

Informe Laboratorio 2

Sección 1

Alejandro Lagos
e-mail: alejandro.lagos@mail.udp.cl

Septiembre 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. Obtención de consulta a replicar (burp)	3
2.3. Identificación de campos a modificar (burp)	4
2.4. Obtención de diccionarios para el ataque (burp)	4
2.5. Obtención de al menos 2 pares (burp)	6
2.6. Obtención de código de inspect element (curl)	6
2.7. Utilización de curl por terminal (curl)	7
2.8. Demuestra 4 diferencias (curl)	8
2.9. Instalación y versión a utilizar (hydra)	8
2.10. Explicación de comando a utilizar (hydra)	8
2.11. Obtención de al menos 2 pares (hydra)	9
2.12. Explicación paquete curl (tráfico)	9
2.13. Explicación paquete burp (tráfico)	10
2.14. Explicación paquete hydra (tráfico)	11
2.15. Mención de las diferencias (tráfico)	11
2.16. Detección de SW (tráfico)	12

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)

Antes que nada se procede a actualizar el sistema de forma que no hayan problemas imprevistos. Luego de esto, se instala Docker según el comando mas recomendado y se verifica si se instalo correctamente Docker solicitando la versión de este.

```
metroh@METROH-NOTEBOOK-LINUX:~$  
// Actualizar sistema  
  
sudo apt update  
sudo apt upgrade  
  
// Instalar docker  
  
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
  
// Verificar instalación de docker  
  
docker -v
```

Figura 1: Pasos para instalar Docker

Finalmente se levanta el servidor vulnerable con Docker al ejecutar el siguiente comando:

```
metroh@METROH-NOTEBOOK-LINUX:~$  
sudo docker run --rm -it -p 2023:80 vulnerables/web-dvwa
```

Figura 2: Comando para levantar servidor vulnerable DVWA

Como se puede ver se levanta el servidor de forma que accediendo desde el host al puerto 2023 se pueda acceder al puerto 80 del servidor del contenedor, el cual contiene el servidor DVWA.

2.2. Obtención de consulta a replicar (burp)

Accediendo al servidor a través del puerto 2023 del local host, y sabiendo un usuario y contraseña (admin-password), se accedió al apartado "Brute force" de la pagina para poder realizar el ataque a esa dirección. Primero, intentamos con un usuario y contraseña al azar para que pueda captarse el URL correcto. Luego, nuevamente llenamos los campos del formulario pero antes de mandar la petición, activamos una interceptación de esta con el apartado proxy de Burp suite. Luego de mandar la petición logramos captar la consulta a replicar, la cual se ve en la siguiente figura:

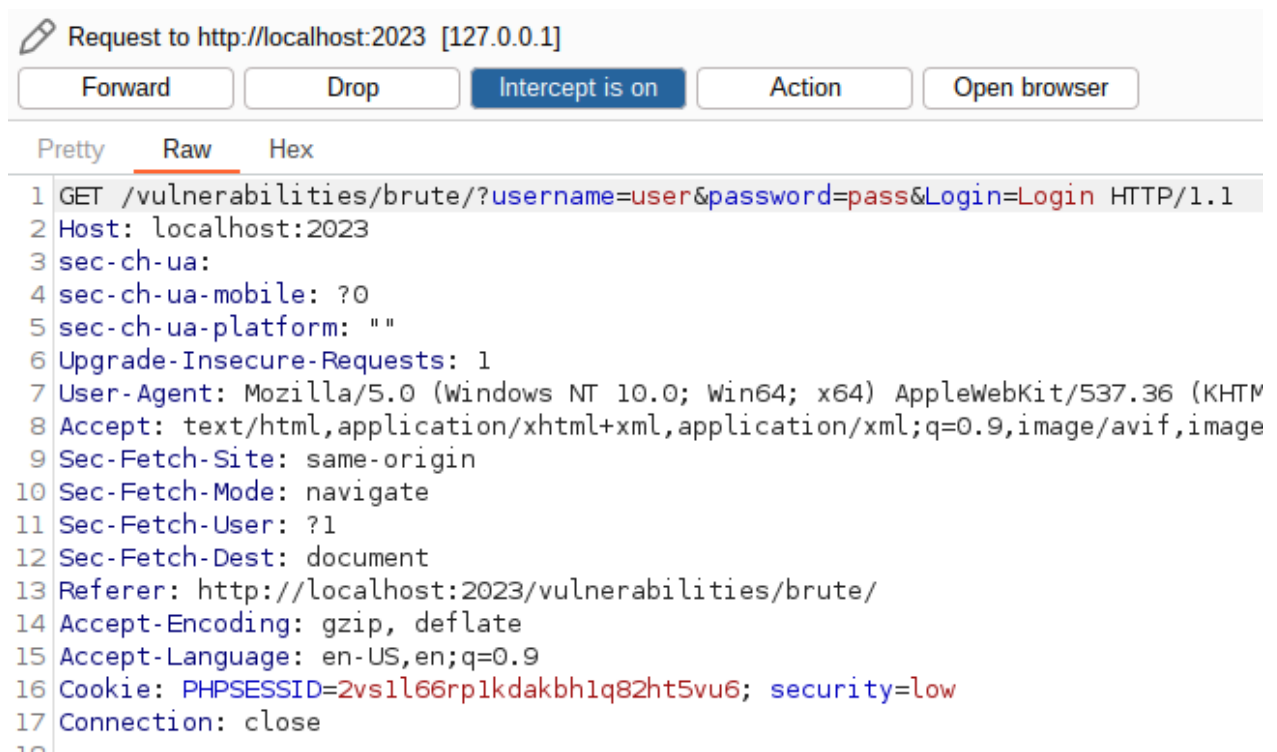


Figura 3: Consulta a replicar

2.3. Identificación de campos a modificar (burp)

Agregamos como payloads el usuario y contraseña del formulario.

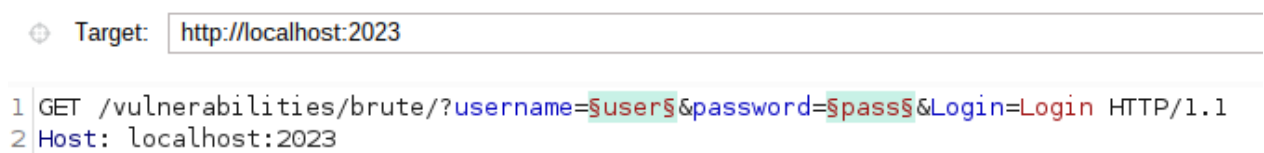


Figura 4: Campos con los que realizaremos el ataque

2.4. Obtención de diccionarios para el ataque (burp)

Ya sabiendo una contraseña valida para el apartado de fuerza bruta, uno se puede dar cuenta que hay un patrón en las imágenes resultantes de una petición valida. Haciendo que que se pueda limitar la cantidad de palabras a utilizar para el apartado usuarios.

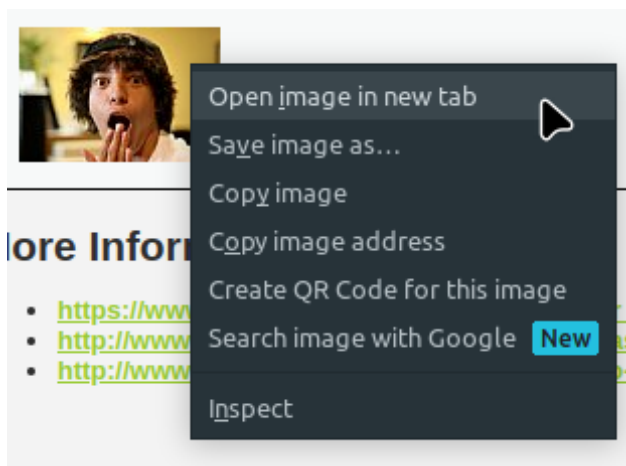


Figura 5: Buscando path de la imagen resultante en caso de éxito

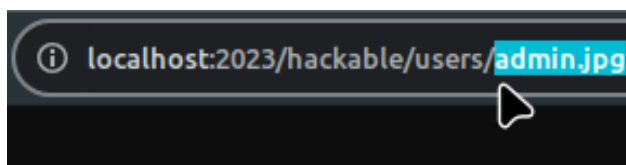


Figura 6: Intentando ver si hay un patrón en el path de las imágenes con respecto a los usuarios

