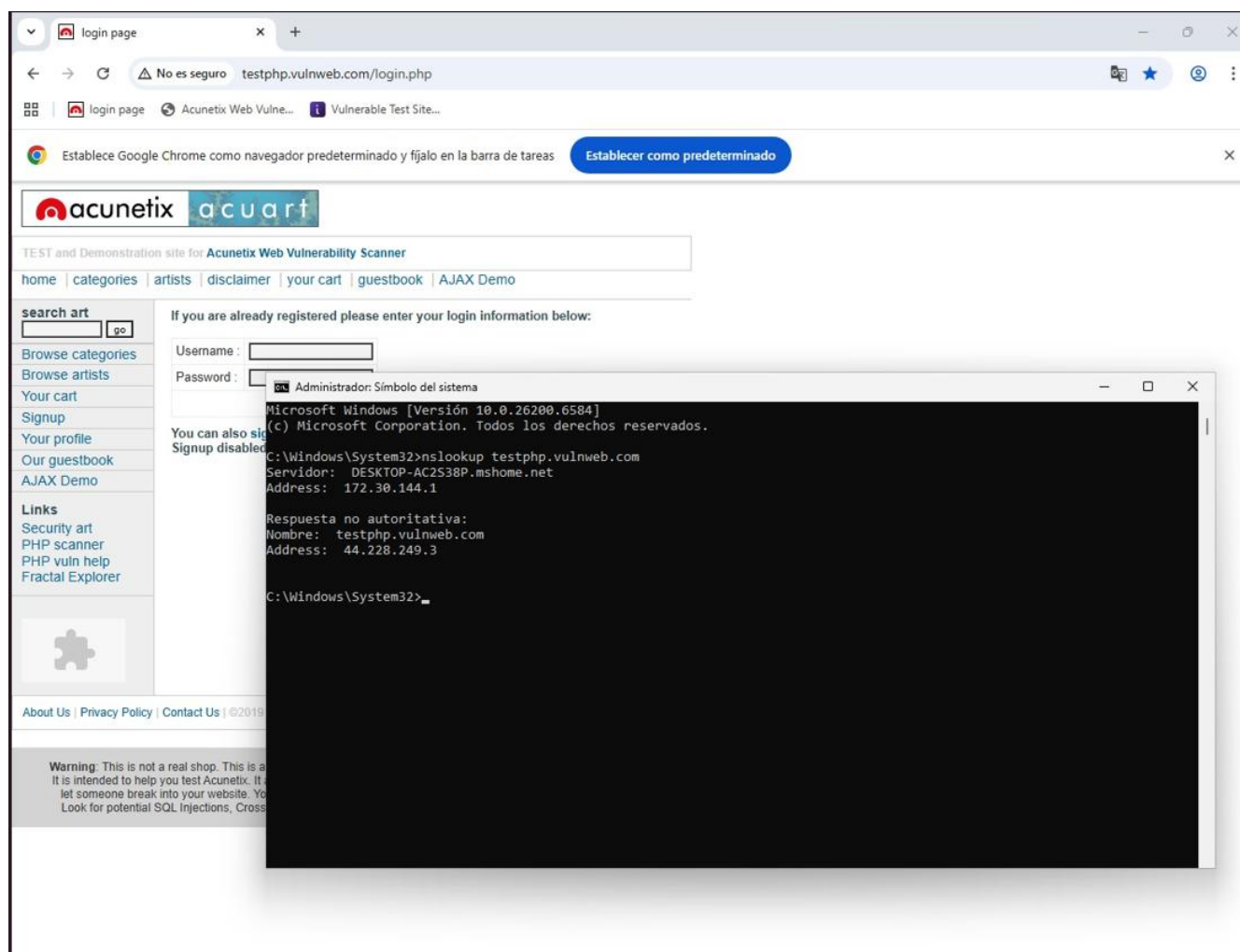
[About Us](#) | [Privacy Policy](#) | [Contact Us](#) | ©2019 Acunetix Ltd

**Warning:** This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

# Ataque al sitio

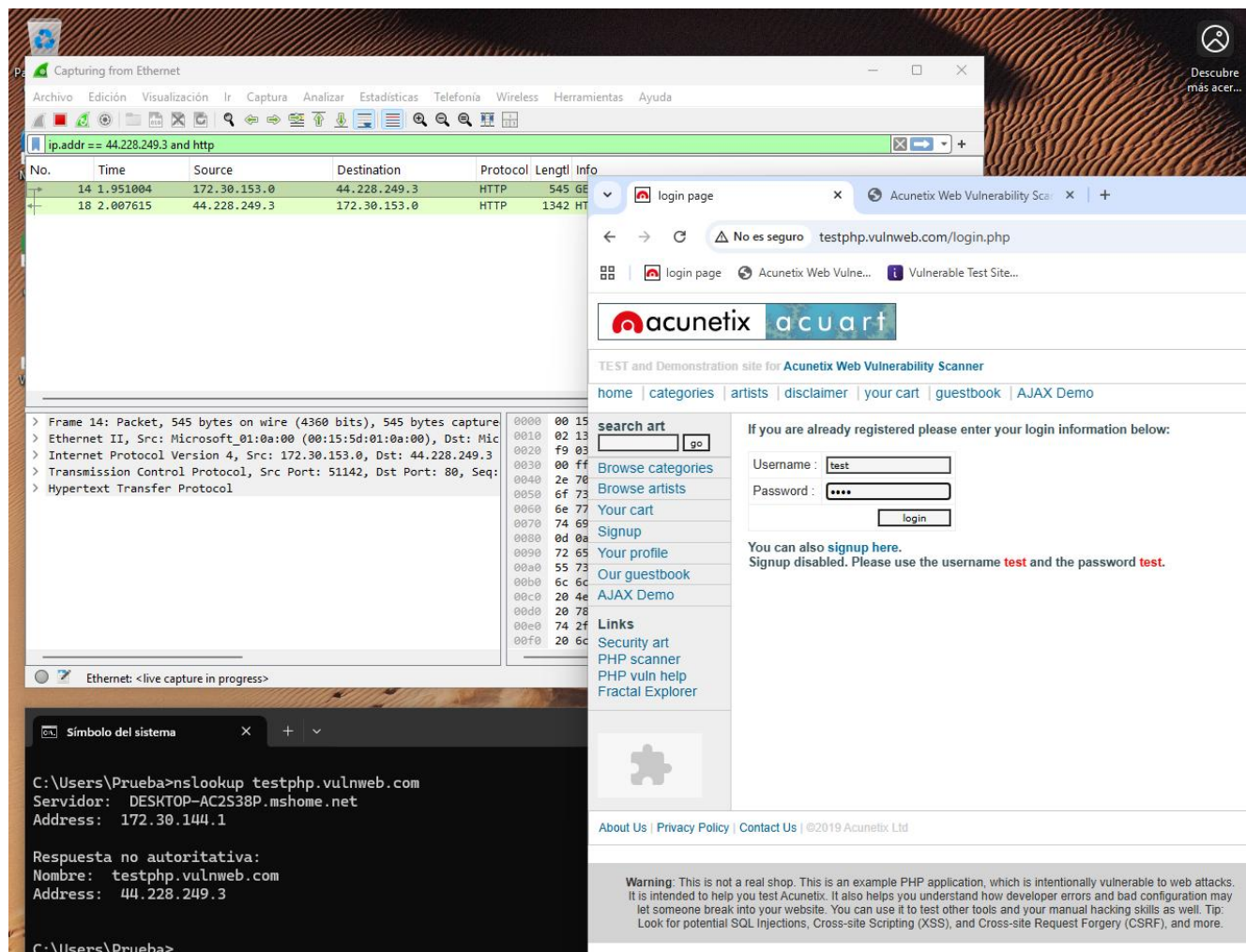
Para realizar el ataque al sitio como es solicitado, en la máquina virtual se instaló el programa Wireshark. El cual descargamos de la página <https://www.wireshark.org/> en la sección de download y se instaló.

