Informe Laboratorio 2

Sección 1

cristóbal León
e-mail: cristobal.leon1@mail.udp.cl

19 de Abril de 2024

Índice

1. Descripción de actividades

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

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

2. Desarrollo de actividades según criterio de rúbrica

2.1. Levantamiento de docker para correr DVWA (dvwa)

Como se puede observar en la figura 1, el comando de Docker se utiliza para ejecutar un contenedor basado en la imagen vulnerables/web-dvwa en el entorno. Damn Vulnerable Web Application (DVWA) es una aplicación web deliberadamente insegura utilizada para practicar y aprender sobre vulnerabilidades web y pruebas de penetración.

```
(base) admin@iMac-de-Cristobal Lab2 % docker run --rm -it -p 8880:80 --platform
linux/amd64 vulnerables/web-dvwa
[+] Starting mysql...
[ ok ] Starting MariaDB database server: mysqld.
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reli
ably determine the server's fully qualified domain name, using 172.17.0.2. Set t
he 'ServerName' directive globally to suppress this message
. ok
==> /var/log/apache2/access.log <==

==> /var/log/apache2/error.log <==
[Wed Apr 17 21:59:16.588689 2024] [mpm_prefork:notice] [pid 316] AH00163: Apache
/2.4.25 (Debian) configured -- resuming normal operations
[Wed Apr 17 21:59:16.588794 2024] [core:notice] [pid 316] AH00094: Command line:
 '/usr/sbin/apache2'

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

Figura 1

2.2. Redirección de puertos en docker (dvwa)

-p 8880:80: Esta opción mapea el puerto 8880 del host al puerto 80 del contenedor. Especificar -p host-port:container-port permite redirigir el tráfico desde un puerto en el host hacia un puerto en el contenedor. En este caso, el servicio que se ejecuta en el puerto 80 dentro del contenedor estará accesible desde el puerto 8880 del host.

2.3. Obtención de consulta a replicar (burp)

Como se puede observar en la figura 2, dentro de la pestaña Proxy, la solicitud interceptada en Burp Suite fue una solicitud HTTP GET dirigida a la URL http://localhost:8880/vulnerabilities/brute/.