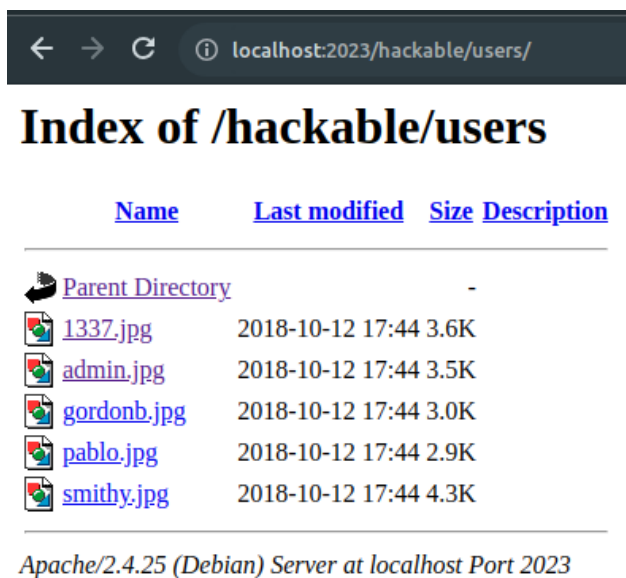


Figura 7: Dándome cuenta de que las imágenes tienen el nombre de los usuarios registrados

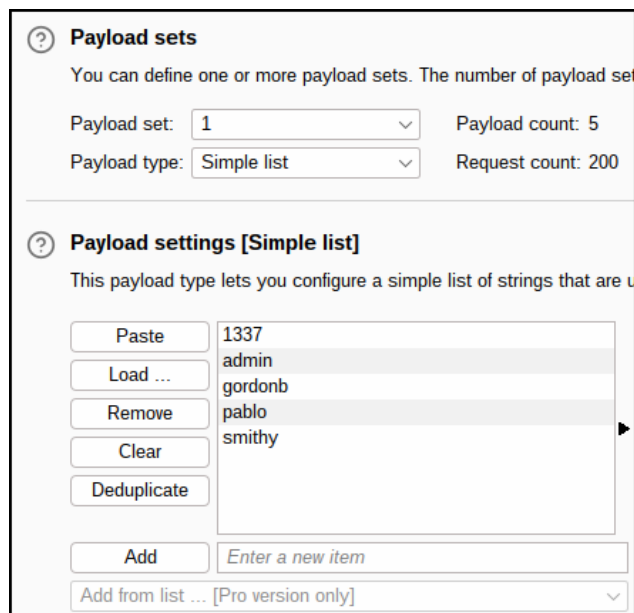


Figura 8: Usando estos nombres para acortar el espacio de búsqueda en el ataque

En cuanto a las contraseñas use una versión ligera de la wordlist Rockyou.txt.

2.5. Obtención de al menos 2 pares (burp)

A continuación se ve el resultado del ataque obteniendo todas las llaves (esto se sabe ya que el largo de los mensajes de respuesta es diferente a los intentos fallidos).

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length
13	gordonb	abc123	200	<input type="checkbox"/>	<input type="checkbox"/>	4745
85	smithy	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4743
82	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4741
199	pablo	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4741
121	1337	charley	200	<input type="checkbox"/>	<input type="checkbox"/>	4739
0			200	<input type="checkbox"/>	<input type="checkbox"/>	4703
1	1337	bobby	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
2	admin	bobby	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
3	gordonb	bobby	200	<input type="checkbox"/>	<input type="checkbox"/>	4703

Figura 9: Obteniendo las 5 llaves del servidor por fuerza bruta

2.6. Obtención de código de inspect element (curl)

Para esta parte se hace una petición con el form, obteniendo el Curl de esta gracias a inspección de elementos en el apartado network.

2.8. Demuestra 4 diferencias (curl)

Las principales diferencias entre los resultados de los 2 Curl es que: - El Curl valido Tiene un mensaje con etiqueta ¡p¡- El Curl invalido Tiene un mensaje con etiqueta ¡pre¡- El Curl valido Tiene una imagen con ¡img¡- El Curl invalido Tiene un mensaje con etiqueta ¡br¡

```
<p>Welcome to the password protected area pablo</p>
```

Figura 14: Etiquetas para Curl valido

```
<pre><br />Username and/or password incorrect.</pre>
```