A continuación, ejecutamos en un cmd el comando nslookup para obtener la dirección ip del sitio que seleccionamos para realizar el ataque.



La cual nos da que la ip de la página es 44.228.249.3.

A continuación, abrimos el programa WireShark y como se nos explicó ponemos el filtro con la ip del sitio de la siguiente manera `ip.addr == 44.228.249.3 and http` y lo aplicamos con el icono de la flechita del lado derecho en el filtro. Activamos la captura de paquetes en el programa y ingresamos a la pagina



Los datos de la prueba en la página son los siguientes:

Usuario: test

Contraseña: test

Los ingresamos y dejamos que el programa realice la captura de datos.



En la siguiente captura se muestra cómo se realiza de manera correcta el robo de la autenticación en la página no protegida. En la captura de WireShark se ve que nos muestra la información del usuario test y el pass test.

The image displays a network traffic capture using Wireshark and a web browser window. The Wireshark interface shows a packet capture from Ethernet. The selected packet is a POST request to testphp.vulnweb.com/userinfo... with a body containing 'uname=test' and 'pass=test'.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.30.153.0	44.228.249.3	HTTP	545	GET /log...
3	0.056892	44.228.249.3	172.30.153.0	HTTP	1342	HTTP/1.1
7	5.680529	172.30.153.0	44.228.249.3	HTTP	701	POST /us...
9	5.737923	44.228.249.3	172.30.153.0	HTTP	1509	HTTP/1.1

The browser window shows the 'user info' page for 'rehana (test)'. The user information is displayed as follows:

- Name: rehana
- Credit card number: 5500-87600-456808
- E-Mail: rehana65@gmail.com
- Phone number: 644082537
- Address: karachi

The page also shows a warning message: 'Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Look for potential SQL injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.'

También se realizó un ingreso erróneo con usuario can y contraseña can el cual no puedo iniciar sesión, pero en WireShark salió también los datos mal ingresados y fueron capturados.

The image displays a network traffic capture in Wireshark and the Acunetix web application interface.

**Wireshark Packet Capture:**

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.30.153.0	44.228.249.3	HTTP	545	GET /log...
3	0.056892	44.228.249.3	172.30.153.0	HTTP	1342	HTTP/1.1
7	5.680529	172.30.153.0	44.228.249.3	HTTP	701	POST /us...
9	5.737923	44.228.249.3	172.30.153.0	HTTP	1509	HTTP/1.1
256	295.048706	172.30.153.0	44.228.249.3	HTTP	575	GET /log...
273	295.107111	44.228.249.3	172.30.153.0	HTTP	1172	HTTP/1.1
312	296.847964	172.30.153.0	44.228.249.3	HTTP	545	GET /log...
317	296.906497	44.228.249.3	172.30.153.0	HTTP	1342	HTTP/1.1
330	302.716488	172.30.153.0	44.228.249.3	HTTP	697	POST /us...
337	302.774998	44.228.249.3	172.30.153.0	HTTP	330	HTTP/1.1
338	302.782724	172.30.153.0	44.228.249.3	HTTP	570	GET /log...
340	302.841350	44.228.249.3	172.30.153.0	HTTP	1342	HTTP/1.1

Frame 330: Packet, 697 bytes on wire (5576 bits), 697 bytes captured on interface 0 (Ethernet II), Src: Microsoft\_01:0a:00 (00:15:5d:01:0a:00), Dst: Microsoft\_01:0a:00 (00:15:5d:01:0a:00), Protocol: Hypertext Transfer Protocol, Seq: 62125, Dst Port: 80, Seq: 62125, Length: 697, Win: 0, Len: 697

HTML Form URL Encoded: application/x-www-form-urlencoded

- Form item: "uname" = "can"
- Form item: "pass" = "can"

**Acunetix Web Vulnerability Scanner Interface:**

login page x Acunetix Web Vulnerability Scanner x

testphp.vulnweb.com/login.php

Acunetix acunetix

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art go

If you are already registered please enter your login information below:

Username:

Password:

login

You can also [signup here](#).  
Signup disabled. Please use the username **test** and the password **test**.

Links

- Security art
- PHP scanner
- PHP vuln help
- Fractal Explorer

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Look for potential SQL injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

**Command Prompt:**

```
C:\Users\Prueba>nslookup testphp.vulnweb.com
Servidor: DESKTOP-AC2S38P.mshome.net
Address: 172.30.144.1

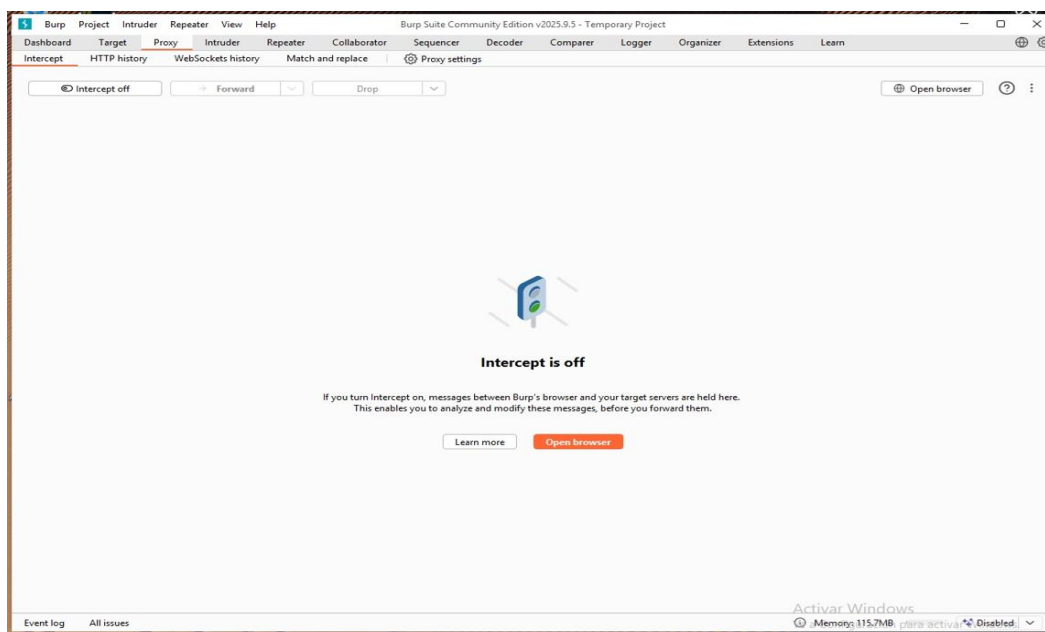
Respuesta no autoritativa:
Nombre: testphp.vulnweb.com
Address: 44.228.249.3
```

## Etapa 2

### Ataque al sitio

Para este ejercicio realizaremos un ataque a una página proporcionada en un laboratorio en la plataforma PortSwigger. Para tal motivo, ingresamos a la página donde descargaremos el software y lo instalaremos.