Los parámetros de la solicitud incluían un campo de usuario (username) y un campo de contraseña (password), así como un parámetro adicional para el botón de inicio de sesión (Login).
El método HTTP utilizado fue GET, ya que los datos de inicio de sesión se enviaron como parte de la URL.

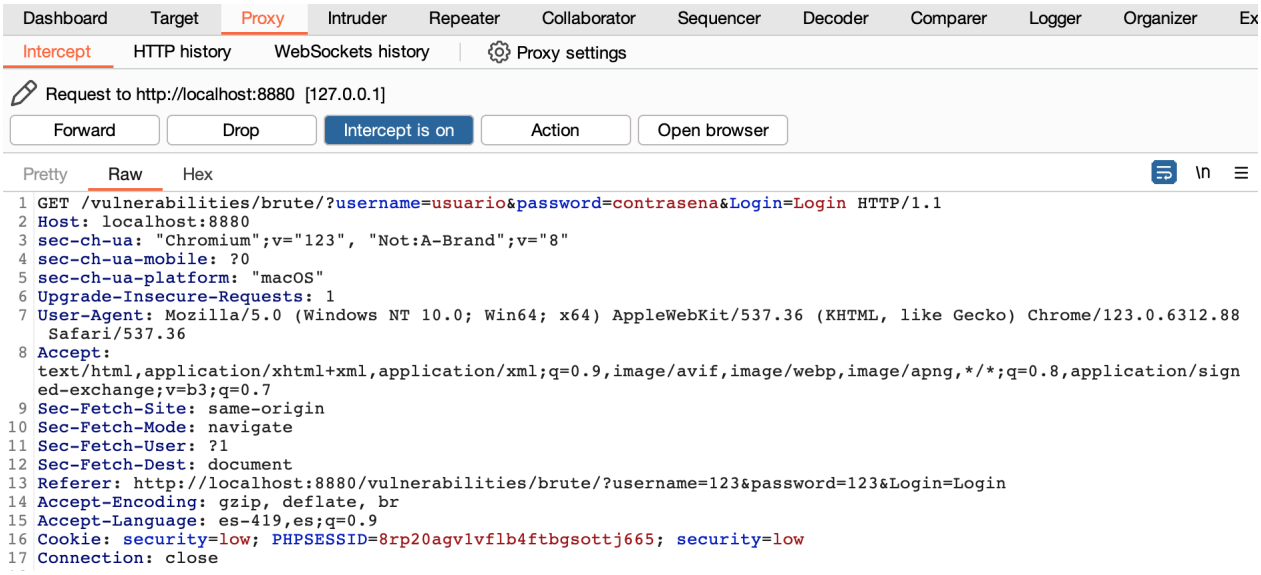

Figura 2

2.4. Identificación de campos a modificar (burp)

Como se puede observar en la figura 3, durante el ataque de fuerza bruta, se identificaron los campos de usuario (username) y contraseña (password) como objetivos para la modificación.
Estos campos fueron seleccionados como posiciones en Burp Intruder, lo que permitió la variación de los valores de usuario y contraseña en cada solicitud.

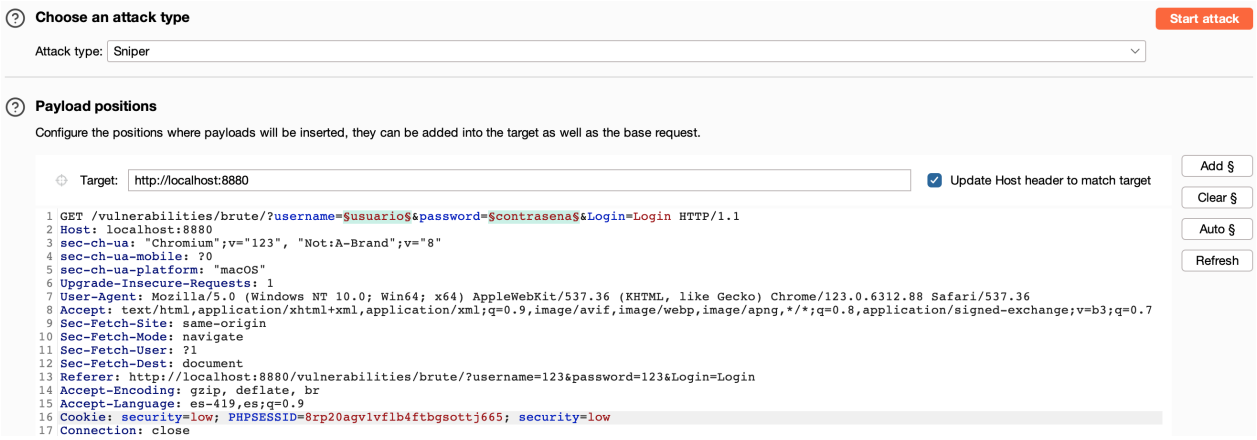

Figura 3

2.5. Obtención de diccionarios para el ataque (burp)

Como se puede observar en la figura 4, se utiliza un diccionario predefinido que contiene una lista de usuarios y contraseñas para el ataque de fuerza bruta.

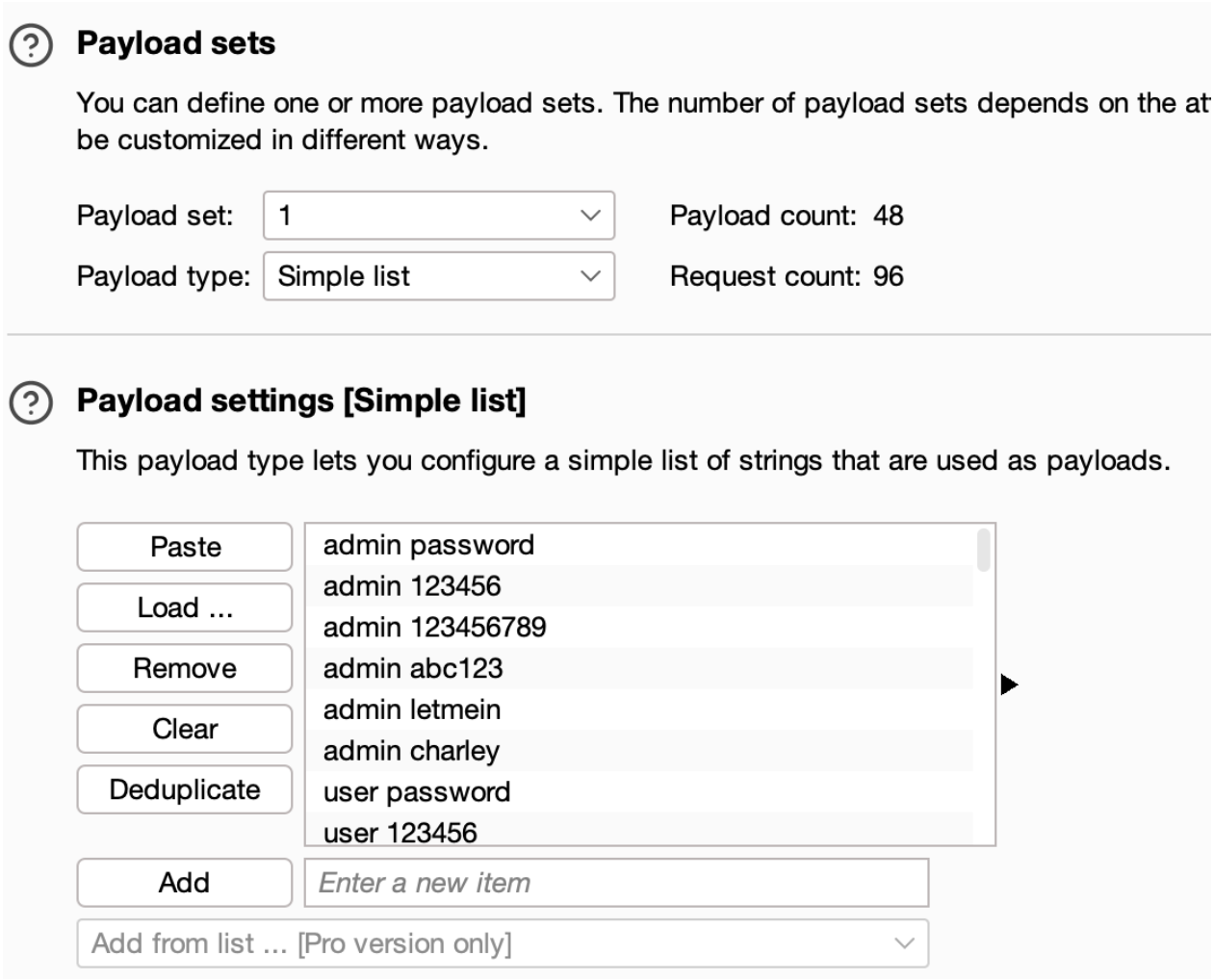

Figura 4

2.6. Obtención de al menos 2 pares (burp)

Como se puede observar en la figura 5, Durante el ataque de fuerza bruta, se identificaron varios pares de usuario/contraseña válidos y no válidos.

Dos ejemplos de pares válidos son:
Usuario: admin / Contraseña: password
Usuario: 1337 / Contraseña: charley

Dos ejemplos de pares no válidos son:
Usuario: admin / Contraseña: 123456
Usuario: 123 / Contraseña: password

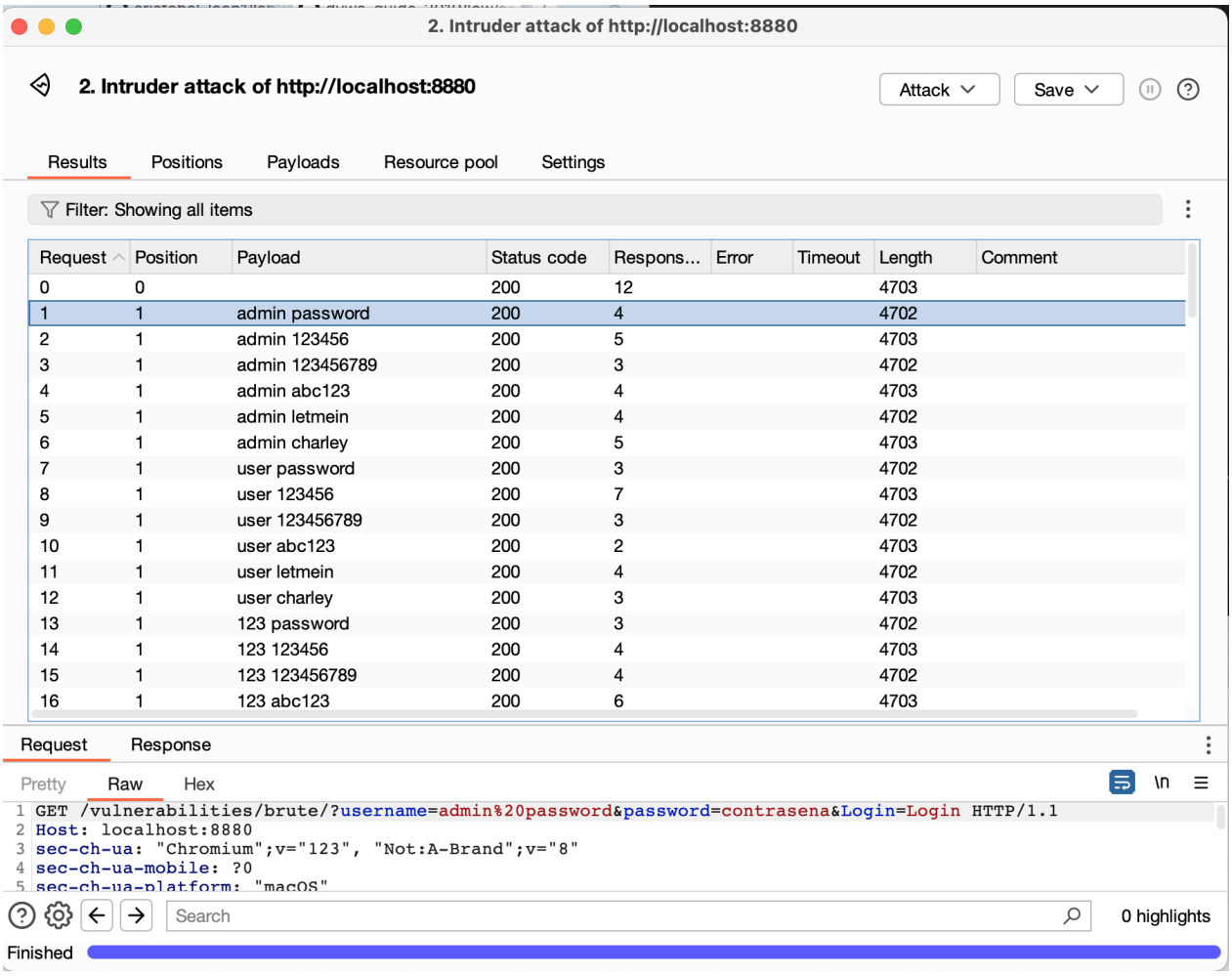

Figura 5

2.7. Obtención de código de inspect element (curl)

Como se puede observar en la figura 6, se inspeccionan y se identifican los elementos relevantes de la página que se atacará con fuerza bruta. En este caso, se refiere a todo lo que está dentro de `<div class="vulnerable_code_area»`.

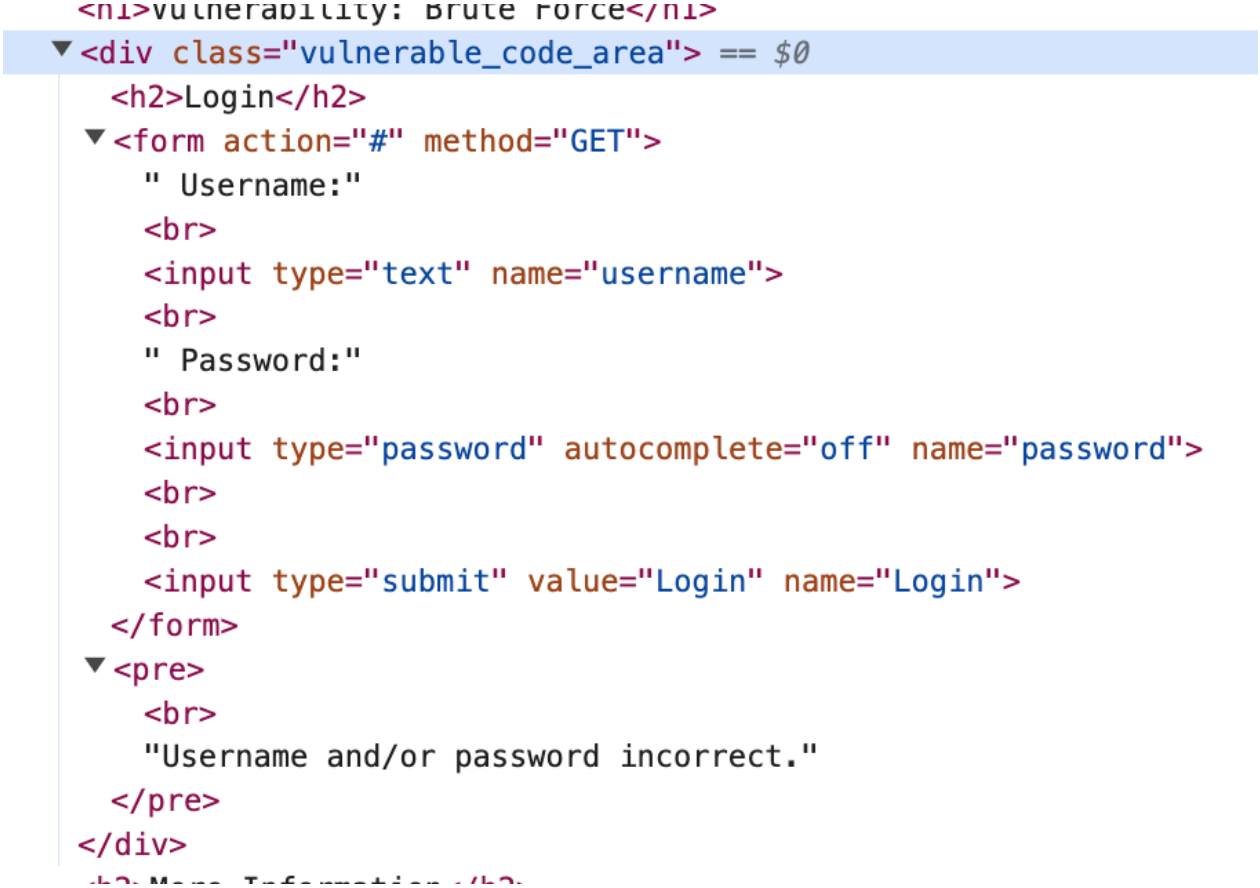

Figura 6

2.8. Utilización de curl por terminal (curl)

Comando con credenciales válidas:

```
(base) admin@iMac-de-Cristobal 3curl % curl -X POST -L --data "username=admin&password=
hola&Login=Login" http://localhost:8880/vulnerabilities/brute > respuesta_correcta.txt

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   369  100   329  100    40   151k  18885 --:--:-- --:--:-- --:--:--  180k
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  1523  100  1523    0     0   216k      0 --:--:-- --:--:-- --:--:--  216k
```

Figura 7

- `curl`: Es el comando utilizado para realizar solicitudes HTTP desde la línea de comandos.
- `-X POST`: Especifica que se enviará una solicitud POST al servidor.
- `-L`: Indica a cURL que siga cualquier redirección que el servidor pueda enviar en respuesta a la solicitud.
- `--data ünername=admin&password=hola&Login=Login"`: Especifica los datos que se enviarán en el cuerpo de la solicitud. En este caso, se están proporcionando credenciales de inicio de sesión válidas: un nombre de usuario .ªdminz una contraseña "hola".
- `http://localhost:8880/vulnerabilities/brute`: Es la URL a la que se enviará la solicitud POST.
- `>respuesta_correcta.txt`: Redirige la salida de la solicitud HTTP (la respuesta del servidor) al archivo `respuesta_correcta.txt`.

Comando con credenciales inválidas:

```
(base) admin@iMac-de-Cristobal 3curl % curl -X POST -L --data "username=admin&hola=hola
&Login=Login" http://localhost:8880/vulnerabilities/brute > respuesta_incorrecta.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   365  100   329  100    36   113k  12689 --:--:-- --:--:-- --:--:--  178k
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  1523  100  1523    0     0   180k      0 --:--:-- --:--:-- --:--:--  180k
```

Figura 8

- `curl`: Igual que en el comando anterior.
- `-X POST`: También igual al comando anterior.

- `-L`: Lo mismo que en el comando anterior.
- `--data ünername=admin&hola=hola&Login=Login"`: En este caso, se están proporcionando credenciales de inicio de sesión inválidas: un nombre de usuario .ªdminz una contraseña "hola"(la contraseña se especifica como "hola").
- `http://localhost:8880/vulnerabilities/brute`: La misma URL que en el comando anterior.
- `>respuesta_incorrecta.txt`: Redirige la salida de la solicitud HTTP (la respuesta del servidor) al archivo `respuesta_incorrecta.txt`.

2.9. Demuestra 5 diferencias (curl)

La diferencia mostrada en la figura 9 identificada en la salida del comando `diff respuesta_correcta respuesta_incorrecta.txt` es la siguiente:

En la línea 47 del archivo `respuesta_correcta.txt`, el valor del atributo `value` del elemento `<input>` con el atributo `name` igual a `'user_token'` es `28be2ea5ffaddbbeb94979ca57ad8889`.

Mientras que en la línea 47 del archivo `respuesta_incorrecta.txt`, el valor del atributo `value` del mismo elemento `<input>` es `8599347327ad760d6894c71d42c6f866`.

Esta diferencia sugiere que el token de usuario (`user_token`) generado por el servidor es distinto en cada caso, lo que podría indicar un mecanismo de seguridad para prevenir ataques de reproducción o falsificación de solicitudes entre sitios (CSRF).