Figura 15: Etiquetas para Curl invalido

2.9. Instalación y versión a utilizar (hydra)

Instalacion de Hydra

```
metroh@METROH-NOTEBOOK-LINUX:~/Desktop/Cripto Labs/Lab 2/wordlists$
sudo apt update;
sudo apt upgrade;
sudo apt install hydra;
```

Figura 16: Instalación de Hydra

La instalación usada de Hydra es la version 9.2-1ubuntu1

2.10. Explicación de comando a utilizar (hydra)

El comando utilizado para realizar el ataque con Hydra es el siguiente:

```
metroh@METROH-NOTEBOOK-LINUX:~/Desktop/Cripto Labs/Lab 2/wordlists$
hydra -L users.txt -P passwords.txt "http-get-form://172.17.0.1:2023/vulnerabilities/
brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect
.:H=Cookie\; security=low; PHPSESSID=2vs1l66rplkdakh1q82ht5vu6" -I -v
```

Figura 17: Ataque por fuerza bruta con Hydra

El comando proporcionado utiliza la herramienta Hydra para llevar a cabo un ataque de fuerza bruta en un formulario de inicio de sesión web. Se configura para tomar nombres de usuario de un archivo llamado users.txt y contraseñas de passwords.txt. La URL del formulario de inicio de sesión es 172.17.0.1:2023/vulnerabilities/brute/, y se utiliza el método

HTTP GET para enviar los datos. Si la respuesta del servidor indica que las credenciales son incorrectas con el mensaje "Username and/or password incorrect.", Hydra lo registrará. Además, se incluyen cabeceras de solicitud HTTP, en este caso, una cookie PHPSESSID. El comando también instruye a Hydra para ignorar los bloqueos de cuentas por intentos fallidos y proporciona una salida detallada gracias a la opción "verbose". Es importante mencionar que realizar ataques de fuerza bruta sin autorización explícita es ilegal y éticamente cuestionable. Asegúrate de tener los permisos adecuados antes de utilizar esta herramienta.

2.11. Obtención de al menos 2 pares (hydra)

A continuación se muestra evidencia de las llaves obtenidas.

```
metroh@METROH-NOTEBOOK-LINUX:~/Desktop/Cripto Labs/Lab 2/wordlists$ hydra -L users.txt -P passwords.txt
"http-get-form://172.17.0.1:2023/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Us
ername and/or password incorrect.:H=Cookie\; security=low; PHPSESSID=2vs1l66rp1kdakbh1q82ht5vu6" -I -v
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret servic
e 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 2023-09-14 06:45:02
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 5005 login tries (l:5/p:1001), ~313 tries per task
[DATA] attacking http-get-form://172.17.0.1:2023/vulnerabilities/brute/:username=^USER^&password=^PASS^
&Login=Login:Username and/or password incorrect.:H=Cookie\; security=low; PHPSESSID=2vs1l66rp1kdakbh1q8
2ht5vu6
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[2023][http-get-form] host: 172.17.0.1 login: 1337 password: charley
[2023][http-get-form] host: 172.17.0.1 login: admin password: password
[2023][http-get-form] host: 172.17.0.1 login: gordonb password: abc123
[2023][http-get-form] host: 172.17.0.1 login: pablo password: letmein
[2023][http-get-form] host: 172.17.0.1 login: smithy password: password
[STATUS] attack finished for 172.17.0.1 (waiting for children to complete tests)
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-09-14 06:45:05
metroh@METROH-NOTEBOOK-LINUX:~/Desktop/Cripto Labs/Lab 2/wordlists$
```

Figura 18: Llaves obtenidas por ataque de fuerza bruta con Hydra

2.12. Explicación paquete curl (tráfico)

Se hace una petición GET teniendo como target una dirección de este tipo:

http://localhost:2023/vulnerabilities/brute/?username=\$user&password=\$pass&Login=Login
HTTP/1.1

Se envía el paquete desde la dirección IP local 172.17.0.1 desde el puerto 2023 a la dirección IP del contenedor 172.17.0.2 con puerto 80, esto debido a la configuración con la que se levanto el servidor de docker, el cual definio que se redirigieran las solicitudes del puerto 2023 al puerto 80 del contenedor. El largo del Curl valido y Curl invalido solo varia por el tamaño de la contraseña y usuario. En los campos "Accept" del paquete se definen el tipo y formato de respuesta que se desea. Finalmente, se puede ver que se ha mentenido el Cookie de envio.