Con el programa ya abierto completamente, seleccionamos la pestaña de proxy, presionamos la opción Open browser, la cual nos abrirá un explorador donde pondremos el enlace del laboratorio compartido en el ejercicio (<https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-modifying-serialized-objects>).



Se nos abrirá la página del laboratorio y le damos Access the lab (debemos logearnos para poder realizarlo).

Lab: Modifying serialized objects

Products | Solutions | Research | Academy | Support

Dashboard | Learning paths | Latest topics | All content | Hall of Fame | Get started | Get certified

Web Security Academy > Insecure deserialization > Exploiting > Lab

## Lab: Modifying serialized objects

APRENTICE LAB Not solved

This lab uses a serialization-based session mechanism and is vulnerable to privilege escalation as a result. To solve the lab, edit the serialized object in the session cookie to exploit this vulnerability and gain administrative privileges. Then, delete the user `carlos`.

You can log in to your own account using the following credentials:

`wiener:peter`

ACCESS THE LAB

Solution

1. Log in using your own credentials. Notice that the post-login `GET /my-account` request contains a session cookie that appears to be URL and Base64-encoded.
2. Use Burp's Inspector panel to study the request in its decoded form: Notice that the cookie is in fact a serialized PHP object. The `__class__` attribute

My Account - PortSwigger

https://portswigger.net/users/youraccount/personaldetails

Your agentic AI partner in Burp Suite - Discover Burp AI now [Read more](#)

PortSwigger

Log out MY ACCOUNT

Products | Solutions | Research | Academy | Support

## My Account



Personal Details


Certifications

Subscriptions


Order History

**Personal Details**

 **Carlos Nicolini**   
elfoludo@gmail.com  
[Change Password](#)

**Account Address**   
No address associated with this account

**Saved Cards**

  
Add new card

Burp Suite  
 Web vulnerability scanner  
 Burp Suite Editions  
 Release Notes

Vulnerabilities  
 Cross-site scripting (XSS)  
 SQL injection  
 Cross-site request forgery  
 XML external entity injection

Customers  
 Organizations  
 Testers  
 Developers

Company  
 About  
 Careers  
 Contact  
 Legal


Insights  
 Web Security Academy  
 Blog  
 Research


PortSwigger

[Follow us](#)

Lab: Modifying serialized object x Modifying serialized objects x +

https://0a69007e03b943b1aaa77dc300740004.web-security-academy.net


WebSecurity Academy  Modifying serialized objects


LAB Not solved 

Back to lab description >>


---

Home | My account


WE LIKE TO SHOP 




Gym Suit  
★★★★★ \$3.57  
[View details](#)




Eggstastic, Fun, Food Eggcessories  
★★★★★ \$55.49  
[View details](#)





Picture Box  
★★★★★ \$19.02  
[View details](#)




What Do You Meme?  
★★★★★ \$22.58  
[View details](#)











Presionamos en My Account ingresaremos con los siguientes datos wiener:peter

Lab: Modifying serialized object x Modifying serialized objects x +

https://0a69007e03b943b1aaa77dc300740004.web-security-academy.net/login

WebSecurity Academy  Modifying serialized objects

LAB Not solved 

Back to lab description >>

---

Home | My account

## Login

Username

Password

[Log in](#)

Una vez realizado el login encendemos la opción Intercept para empezar con la captura. Como se muestra en la siguiente imagen detectamos la cookie, el cual elegimos y se muestra en el decoded base64

The screenshot shows the Burp Suite interface with the 'Intercept' tab selected. The 'Request' tab displays the raw HTTP request, and the 'Inspector' tab shows the decoded content of the selected text. The decoded content is a JSON object representing a user session.

```

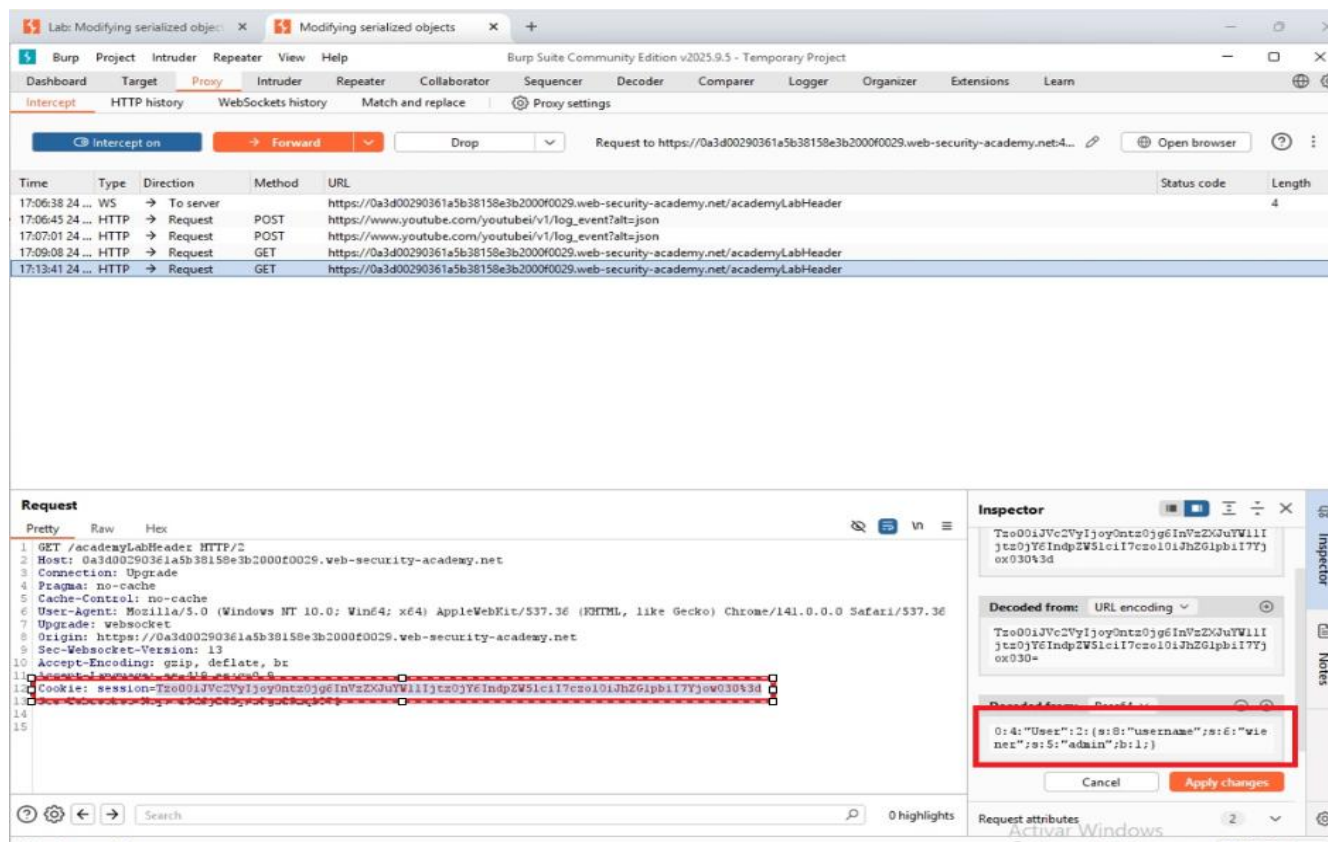
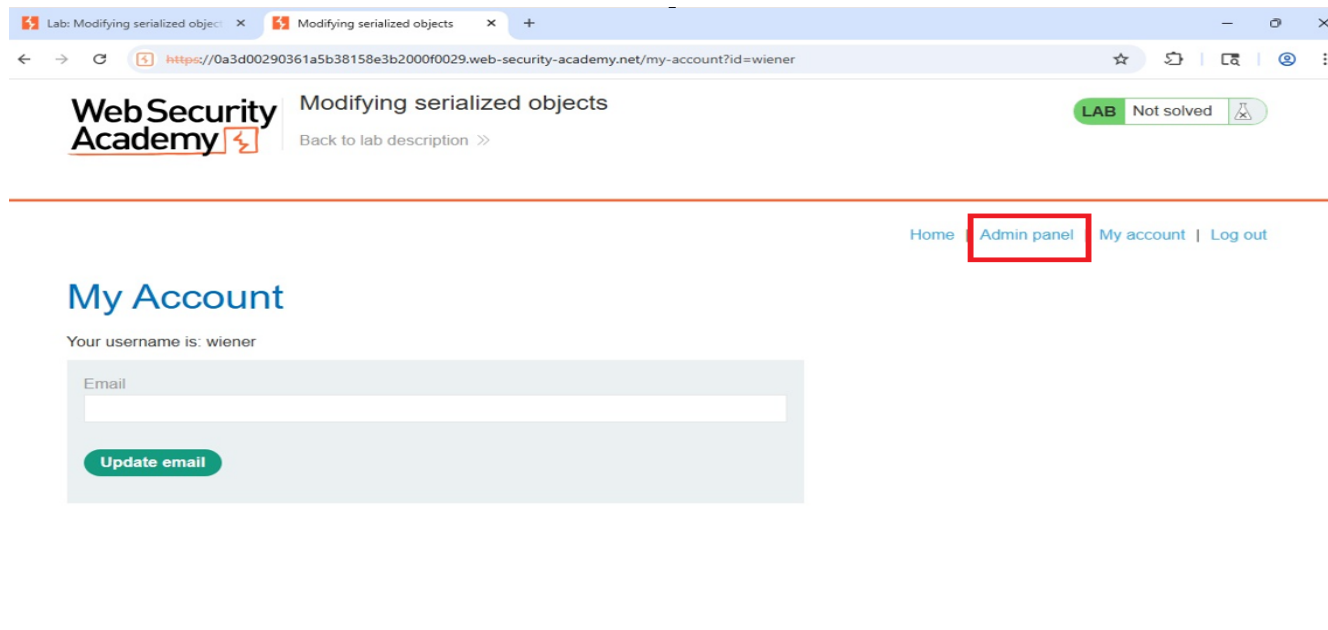
O:4:{"User":2:{s:8:"username";s:6:"wiener";s:5:"admin";b:0;}
  