```
(base) admin@iMac-de-Cristobal 3curl %
diff respuesta_correcta.txt respuesta_incorrecta.txt

47c47
<       <input type='hidden' name='user_token' value='28be2ea5ffaddbbeb94979ca57ad8889'
 />
---
>       <input type='hidden' name='user_token' value='8599347327ad760d6894c71d42c6f866'
 />
```

Figura 9

2.10. Instalación y versión a utilizar (hydra)

Para instalar Hydra en sistemas basados en macOS, se utilizó el gestor de paquetes Homebrew. El comando utilizado fue brew install hydra. Este comando descargó e instaló la última versión disponible de Hydra en el sistema local.

Como se puede observar en la figura 10, Se ejecutó el comando hydra –version en la terminal para verificar la versión de Hydra instalada en el sistema. El propósito de esta verificación fue asegurarse de que la instalación de Hydra se realizó correctamente y determinar si se está utilizando la versión más reciente del software.

La salida del comando mostró que la versión instalada de Hydra es la 9.5. Esto confirma que la instalación se realizó correctamente y que se está utilizando la última versión disponible de Hydra para llevar a cabo los ataques de fuerza bruta de manera efectiva.

```
(base) admin@iMac-de-Cristobal 4HYDRA % hydra --version
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in mi
litary or secret service organizations, or for illegal purposes (this is non-bin
ding, these *** ignore laws and ethics anyway).

hydra: illegal option -- -
```

Figura 10

2.11. Explicación de comando a utilizar (hydra)

Comando: `hydra -L usuarios.txt -P contraseñas.txt -V 'http-post-form://localhost:8880/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:Login fallido'`

Parámetros del comando:

-L usuarios.txt: Especifica un archivo de texto llamado usuarios.txt que contiene una lista de nombres de usuario que se probarán durante el ataque. Hydra probará cada nombre de usuario en este archivo.

-P contraseñas.txt: Especifica un archivo de texto llamado contraseñas.txt que contiene una lista de contraseñas que se probarán durante el ataque. Hydra probará cada contraseña en este archivo junto con cada nombre de usuario.

-V: Habilita el modo verbose o detallado, lo que permite que Hydra muestre información detallada sobre el progreso y los resultados del ataque mientras se ejecuta.

`http-post-form://localhost:8880/vulnerabilities/brute/index.php:username=^USER^&password=^PASS^&Login=Login:Login fallido`: Especifica la URL del formulario de inicio de sesión como `http-post-form://localhost:8880/vulnerabilities/brute/index.php`, donde se enviarán las solicitudes de inicio de sesión. Los marcadores ^USER^ y ^PASS^ son marcadores de posición que serán reemplazados por los nombres de usuario y contraseñas de la lista de archivos especificada anteriormente. `Login=Login` indica el botón de inicio de sesión en el formulario. `Login fallido` es una cadena que Hydra utiliza para identificar si el inicio de sesión fue exitoso o no. Si Hydra encuentra esta cadena en la respuesta del servidor, determinará que el inicio de sesión fue fallido y probará la siguiente combinación de usuario/contraseña.

Funcionamiento del comando:

Hydra leerá cada nombre de usuario del archivo `usuarios.txt` y cada contraseña del archivo `contraseñas.txt`.

Para cada combinación de nombre de usuario y contraseña, Hydra construirá una solicitud POST al formulario de inicio de sesión especificado en la URL `http-post-form://localhost:8880/vulnerabilities/brute/index.php`. Los marcadores ^USER^ y ^PASS^ serán reemplazados por el nombre de usuario y la contraseña actualmente probados.

Hydra enviará la solicitud POST al servidor web y verificará si la respuesta contiene la cadena `Login fallido`. Si lo hace, Hydra registrará que el inicio de sesión fue fallido y continuará probando con las siguientes combinaciones de usuario/contraseña. Si no lo hace, Hydra registrará que el inicio de sesión fue exitoso y finalizará el ataque.

2.12. Obtención de al menos 2 pares (hydra)

Como se puede observar en la figura 11, Durante el ataque de fuerza bruta, se identificaron varios pares de usuario/contraseña válidos.

Dos ejemplos de pares válidos son:
Usuario: pablo / Contraseña: letmein
Usuario: 1337 / Contraseña: charley