ip.src == 172.17.0.1 && http						
No.	Time	Source	Destination	Protocol	Length	Info
40		172.17.0.1	172.17.0.2	HTTP	896	GET /vulnerabilities/brute/?username=pablo&password=letmein&Login=Login HTTP/1.1
44	635	172.17.0.1	172.17.0.2	HTTP	894	GET /vulnerabilities/brute/?username=bob&password=trash&Login=Login HTTP/1.1
Hypertext Transfer Protocol						
GET /vulnerabilities/brute/?username=pablo&password=letmein&Login=Login HTTP/1.1\r\n						
Host: localhost:2023\r\n						
Accept-Encoding: deflate, gzip, br, zstd\r\n						
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex						
Accept-Language: en-US,en;q=0.9\r\n						
Cookie: PHPSESSID=1dseb6u43an29c3996o77jj4h6; security=low\r\n						
Proxy-Connection: keep-alive\r\n						
Referer: http://localhost:2023/vulnerabilities/brute/?username=user&password=pass&Login=Login\r\n						
Sec-Fetch-Dest: document\r\n						
Sec-Fetch-Mode: navigate\r\n						
Sec-Fetch-Site: same-origin\r\n						
Sec-Fetch-User: ?1\r\n						
Upgrade-Insecure-Requests: 1\r\n						
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.141 Safari/537.						
sec-ch-ua:\r\n						
sec-ch-ua-mobile: ?0\r\n						
sec-ch-ua-platform: ""\r\n						
\r\n						
[Full request URI: http://localhost:2023/vulnerabilities/brute/?username=pablo&password=letmein&Login=Login]						

Figura 19: Paquete generado por Curl

2.13. Explicación paquete burp (tráfico)

Al igual que el paquete Curl se hace una petición GET al mismo target, teniendo como target una dirección de este tipo:

http://localhost:2023/vulnerabilities/brute/?username=\$user&password=\$pass&Login=Login HTTP/1.1

Se puede observar que se envía desde la dirección IP local a la dirección IP del contenedor de Docker. Cabe destacar que por cada paquete enviado, el puerto de envío y el largo del paquete cambia (Esto ultimo puede ser debido a que los usuarios y contraseñas van cambiando en cada paquete). Se puede ver que en los cambios "Accept" "Accept-Language" se definen la preferencia de formato de respuesta (html) y el lenguaje de respuesta (Ingles). También que el puerto de destino es el 80, esto tiene sentido, ya que, al igual que el paquete Curl, nosotros al dirigir el ataque al puerto 2023, este es redirigido al puerto 80 del contenedor de docker (De esa forma este definido al levantar el servidor vulnerable). Finalmente, también se puede observar la Cookie con la que se hace el ataque en el campo **Cookie**.