```

Esta decodificación `O:4:{"User":2:{s:8:"username";s:6:"wiener";s:5:"admin";b:0;}` en palabras cortas es donde capturamos los datos del usuario Wiener (con el que ingresamos) y la parte final nos indica con un booleano 0, lo cual nos indica que no es administrador.

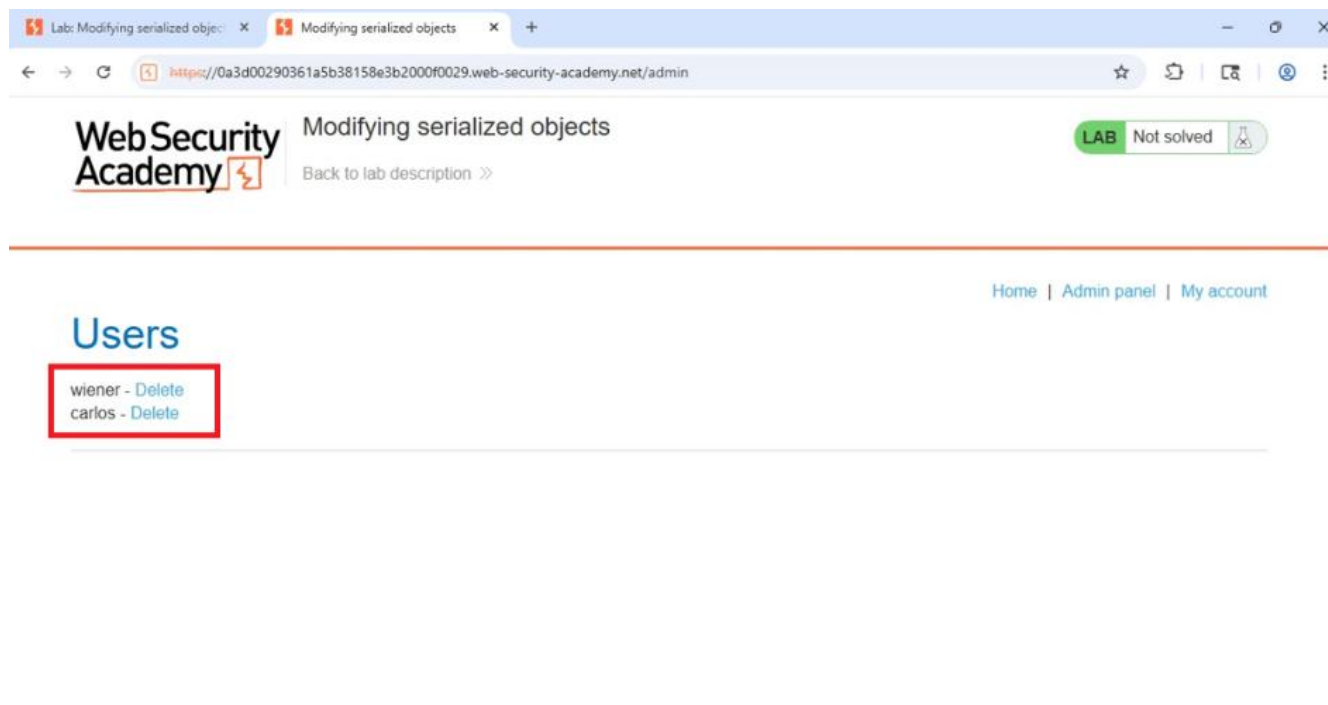
En este punto es donde realizaremos el ataque, al cambiar el 0 por el 1 (le diremos que somos administrador), aplicar el cambio y presionaremos en la opción de forward (esta al lado del botón Intercept).



Al realizar ese paso nos aparecerá en la pantalla la opción de Admin panel. A continuación, presionamos en ese botón y volvemos a la captura, donde buscaremos la cookie y realizaremos el mismo paso anterior, donde cambiaremos el booleano de 0 a 1, aplicaremos y presionaremos forward.



Al realizar ese cambio ingresaremos al panel de administración y podremos ver los usuarios en la pagina con nuestros derechos de administrador por el ataque a la cookie.