```
[8880][http-post-form] host: localhost   login: pablo   password: abc123
[8880][http-post-form] host: localhost   login: pablo   password: 123456789
[8880][http-post-form] host: localhost   login: felipe   password: password
[8880][http-post-form] host: localhost   login: pablo   password: letmein
[8880][http-post-form] host: localhost   login: felipe   password: 123456
[8880][http-post-form] host: localhost   login: felipe   password: 123456789
[8880][http-post-form] host: localhost   login: pablo   password: charley
[8880][http-post-form] host: localhost   login: 1337   password: 123456789
[8880][http-post-form] host: localhost   login: 1337   password: password
[8880][http-post-form] host: localhost   login: felipe   password: charley
[8880][http-post-form] host: localhost   login: felipe   password: abc123
[8880][http-post-form] host: localhost   login: felipe   password: letmein
[8880][http-post-form] host: localhost   login: 1337   password: 123456
[8880][http-post-form] host: localhost   login: 1337   password: abc123
[8880][http-post-form] host: localhost   login: 1337   password: charley
[8880][http-post-form] host: localhost   login: 1337   password: letmein
1 of 1 target successfully completed, 48 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-17 19:33:19
```

Figura 11

2.13. Explicación paquete curl (tráfico)

Este fragmento de Hypertext Transfer Protocol (HTTP) el cual es mostrado en la figura 12 corresponde a una solicitud POST generada por el comando curl que se envió para acceder a un formulario de inicio de sesión en una aplicación web. Aquí está el análisis detallado:

- Método de solicitud (*Request Method*): Es POST, lo que indica que la solicitud está enviando datos al servidor para ser procesados.
- URI de solicitud (*Request URI*): /vulnerabilities/brute, indica la ruta en la que se está realizando la solicitud dentro del servidor.
- Versión de solicitud (*Request Version*): HTTP/1.1, la versión del protocolo HTTP utilizada en la solicitud.
- Host: localhost:8880, el nombre de host y el puerto del servidor al que se dirige la solicitud.
- User-Agent: curl/7.87.0, el User-Agent indica la aplicación cliente que está realizando la solicitud. En este caso, es curl con la versión 7.87.0.
- Accept: /, indica que el cliente acepta cualquier tipo de respuesta del servidor.
- Content-Length: 40, indica la longitud del cuerpo de la solicitud en bytes.
- Content-Type: application/x-www-form-urlencoded, especifica el tipo de contenido del cuerpo de la solicitud, que en este caso es una serie de pares de nombre-valor codificados en URL.
- Cuerpo de la solicitud (*Request Body*): Los datos del formulario de inicio de sesión se envían en el cuerpo de la solicitud. En este caso, se están enviando los datos de inicio de sesión del usuario .ªdminçon la contraseña "hola".

En resumen, este fragmento de HTTP representa una solicitud POST que envía datos de inicio de sesión al formulario de inicio de sesión en la ruta /vulnerabilities/brute del servidor local en el puerto 8880. La solicitud está siendo realizada por curl en nombre del usuario.

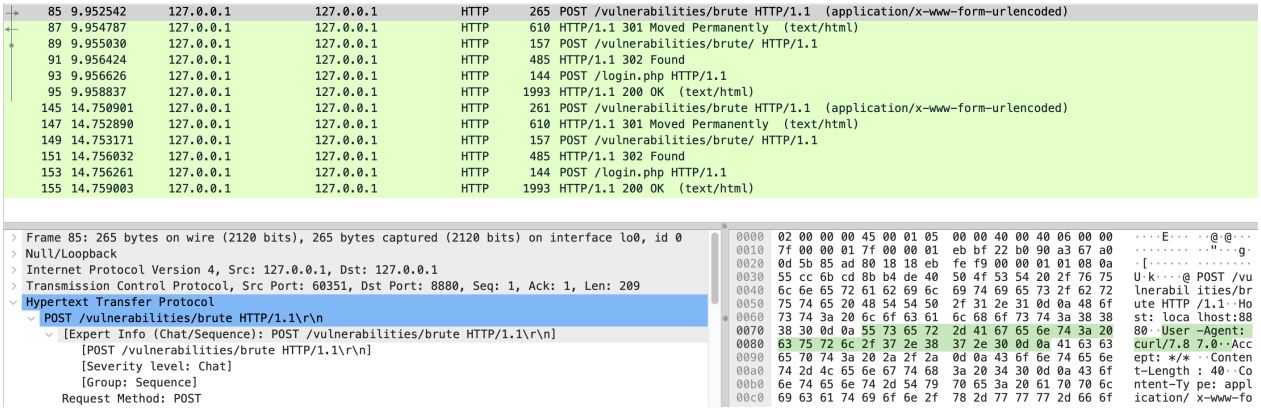
85 9.952542 127.0.0.1 127.0.0.1 HTTP 265 POST /vulnerabilities/brute HTTP/1.1 (application/x-www-form-urlencoded)
87 9.954787 127.0.0.1 127.0.0.1 HTTP 610 HTTP/1.1 301 Moved Permanently (text/html)
89 9.955030 127.0.0.1 127.0.0.1 HTTP 157 POST /vulnerabilities/brute/ HTTP/1.1
91 9.956424 127.0.0.1 127.0.0.1 HTTP 485 HTTP/1.1 302 Found
93 9.956626 127.0.0.1 127.0.0.1 HTTP 144 POST /login.php HTTP/1.1
95 9.958837 127.0.0.1 127.0.0.1 HTTP 1993 HTTP/1.1 200 OK (text/html)
145 14.750901 127.0.0.1 127.0.0.1 HTTP 261 POST /vulnerabilities/brute HTTP/1.1 (application/x-www-form-urlencoded)
147 14.752890 127.0.0.1 127.0.0.1 HTTP 610 HTTP/1.1 301 Moved Permanently (text/html)
149 14.753171 127.0.0.1 127.0.0.1 HTTP 157 POST /vulnerabilities/brute/ HTTP/1.1
151 14.756032 127.0.0.1 127.0.0.1 HTTP 485 HTTP/1.1 302 Found
153 14.756261 127.0.0.1 127.0.0.1 HTTP 144 POST /login.php HTTP/1.1
155 14.759003 127.0.0.1 127.0.0.1 HTTP 1993 HTTP/1.1 200 OK (text/html)
Frame 85: 265 bytes on wire (2120 bits), 265 bytes captured (2120 bits) on interface lo0, id 0
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 60351, Dst Port: 8880, Seq: 1, Ack: 1, Len: 209
Hypertext Transfer Protocol
POST /vulnerabilities/brute HTTP/1.1\r\n
[Expert Info (Chat/Sequence): POST /vulnerabilities/brute HTTP/1.1\r\n]
[POST /vulnerabilities/brute HTTP/1.1\r\n]
[Severity level: Chat]
[Group: Sequence]
Request Method: POST
0000 02 00 00 00 45 00 01 05 00 00 40 00 40 06 00 00 ····E··· ··@·@···
0010 7f 00 00 01 7f 00 00 01 eb bf 22 b0 90 a3 67 a0 ·····"···g·
0020 0d 5b 85 ad 80 18 18 eb fe f9 00 00 01 01 08 0a ·[·····
0030 55 cc 6b cd 8b b4 de 40 50 4f 53 54 20 2f 76 75 U·k····@ POST /vu
0040 6c 6e 65 72 61 62 69 6c 69 74 69 65 73 2f 62 72 lnerabil ities/br
0050 75 74 65 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f ute HTTP /1.1··Ho
0060 73 74 3a 20 6c 6f 63 61 6c 68 6f 73 74 3a 38 38 st: loca lhost:88
0070 38 30 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 80··User -Agent:
0080 63 75 72 6c 2f 37 2e 38 37 2e 30 0d 0a 41 63 63 curl/7.8 7.0··Acc
0090 65 70 74 3a 20 2a 2f 2a 0d 0a 43 6f 6e 74 65 6e ept: */* ··Conten
00a0 74 2d 4c 65 6e 67 74 68 3a 20 34 30 0d 0a 43 6f t-Length : 40··Co
00b0 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c ntent-Ty pe: appl
00c0 69 63 61 74 69 6f 6e 2f 78 2d 77 77 77 2d 66 6f ication/ x-www-fo

Figura 12

Además tiene un fragmento HTML Form URL Encoded: application/x-www-form-urlencoded tal como se puede ver en la figura 13 que describe el contenido del cuerpo de la solicitud HTTP en formato de formulario URL codificado (application/x-www-form-urlencoded). Aquí está el análisis detallado:

- Formulario codificado en URL (*URL Encoded Form*): Indica que los datos están codificados en el formato de formulario URL, donde los pares clave-valor están separados por el símbolo "z los valores están codificados para ser transmitidos a través de una URL.
- Elementos del formulario (*Form items*):
  - üsername- .ªdmin": Representa el campo del formulario con el nombre üsernamez el valor .ªdmin". En este caso, el nombre de usuario que se está enviando al servidor es .ªdmin".
  - Clave (*Key*): username, Valor (*Value*): admin
  - "password- "hola": Representa el campo del formulario con el nombre "passwordz el valor "hola". Aquí, la contraseña que se está enviando al servidor es "hola".
  - Clave (*Key*): password, Valor (*Value*): hola
  - "Login- "Login": Representa el botón de envío del formulario con el nombre "Loginz el valor "Login". Este campo indica que el usuario ha hecho clic en el botón de inicio de sesión para enviar los datos al servidor.
  - Clave (*Key*): Login, Valor (*Value*): Login

En resumen, este fragmento describe los datos enviados al servidor en el cuerpo de la solicitud HTTP. Incluye los valores de los campos del formulario de inicio de sesión, como el nombre de usuario (.ªdmin"), la contraseña ("hola"), y la acción de enviar el formulario ("Login"). Los valores están codificados en formato de formulario URL para la transmisión a través de la red.

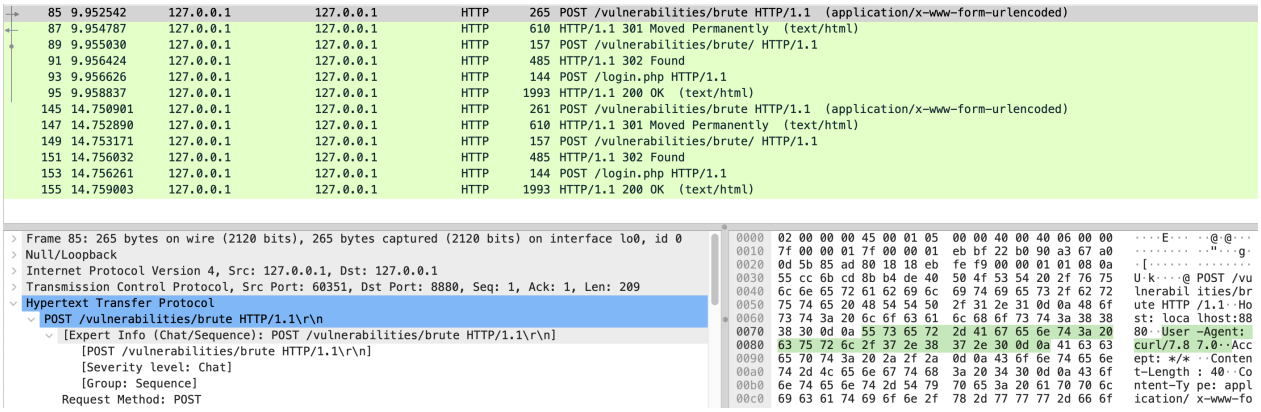

Figura 13

2.14. Explicación paquete burp (tráfico)

Este es un paquete capturado en Wireshark que muestra una solicitud HTTP GET enviada al servidor web local en el puerto 8880, el cual se puede observar en la figura 14. Aquí está el análisis de la solicitud:

```
Request Method: GET
Request URI: /vulnerabilities/brute/?username=user%20abc123&password=123&Login=Login
Request URI Path: /vulnerabilities/brute/
Request URI Query: username=user%20abc123&password=123&Login=Login
Request URI Query Parameter: username=user%20abc123
Request URI Query Parameter: password=123
Request URI Query Parameter: Login=Login
Request Version: HTTP/1.1
Host: localhost:8880
sec-ch-ua: Información sobre el agente de usuario del navegador
sec-ch-ua-mobile: Información sobre la versión móvil del agente de usuario
sec-ch-ua-platform: Plataforma del agente de usuario del navegador
Upgrade-Insecure-Requests: Indica si el navegador prefiere usar una conexión segura HT
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Accept: Tipos de contenido que el cliente puede procesar
Sec-Fetch-Site: Indica el sitio desde el que se realizó la solicitud
Sec-Fetch-Mode: Modo en que se realiza la solicitud (en este caso, navegación)
Sec-Fetch-User: Indica si el usuario está navegando en modo incógnito
Sec-Fetch-Dest: Indica el destino de la solicitud (en este caso, documento)
Referer: La URL de la página desde la que se realizó la solicitud
Accept-Encoding: Tipos de compresión aceptados por el cliente
Accept-Language: Idiomas preferidos por el cliente
Cookie: Información de las cookies enviadas con la solicitud
Connection: Tipo de conexión (en este caso, keep-alive)
```

Esta solicitud parece ser parte de un intento de inicio de sesión en un formulario de inicio de sesión en el sitio web vulnerable DVWA (Damn Vulnerable Web Application). Contiene información de usuario y contraseña en los parámetros de la URL, junto con otros datos de solicitud típicos.

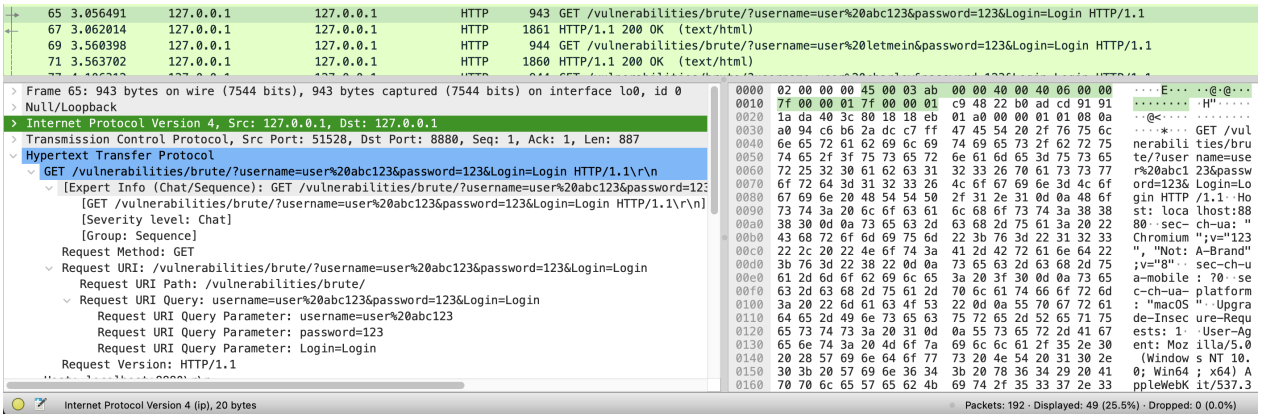
65 3.056491 127.0.0.1 127.0.0.1 HTTP 943 GET /vulnerabilities/brute/?username=user%20abc123&password=123&Login=Login HTTP/1.1
67 3.062014 127.0.0.1 127.0.0.1 HTTP 1861 HTTP/1.1 200 OK (text/html)
69 3.560398 127.0.0.1 127.0.0.1 HTTP 944 GET /vulnerabilities/brute/?username=user%20letmein&password=123&Login=Login HTTP/1.1
71 3.563702 127.0.0.1 127.0.0.1 HTTP 1860 HTTP/1.1 200 OK (text/html)
Frame 65: 943 bytes on wire (7544 bits), 943 bytes captured (7544 bits) on interface lo0, id 0
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 51528, Dst Port: 8880, Seq: 1, Ack: 1, Len: 887
Hypertext Transfer Protocol
GET /vulnerabilities/brute/?username=user%20abc123&password=123&Login=Login HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=user%20abc123&password=123
[GET /vulnerabilities/brute/?username=user%20abc123&password=123&Login=Login HTTP/1.1\r\n]
[Severity level: Chat]
[Group: Sequence]
Request Method: GET
Request URI: /vulnerabilities/brute/?username=user%20abc123&password=123&Login=Login
Request URI Path: /vulnerabilities/brute/
Request URI Query: username=user%20abc123&password=123&Login=Login
Request URI Query Parameter: username=user%20abc123
Request URI Query Parameter: password=123
Request URI Query Parameter: Login=Login
Request Version: HTTP/1.1
0000 02 00 00 00 45 00 03 ab 00 00 40 00 40 06 00 00
0010 7f 00 00 01 7f 00 00 01 c9 48 22 b0 ad cd 91 91
0020 1a da 40 3c 80 18 18 eb 01 a0 00 00 01 01 08 0a
0030 a0 94 c6 b6 2a dc c7 ff 47 45 54 20 2f 76 75 6c
0040 6e 65 72 61 62 69 6c 69 74 69 65 73 2f 62 72 75
0050 74 65 2f 3f 75 73 65 72 6e 61 6d 65 3d 75 73 65
0060 72 25 32 30 61 62 63 31 32 33 26 70 61 73 73 77
0070 6f 72 64 3d 31 32 33 26 4c 6f 67 69 6e 3d 4c 6f
0080 67 69 6e 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f
0090 73 74 3a 20 6c 6f 63 61 6c 68 6f 73 74 3a 38 38
00a0 38 30 0d 0a 73 65 63 2d 63 68 2d 75 61 3a 20 22
00b0 43 68 72 6f 6d 69 75 6d 22 3b 76 3d 22 31 32 33
00c0 22 2c 20 22 4e 6f 74 3a 41 2d 42 72 61 6e 64 22
00d0 3b 76 3d 22 38 22 0d 0a 73 65 63 2d 63 68 2d 75
00e0 61 2d 6d 6f 62 69 6c 65 3a 20 3f 30 0d 0a 73 65
00f0 63 2d 63 68 2d 75 61 2d 70 6c 61 74 66 6f 72 6d
0100 3a 20 22 6d 61 63 4f 53 22 0d 0a 55 70 67 72 61
0110 64 65 2d 49 6e 73 65 63 75 72 65 2d 52 65 71 75
0120 65 73 74 73 3a 20 31 0d 0a 55 73 65 72 2d 41 67