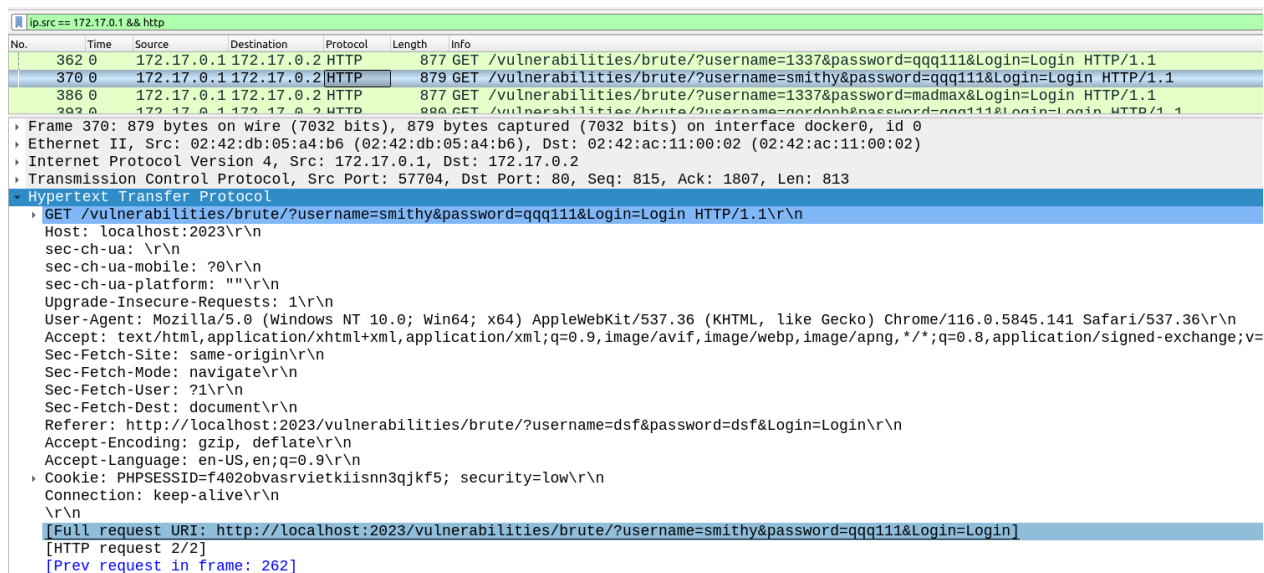


Figura 20: Paquete generado por Burp suite

2.14. Explicación paquete hydra (tráfico)

A diferencia de los dos paquetes anteriores, los paquetes generados por Hydra no llevan mucha información. Solo se incluye el tipo de paquete HTTP (GET) el target de petición, la cookie con la que se quiera mandar, etc.

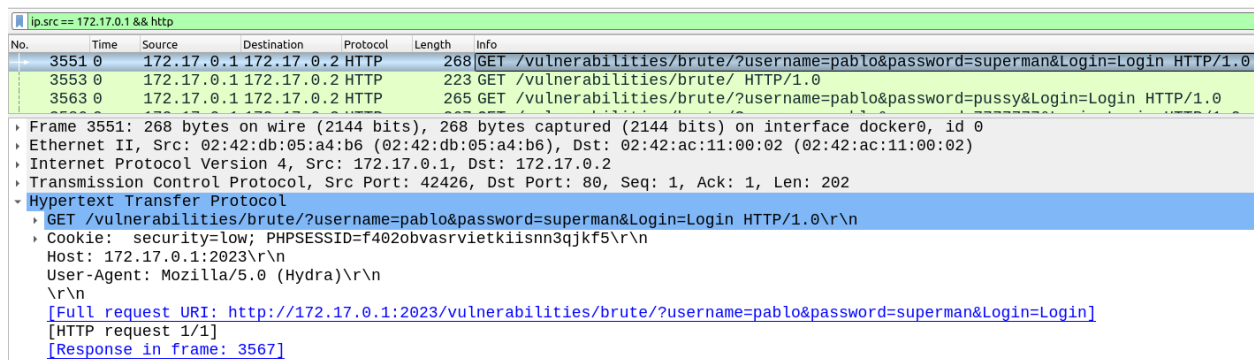


Figura 21: Paquete generado por Hydra

2.15. Mención de las diferencias (tráfico)

La única diferencia importante entre Curl y Burp suite es que este último envía varios paquetes para realizar ataques, en cambio, con Curl solo se envía una vez, en general los paquetes generados por Burp suite y Curl no se diferencian mucho entre sí (solo en el orden en el que se concatenan los campos de las peticiones), ya que estos dos intentan simular el comportamiento de un navegador. En cambio, los paquetes de Hydra son muy distintos

a estos últimos, estos no intentan parecer solicitudes normales de un navegador, ya que no incluyen mucha información en los headers que definan el formato de respuesta, solo intenta enviar los paquetes con la información justa y necesaria para hacer ataques, y así conseguir llaves validas.

2.16. Detección de SW (tráfico)

Para poder detectar cual herramienta esta siendo usada para hacer el ataque basta con ver los campos propios de los paquetes HTTP de las peticiones. Si las definiciones de los campos tipo `.Accept.` estan concatenadas primero, entonces es un ataque con Curl, si no entonces es un ataque con Burp suite. Por otro lado, para saber si es un ataque con Hydra, basta con ver la poca cantidad de campos `Http` definidos en los paquetes.

Conclusiones y comentarios

Finalmente, gracias a las diferentes herramientas de ataque pude ver las diferencias en su funcionamiento en cuento a rendimiento, rapidez, propósito, etc. De la importancia de usar buenos paths para los recursos, y de usar contraseñas con la menor cantidad de patrones posibles, de forma que no sea rentable hacer ataques de fuerza bruta.