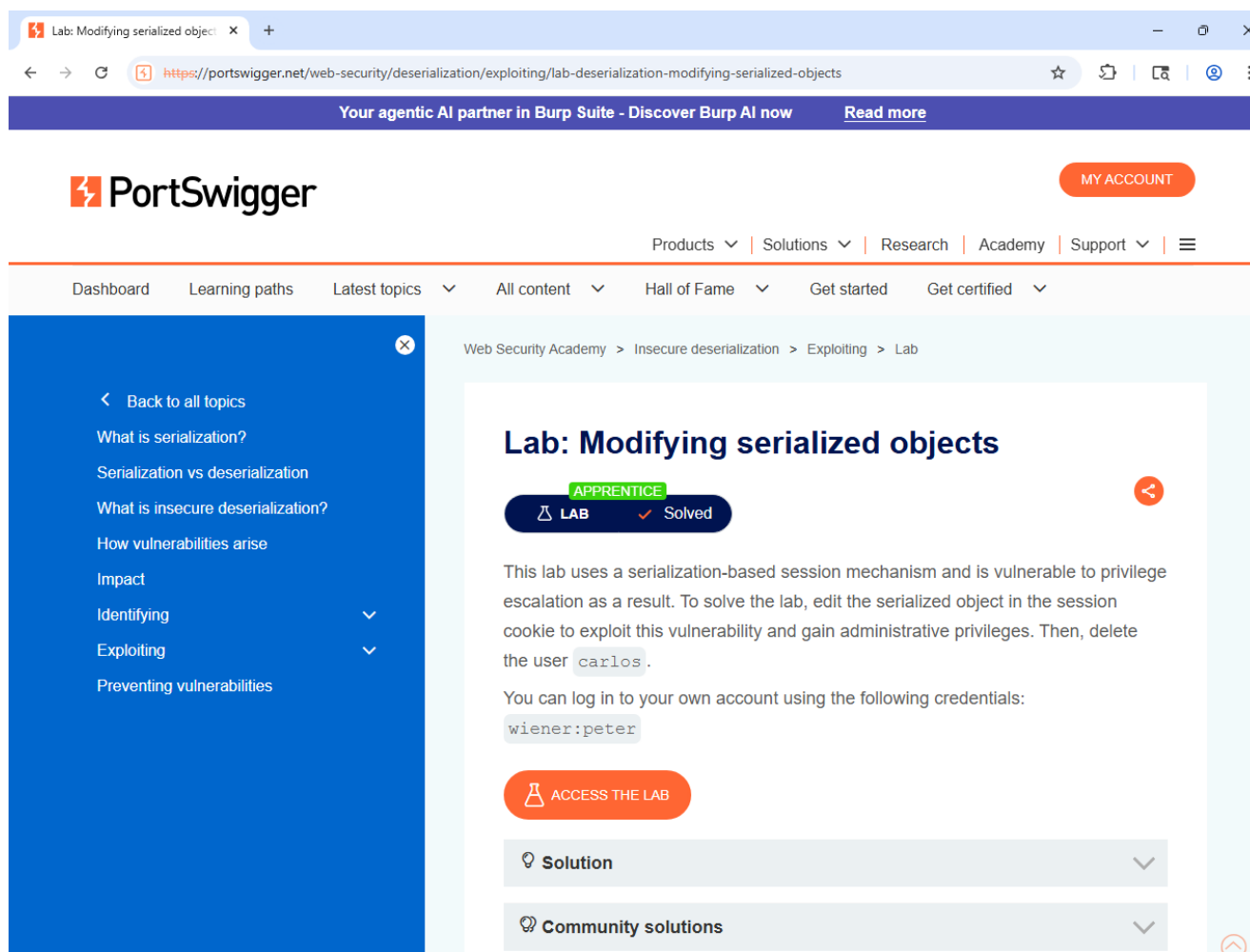
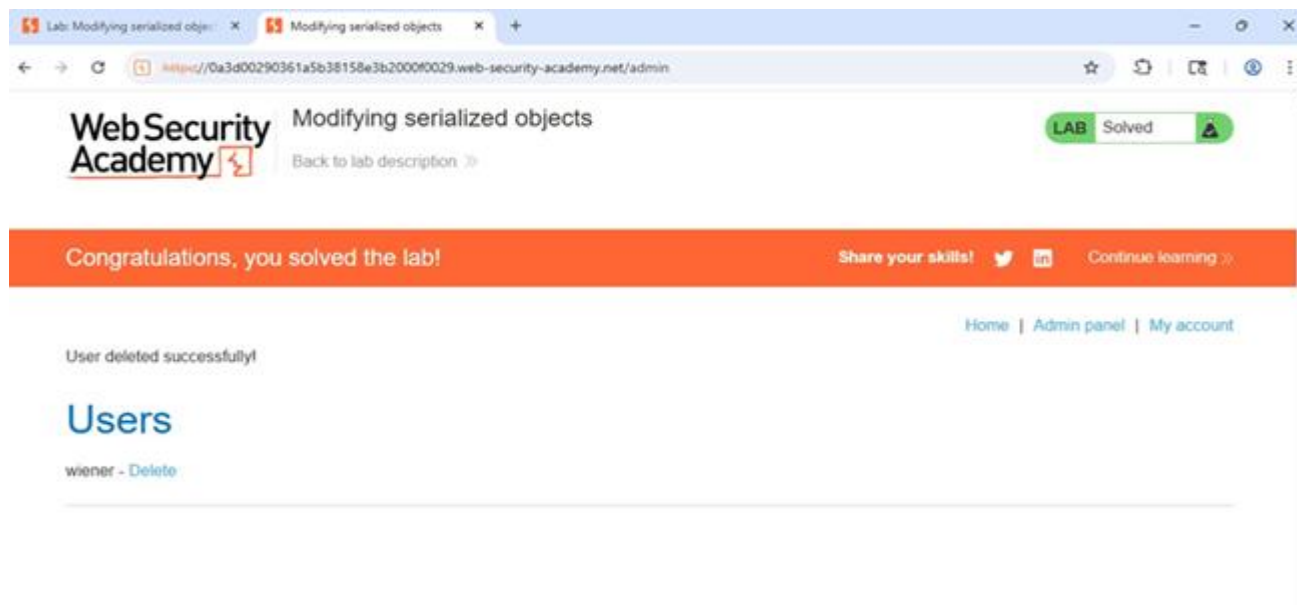


A continuación, realizaremos el mismo proceso, presionaremos en eliminar el usuario carlos, iremos a la captura, seleccionaremos la cookie, cambiaremos el valor del booleano a 1, aplicaremos el cambio y presionaremos forward (es posible que se deba realizar varias veces, hay que realizarlo más veces necesarias).

Al terminar el proceso veremos que el usuario carlos fue eliminado y nos da un mensaje de felicitaciones por haber terminado el ejercicio como fue solicitado.

Se adjuntan las imágenes donde señala que el ejercicio fue resuelto como fue solicitado por la profesora.