0130 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30
0140 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e
0150 30 3b 20 57 69 6e 36 34 3b 20 78 36 34 29 20 41
0160 70 70 6c 65 57 65 62 4b 69 74 2f 35 33 37 2e 33
Internet Protocol Version 4 (ip), 20 bytes
Packets: 192 · Displayed: 49 (25.5%) · Dropped: 0 (0.0%)

Figura 14

2.15. Explicación paquete hydra (tráfico)

Este paquete mostrado en la figura 15 es una solicitud HTTP POST que está siendo enviada al recurso `/vulnerabilities /brute/index.php` en el servidor HTTP local que escucha en el puerto 8880. A continuación, se detallan los componentes clave de esta solicitud:

- **Método de solicitud (Request Method)**: POST. Indica que se está enviando información al servidor para ser procesada.
- **URI de solicitud (Request URI)**: `/vulnerabilities/brute/index.php`. Es la ruta del recurso al que se está accediendo en el servidor.
- **Versión de HTTP (Request Version)**: HTTP/1.0. Es la versión del protocolo HTTP utilizada en la solicitud.
- **Host**: `localhost:8880`. Especifica el nombre de host y el número de puerto del servidor al que se está enviando la solicitud.
- **User-Agent**: `Mozilla/5.0 (Hydra)`. Identifica el agente de usuario que está enviando la solicitud. En este caso, indica que el agente de usuario es Hydra.
- **Content-Length**: 48. Indica la longitud del cuerpo de la solicitud en bytes.
- **Content-Type**: `application/x-www-form-urlencoded`. Especifica el tipo de contenido del cuerpo de la solicitud. En este caso, indica que los datos se envían en formato de formulario codificado.
- **Cookie**: `PHPSESSID=t8bqjdau17euck2o36mu31jo17; security=low`. Contiene las cookies enviadas al servidor como parte de la solicitud. En este caso, incluye la cookie PHPSESSID y la cookie de seguridad.

En resumen, este paquete es una solicitud POST que contiene datos en formato de formulario codificado y está dirigida al recurso `/vulnerabilities/brute/index.php` en el servidor local que escucha en el puerto 8880. La solicitud incluye información sobre el agente de usuario Hydra y las cookies PHPSESSID y seguridad.

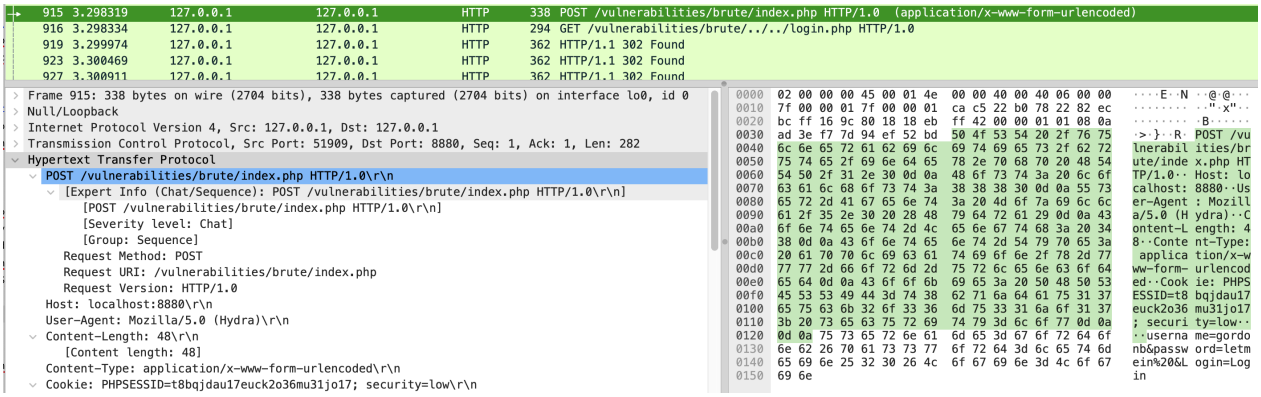
915 3.298319 127.0.0.1 127.0.0.1 HTTP 338 POST /vulnerabilities/brute/index.php HTTP/1.0 (application/x-www-form-urlencoded)
916 3.298334 127.0.0.1 127.0.0.1 HTTP 294 GET /vulnerabilities/brute/../../login.php HTTP/1.0
919 3.299974 127.0.0.1 127.0.0.1 HTTP 362 HTTP/1.1 302 Found
923 3.300469 127.0.0.1 127.0.0.1 HTTP 362 HTTP/1.1 302 Found
927 3.300911 127.0.0.1 127.0.0.1 HTTP 362 HTTP/1.1 302 Found
Frame 915: 338 bytes on wire (2704 bits), 338 bytes captured (2704 bits) on interface lo0, id 0
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 51909, Dst Port: 8880, Seq: 1, Ack: 1, Len: 282
Hypertext Transfer Protocol
POST /vulnerabilities/brute/index.php HTTP/1.0\r\n
[Expert Info (Chat/Sequence): POST /vulnerabilities/brute/index.php HTTP/1.0\r\n]
[POST /vulnerabilities/brute/index.php HTTP/1.0\r\n]
[Severity level: Chat]
[Group: Sequence]
Request Method: POST
Request URI: /vulnerabilities/brute/index.php
Request Version: HTTP/1.0
Host: localhost:8880\r\n
User-Agent: Mozilla/5.0 (Hydra)\r\n
Content-Length: 48\r\n
[Content length: 48]
Content-Type: application/x-www-form-urlencoded\r\n
Cookie: PHPSESSID=t8bqjdau17euck2o36mu31jo17; security=low\r\n
0000 02 00 00 00 45 00 01 4e 00 00 40 00 40 06 00 00
0010 7f 00 00 01 7f 00 00 01 ca c5 22 b0 78 22 82 ec
0020 bc ff 16 9c 80 18 18 eb ff 42 00 00 01 01 08 0a
0030 ad 3e f7 7d 94 ef 52 bd 50 4f 53 54 20 2f 76 75
0040 6c 6e 65 72 61 62 69 6c 69 74 69 65 73 2f 62 72
0050 75 74 65 2f 69 6e 64 65 78 2e 70 68 70 20 48 54
0060 54 50 2f 31 2e 30 0d 0a 48 6f 73 74 3a 20 6c 6f
0070 63 61 6c 68 6f 73 74 3a 38 38 38 30 0d 0a 55 73
0080 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c
0090 61 2f 35 2e 30 20 28 48 79 64 72 61 29 0d 0a 43
00a0 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 34
00b0 38 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a
00c0 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 2d 77