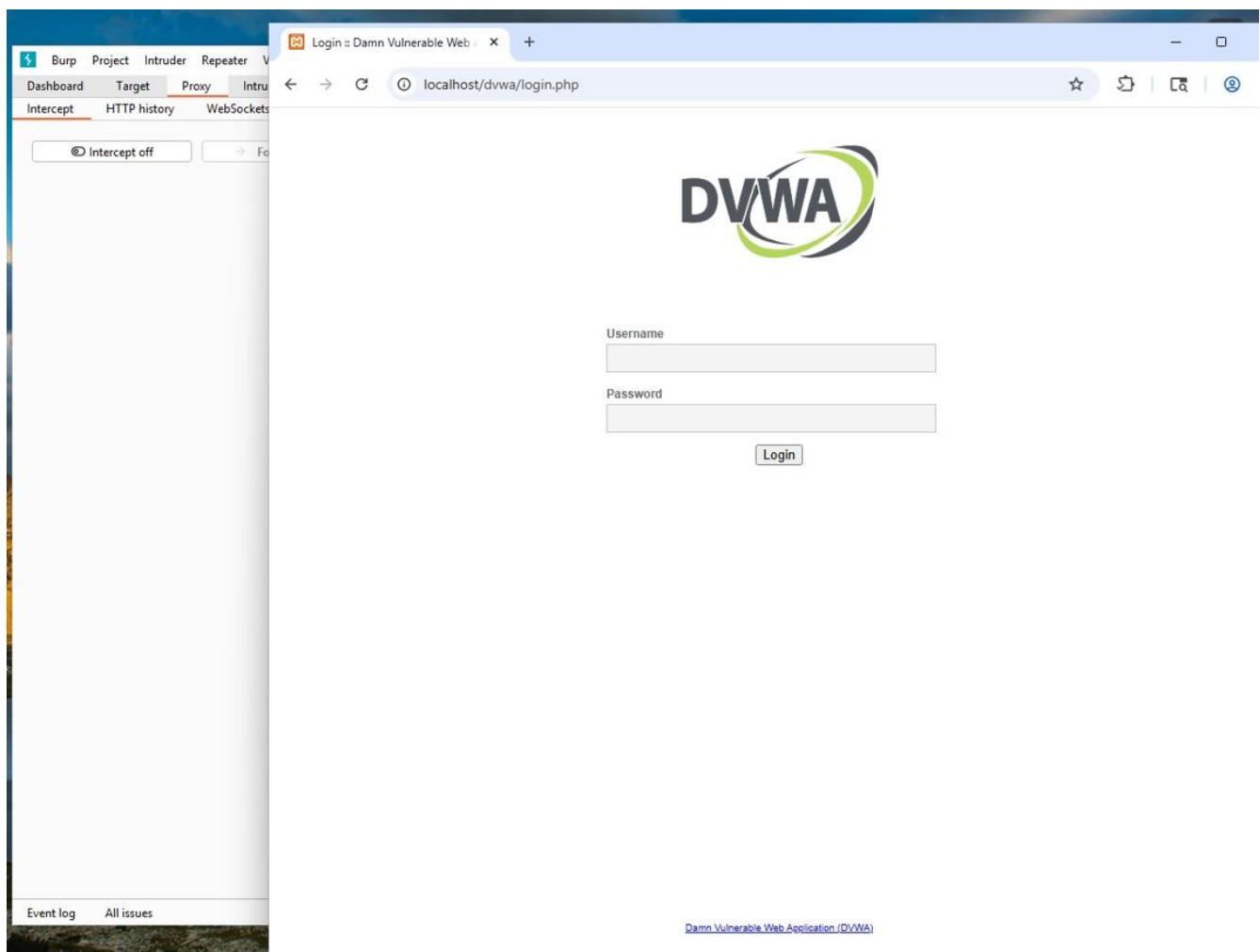




## Etapa 3

### Ataque al sitio

En este ejercicio abrimos el programa Burp Suite Community Edition para realizar el ataque a la página. Abrimos la página, nos logeamos con un usuario y contraseña (usuario: carlos, contraseña: cumple) y activamos la opción Intercept en el programa Burp para capturar la información de inicio de sesión.



En la siguiente imagen se puede notar como se capturan los datos de inicio de sesión. Para continuar es necesario presionar Forward (dos veces) para que inicie sesión.

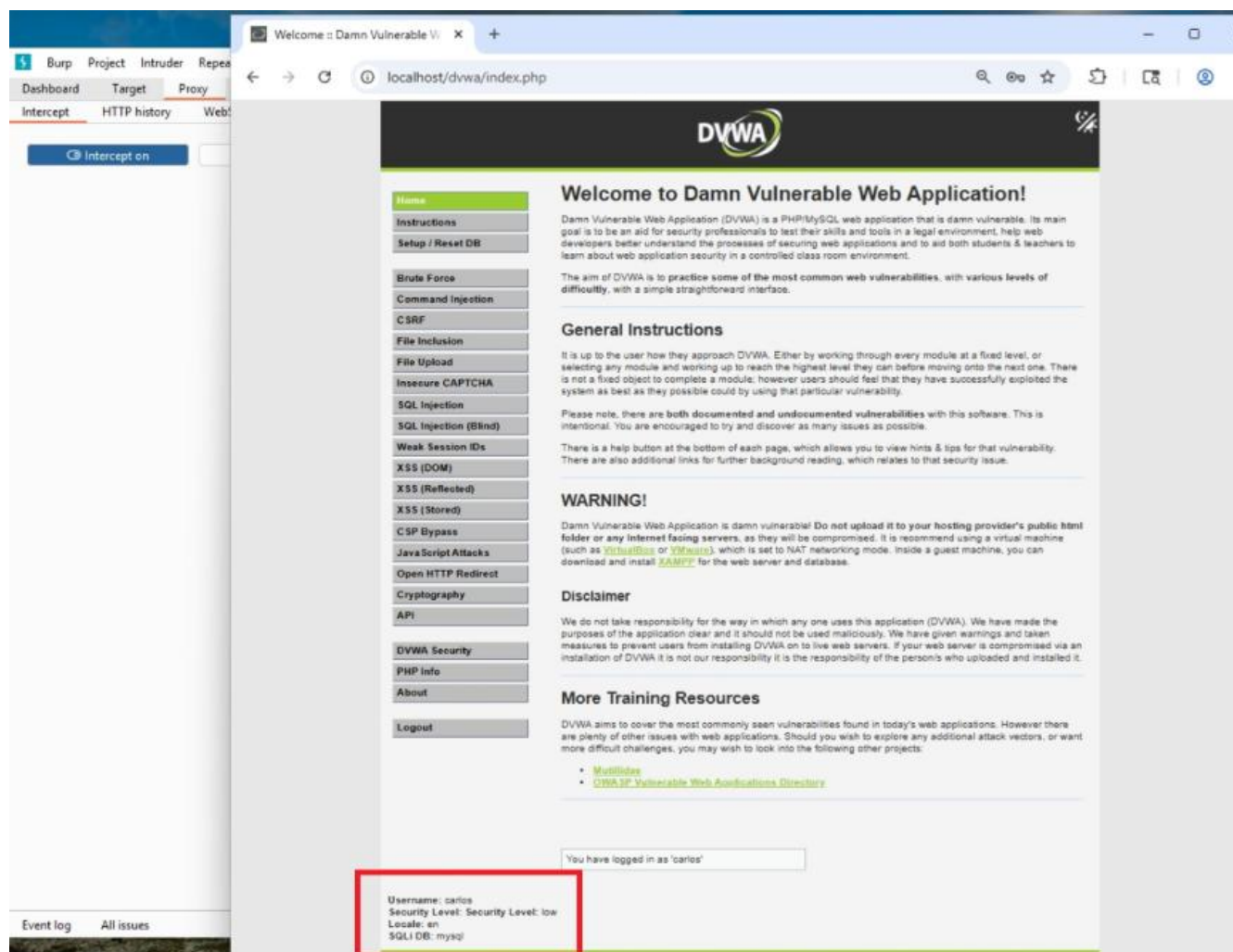
The screenshot displays the Burp Suite interface with a captured HTTP request. The 'Request' tab is active, showing the raw request data. The request is a POST to `http://localhost/dvwa/login.php`. The body of the request is highlighted with a red box, showing the following data:

```
username=carlos&password=cumple&Login=Login&user_token=a3a825e4e8fe2a5e067ba4bcb530131
```

The 'Inspector' panel on the right shows the request attributes, query parameters, body parameters, cookies, and headers. The request headers include:

- Host: localhost
- Content-Length: 87
- Cache-Control: max-age=0
- sec-ch-ua: "Chromium";v="141", "Not?A\_Brand";v="8"
- sec-ch-ua-mobile: ?0
- sec-ch-ua-platform: "Windows"
- Accept-Language: es-419,es;q=0.9
- Origin: http://localhost
- Content-Type: application/x-www-form-urlencoded
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 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-User: ?1
- Sec-Fetch-Dest: document
- Referer: http://localhost/dvwa/login.php
- Accept-Encoding: gzip, deflate, br
- Cookie: PHPSESSID=89fe85ulu4336213gcj96vz6qi; security=low
- Connection: keep-alive

Y se valida que el usuario inicia sesión en la plataforma.



A continuación, se cierra sesión. Y se intenta iniciar sesión nuevamente con las mismas credenciales y se capturan los datos. En esta ocasión nos vamos a la opción de inspector, en el apartado de Request body parameters cambiamos el usuario de carlos por luis y presionamos forward (2 veces) a lo cual nos va a marcar login failed.

Log in :: Damn Vulnerable Web

Intercept on Forward Drop

Request to http://localhost:80 [127.0.0.1]

Time Type Direction Method URL Status code Length

01:19:47 26... HTTP → Request POST http://localhost/dvwa/login.php

**Request**

Pretty Raw Hex

```
1 POST /dvwa/login.php HTTP/1.1
2 Host: localhost
3 Content-Length: 87
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: es-419,es;q=0.9
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://localhost/dvwa/login.php
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=85fe5ulu4336d13gcj96vr6ql; security=low
21 Connection: keep-alive
22
23 username=luis&password=cumple&Login=Login&user_token=e60f7c986d3261b7e1233aaec0f2fa05
```

**Inspector**

Selection 85 (0x55)

**Selected text**

```
username=luis&password=cumple&Login=Login&user_token=e60f7c986d3261b7e1233aaec0f2fa05
```

**Decoded from:** URL encoding

```
username=luis&password=cumple&Login=Login&user_token=e60f7c986d3261b7e1233aaec0f2fa05
```

Cancel Apply changes

Request attributes 2

Request query parameters 0

Request body parameters 4

**Name Value**

username	luis
----------	------

Event log All issues

Memory: 146.3MB Disabled

Log in :: Damn Vulnerable Web

localhost/dvwa/login.php

**DVWA**

Username

Password

Login failed

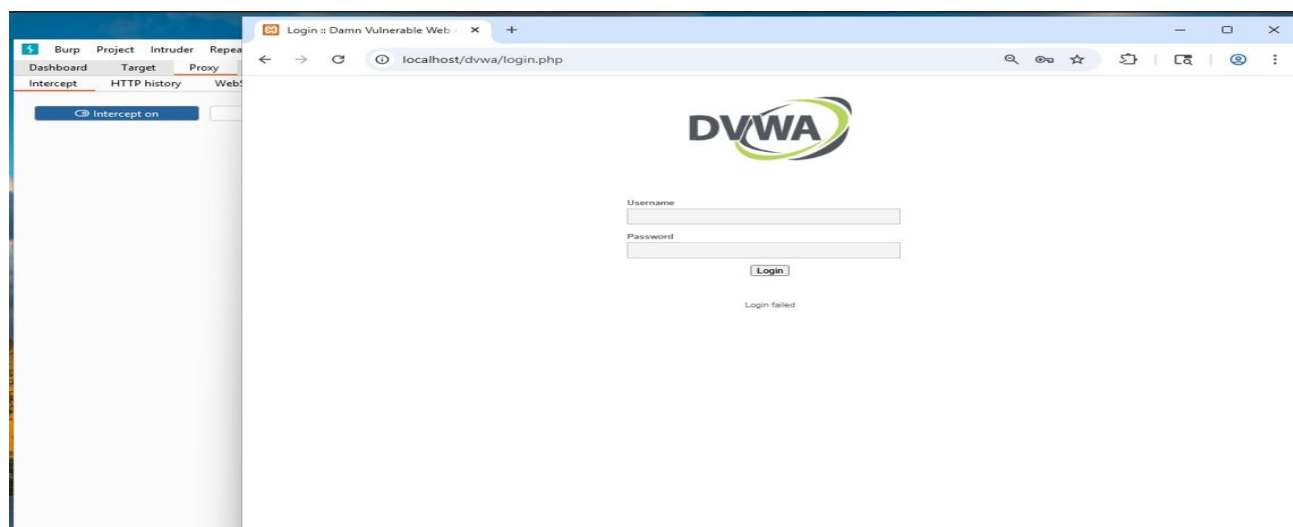
Event log All issues

Damn Vulnerable Web Application (2.0.0)

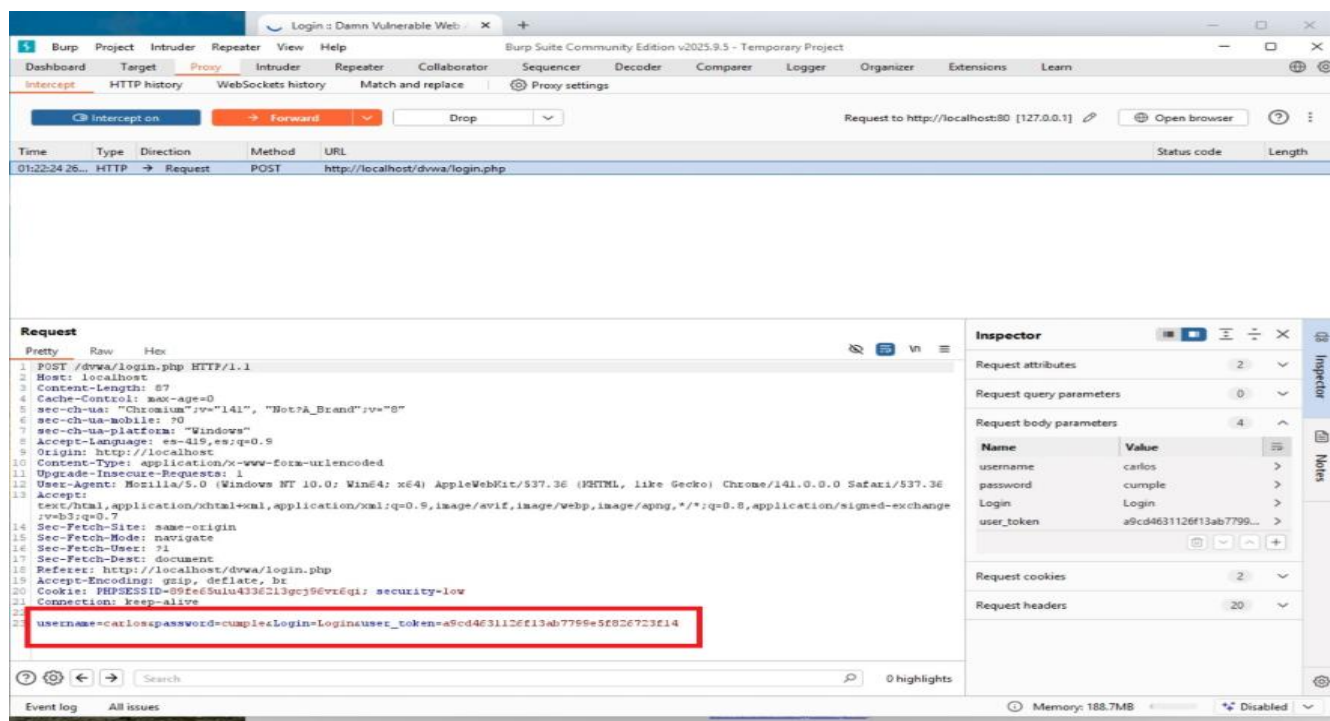
Se nos pide realizar nuevamente la prueba, pero esta vez cambiar la contraseña, realizamos el mismo proceso, pero esta vez en la opción de inspector, en el apartado de Request body parameters cambiamos el password de cumple por felicidades y presionamos forward (2 veces) a lo cual nos va a marcar login failed.