00d0 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63 6f 64
00e0 65 64 0d 0a 43 6f 6f 6b 69 65 3a 20 50 48 50 53
00f0 45 53 53 49 44 3d 74 38 62 71 6a 64 61 75 31 37
0100 65 75 63 6b 32 6f 33 36 6d 75 33 31 6a 6f 31 37
0110 3b 20 73 65 63 75 72 69 74 79 3d 6c 6f 77 0d 0a
0120 0d 0a 75 73 65 72 6e 61 6d 65 3d 67 6f 72 64 6f
0130 6e 62 26 70 61 73 73 77 6f 72 64 3d 6c 65 74 6d
0140 65 69 6e 25 32 30 26 4c 6f 67 69 6e 3d 4c 6f 67
0150 69 6e
····E··N ··@·@···
·····"·x"··
·····B·····
·>·}··R· POST /vu
lnerabil ities/br
ute/inde x.php HT
TP/1.0·· Host: lo
calhost: 8880··Us
er-Agent : Mozill
a/5.0 (H ydra)··C
ontent-L ength: 4
8··Conte nt-Type:
applica tion/x-w
ww-form- urlencod
ed··Cook ie: PHPS
ESSID=t8 bqjdau17
euck2o36 mu31jo17
; securi ty=low··
··userna me=gordo
nb&passw ord=letm
ein%20&L ogin=Log
in

Figura 15

2.16. Mención de las diferencias (tráfico)

Las diferencias a nivel de paquetes generados que se pueden analizar en Wireshark entre Burp Suite, Hydra y Curl incluyen:

Longitud de Bits:

- **Burp Suite**: La longitud de bits puede variar dependiendo de la complejidad de la solicitud y respuesta generada por Burp Suite, que puede incluir datos adicionales como encabezados de seguridad y cookies.
- **Hydra**: Similar a Burp Suite, la longitud de bits puede variar dependiendo de la solicitud y respuesta generada por Hydra, que también puede incluir datos adicionales dependiendo de la configuración del usuario.
- **Curl**: La longitud de bits puede ser más consistente y predecible en comparación con Burp Suite y Hydra, ya que Curl se utiliza principalmente para realizar solicitudes HTTP y HTTPS simples desde la línea de comandos.

Encabezados de Solicitud y Respuesta:

- **Burp Suite**: Puede generar solicitudes con una amplia variedad de encabezados, incluyendo encabezados específicos de Burp Suite para el seguimiento y análisis de solicitudes.
- **Hydra**: Genera solicitudes más simples que pueden tener menos encabezados en comparación con Burp Suite.
- **Curl**: Puede generar solicitudes con encabezados estándar, pero también permite al usuario especificar encabezados personalizados según sea necesario.

Método de Solicitud:

- **Burp Suite**: Puede generar solicitudes con diversos métodos HTTP, como GET, POST, PUT, etc., dependiendo de las acciones realizadas por el usuario en la interfaz de Burp Suite.
- **Hydra**: Generalmente genera solicitudes POST para intentar credenciales en formularios de inicio de sesión.
- **Curl**: Permite al usuario especificar el método de solicitud deseado, como GET o POST, en la línea de comandos.

Contenido del Cuerpo de la Solicitud:

- **Burp Suite**: Puede incluir datos adicionales en el cuerpo de la solicitud, como parámetros de formulario o datos de autenticación.
- **Hydra**: Incluye datos de autenticación en el cuerpo de la solicitud para intentar credenciales en formularios de inicio de sesión.
- **Curl**: Permite al usuario especificar datos de formulario o cualquier otro contenido deseado en el cuerpo de la solicitud mediante la línea de comandos.

En resumen, las diferencias a nivel de paquetes generados entre Burp Suite, Hydra y Curl en Wireshark incluyen la longitud de bits, los encabezados de solicitud y respuesta, el método de solicitud y el contenido del cuerpo de la solicitud, que pueden variar dependiendo de la funcionalidad y la configuración de cada herramienta.

2.17. Detección de SW (tráfico)

Para el caso de los paquetes generador por Burp, en el campo User-Agent: User-Agent: Mozilla/5.0 (Burp Suite) ° se indica que se está utilizando el software Burpsuite.

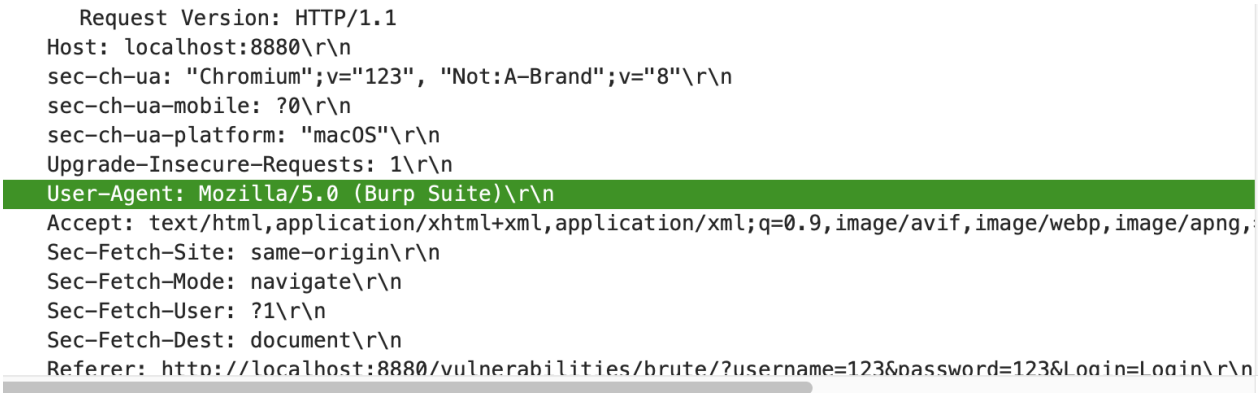

Figura 16

Para el caso de los paquetes generador por Curl, en el campo User-Agent: curl/7.87.0 ° se indica que se está utilizando el software Curl.

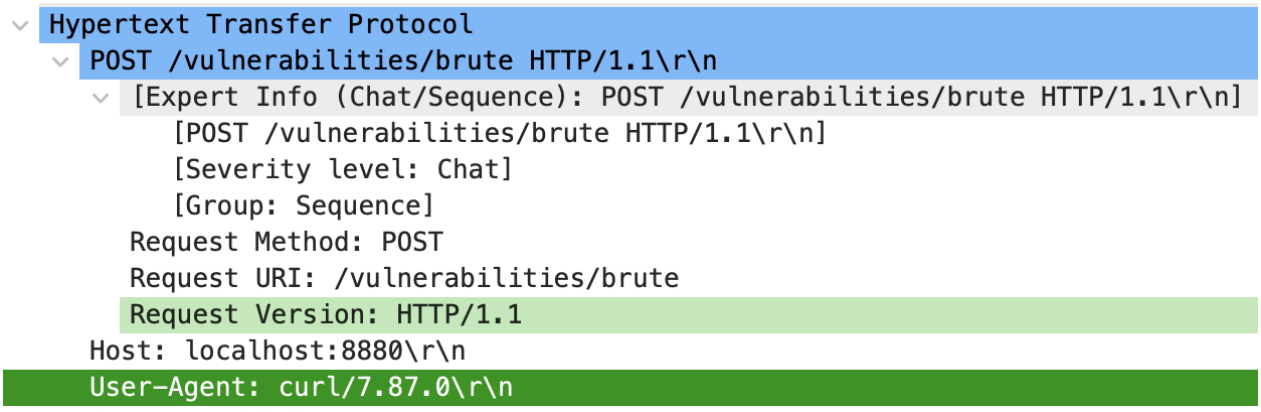

Figura 17

Para el caso de los paquetes generados por Hydra, en el campo User-Agent: Mozilla/5.0 (Hydra) ° se indica que se está utilizando el software Hydra.

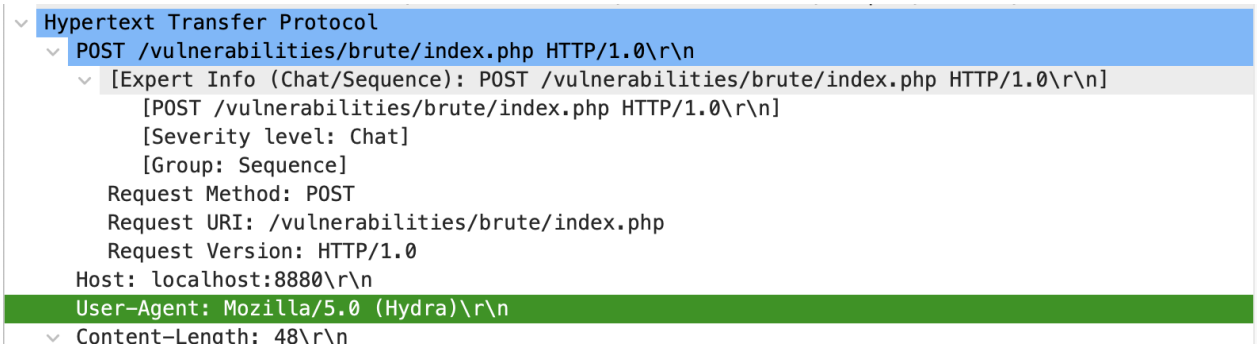

Figura 18