The screenshot shows the Burp Suite interface with a POST request to `http://localhost/dvwa/login.php`. The request body parameters are as follows:

Name	Value
username	carlos
password	felicidades
Login	Login
user_token	1c505bfc0b1e8b1f6f4754bc1594737



Y a continuación realizamos la siguiente prueba. Ingresaremos con los datos de carlos, pero al capturarlos los cambiaremos por los de otro usuario (marlen:marlen) que si esta registrado y nos permite ingresar a la página con los datos del usuario marlen.





The image shows a Burp Suite interface at the top and a web browser displaying the Damn Vulnerable Web Application (DVWA) at the bottom.

**Burp Suite - Request Tab:**

```

1 POST /dvwa/login.php HTTP/1.1
2 Host: localhost
3 Content-Length: 87
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="141", "Not?A_Brand";v="8"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: es-419,es;q=0.9
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
13 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
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://localhost/dvwa/login.php
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=89fe65ulu43362l3gcj96vr6ql; security=low
21 Connection: keep-alive
22
23 username=marlen&password=marlen&Login=Login&user_token=a9cd4631126f13ab7799e5f826723f14
  
```

**Burp Suite - Inspector Tab:**

**Selected text:**

```
username=marlen&password=marlen&Login=Login&user_token=a9cd4631126f13ab7799e5f826723f14
```

**Decoded from: URL encoding**

```
username=marlen&password=marlen&Login=Login&user_token=a9cd4631126f13ab7799e5f826723f14
```

**Request body parameters:**

Name	Value
username	marlen
password	marlen
Login	Login

**DVWA Web Application:**

The browser shows the DVWA homepage. The left sidebar contains a list of modules: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript Attacks, Open HTTP Redirect, Cryptography, API, DVWA Security, PHP Info, About, and Logout.

The main content area displays the "Welcome to Damn Vulnerable Web Application!" message. It includes a "WARNING!" section stating that DVWA is a PHP/MySQL web application designed for security professionals to test their skills in a legal environment. It also includes a "Disclaimer" and "More Training Resources" section.

At the bottom of the page, a message states: "You have logged in as 'marlen'". Below this, a red box highlights the following information:

```

Username: marlen
Security Level: Security Level: low
Locale: en
SQL DB: mysql
  
```



Realizamos la siguiente nuevamente. Ingresaremos con los datos de carlos, pero al capturarlos los cambiaremos por los de otro usuario (gordonb: 12345) que si esta registrado y nos permite ingresar a la página con los datos del usuario gordonb.

The screenshot shows the Burp Suite interface with a captured HTTP POST request to `http://localhost:80/dvwa/login.php`. The request body parameters are as follows:

Name	Value
username	carlos
password	cumples
Login	Login
user_token	a9cd4631126f13ab7799...

The screenshot shows the same Burp Suite interface, but the password parameter in the request body has been modified to '12345'. The updated request body parameters are:

Name	Value
password	12345

The Inspector panel shows the 'Decoded from' dropdown set to 'URL encoding'.

Dashboard Target Proxy Intercept HTTP history Web

Intercept on

Welcome : Damn Vulnerable Web Application

localhost/dvwa/index.php

## Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

### WARNING!

Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or any Internet facing servers, as they will be compromised. It is recommend using a virtual machine (such as [VirtualBox](#) or [VMware](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [SAMBY](#) for the web server and database.

### Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

### More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- [MultiTools](#)
- [OWASP Vulnerable Web Applications Directory](#)

You have logged in as 'gordonb'

Username: gordonb  
Security Level: Security Level: low  
Locale: en  
SQLi DB: mysql

Event log All issues

## Conclusión

Realizar este ejercicio fue de gran valor y de miedo, a nivel profesional me permitió experimentar de primera mano cómo fluyen las credenciales y como se procesan las peticiones en una autenticación, lo cual me ayuda a mejorar en mi comprensión y desarrollo para identificar y mitigar esas vulnerabilidades. Aprender y dominar estas técnicas nos ayudan a tener el conocimiento para el desarrollo, puesta en punto, pruebas de seguridad en los desarrollos que podamos tener, además de en los que podemos apoyar en auditorias o en su remediación de vulnerabilidades.

También nos ayuda a entender mejor por qué es muy importante proteger nuestras credenciales, usar contraseñas robustas, su correcta rotación (no siempre usar las mismas sin que venzan).

Esta materia me ayudo a abrir los ojos en muchas cosas, este trabajo fue muy divertido, para el que tuve que mirar muchos videos en entender cómo utilizar y poder sacarle más provecho a la herramienta, que es impresionante y me gustaría seguir aprendiendo de ella.

Después de terminar este trabajo me voy a ir a cambiar contraseñas, eliminar cookies, etc.

Gracias profesora por todo su apoyo y esas clases muy interesantes. Espero que este trabajo cumpla con lo solicitado.

Este trabajo fue subido al siguiente enlace de GitHub

<https://github.com/CarlosNico/Auditor-a-Inform-tica>

## Referencias

Monteagudo, D. (2010, March 24). *OWASP TOP 10 (III): Pérdida de autenticación y Gestión de Sesiones*. Security Art Work. <https://www.securityartwork.es/2010/03/24/owasp-top-10-iii-perdida-de-autenticacion-y-gestion-de-sesiones/>

*Wireshark • undefined*. (n.d.). Wireshark. Retrieved October 24, 2025, from <https://www.wireshark.org/>

*A2-Pérdida autenticación y gestión sesiones :: PROYECTO SEGURIDAD INFORMÁTICA*. (n.d.). Webnode.es. Retrieved October 24, 2025, from <https://liliseguridadinformatica.webnode.es/guia-de-buenas-practicas/disenio/a2/>

*Insecure deserialization*. (n.d.). Portswigger.net. Retrieved October 25, 2025, from <https://portswigger.net/web-security/deserialization>

Kumar, R., McKeever, G., Wright, M., Hasson, E., Cheng, L., Rohit Kumar, Guillotin, E., & Muly Levy. (n.d.). *Deserialization*. Learning Center; Imperva Inc. Retrieved October 25, 2025, from <https://www.imperva.com/learn/application-security/deserialization/>

*Lab: Authentication bypass via flawed state machine*. (n.d.). Portswigger.net. Retrieved October 25, 2025, from <https://portswigger.net/web-security/logic-flaws/examples/lab-logic-flaws-authentication-bypass-via-flawed-state-machine>

*Download XAMPP*. (n.d.). Apachefriends.org. Retrieved October 26, 2025, from <https://www.apachefriends.org/es/download.html>

*¿Qué es el scripting entre sitios (XSS)?* (n.d.). Fortinet. Retrieved October 26, 2025, from <https://www.fortinet.com/lat/resources/cyberglossary/cross-site-scripting>

Wikipedia contributors. (n.d.). *Cross-site scripting*. Wikipedia, The Free Encyclopedia. [https://es.wikipedia.org/w/index.php?title=Cross-site\\_scripting&oldid=170168179](https://es.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=170168179)  
Wood, R. (n.d.). *DVWA: Damn Vulnerable Web Application (DVWA)*.