

Informe Laboratorio 2

Sección 2

Alejandro Arratia

e-mail: alejandro.arratia@mail.udp.cl

Abril de 2024

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades según criterio de rúbrica	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)	3
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)	9
2.8. Utilización de curl por terminal (curl)	10
2.9. Demuestra 4 diferencias (curl)	16
2.10. Instalación y versión a utilizar (hydra)	17
2.11. Explicación de comando a utilizar (hydra)	17
2.12. Obtención de al menos 2 pares (hydra)	18
2.13. Explicación paquete curl (tráfico)	18
2.14. Explicación paquete burp (tráfico)	19
2.15. Explicación paquete hydra (tráfico)	21
2.16. Mención de las diferencias (tráfico)	22
2.17. Detección de SW (tráfico)	23

1. Descripción de actividades

Utilizando la aplicación web vulnerable DVWA (Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en `vulnerabilities/brute`. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos. Muestre las diferencias observadas en burpsuite.
- Utilice la herramienta cURL, a partir del código obtenido de `inspect elements` de su navegador, para realizar un acceso válido y uno inválido al formulario ubicado en `vulnerabilities/brute`. Indique 4 diferencias entre la página que retorna el acceso válido y la página que retorna un acceso inválido.
- Utilice la herramienta Hydra para realizar un ataque de fuerza bruta contra formulario ubicado en `vulnerabilities/brute`. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos.
- Compare los paquetes generados por hydra, burpsuite y cURL. ¿Qué diferencias encontró? ¿Hay forma de detectar a qué herramienta corresponde cada paquete?

2. Desarrollo de actividades según criterio de rúbrica

2.1. Levantamiento de docker para correr DVWA (dvwa)

Para comenzar en la experiencia, primero se instala docker junto con una imagen de DVWA utilizando el comando

```
docker run -rm -it -p 8880:80 --platform linux/amd64 vulnerables/web-dvwa
```

Que ejecutándolo nos otorga la información mostrada en la Figura 2.1 a continuación:

```

alejandro@omen:~$ docker -v
Docker version 26.0.0, build 2ae903e
alejandro@omen:~$ docker run --rm -it -p 8880:80 --platform linux/amd64 vulnerable
s/web-dvwa
[+] Starting mysql...
[ ok ] Starting MariaDB database server: mysqld.
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reliab
ly 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 <==
[Sun Apr 14 21:10:30.042935 2024] [mpm_prefork:notice] [pid 314] AH00163: Apache/2
.4.25 (Debian) configured -- resuming normal operations
[Sun Apr 14 21:10:30.042975 2024] [core:notice] [pid 314] AH00094: Command line: '
/usr/sbin/apache2'

```

Figura 1: Ejecución de DVWA en docker

Junto a esto verificamos también que la imagen este correctamente siendo ejecutada en docker, para esto usamos *docker ps* en la terminal, lo que nos muestra todas las imágenes y puertos utilizados en docker al momento actual.

```

alejandro@omen:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS
2762c8d79100   vulnerables/web-dvwa               "/main.sh"              54 seconds ago Up 54 seconds
0.0.0.0:8880->80/tcp, :::8880->80/tcp   angry_wilbur

```

Figura 2: Verificación imagen en docker

En la Figura 2 podemos observar que DVWA está efectivamente instalada en docker y se esta ejecutando en el puerto 8880:80.

2.2. Redirección de puertos en docker (dvwa)

Si revisamos la documentación oficial de DVWA, nos indica que el puerto por defecto es el 4280:80, es decir que al momento de ejecutar el comando inicial inmediatamente redireccionamos el puerto al 8880 como comprobamos en la Figura 2, para que no hayan conflictos con otras aplicaciones que puedan estar usando el puerto 80 en nuestro equipo/red.

2.3. Obtención de consulta a replicar (burp)

Abrimos el software Burp Suite, ingresamos a la pestaña de Proxy, activamos el modo interceptar y abrimos el navegador desde la aplicación. Luego, utilizando la dirección IP

2.3 Obtención de Consultas a Replicar (ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA)

obtenida al ejecutar DVWA como se vio en la Figura 2.1 de 172.17.0.2, podemos ingresar a la página de pruebas. En un comienzo ingresamos con el usuario admin con contraseña user y creamos una base de datos para realizar las pruebas, una vez realizado esto nos vamos a la pestaña de Fuerza Bruta como se puede ver en la siguiente Figura 3.

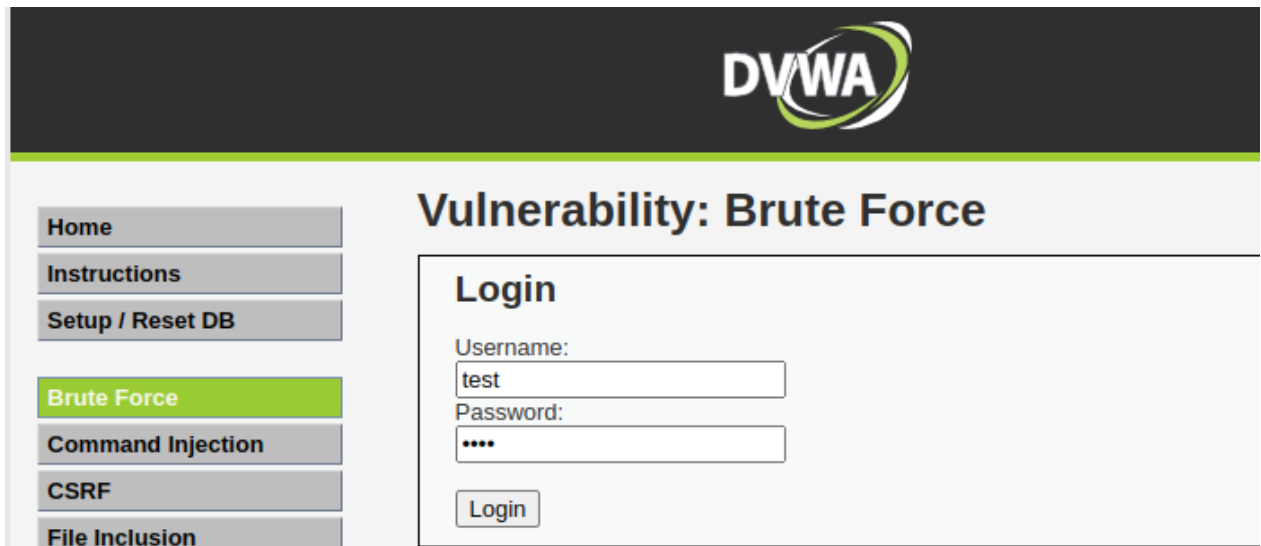


Figura 3: Login Fuerza Bruta DVWA

En esta sección ingresamos inicialmente un usuario de prueba para obtener la consulta que vamos a replicar posteriormente, para esto ingresamos con las credenciales prueba / test, que son erróneas, pero a su vez en Burp Suite podemos observar el tipo de consulta a la que corresponde, como se ve a continuación en la Figura 4:

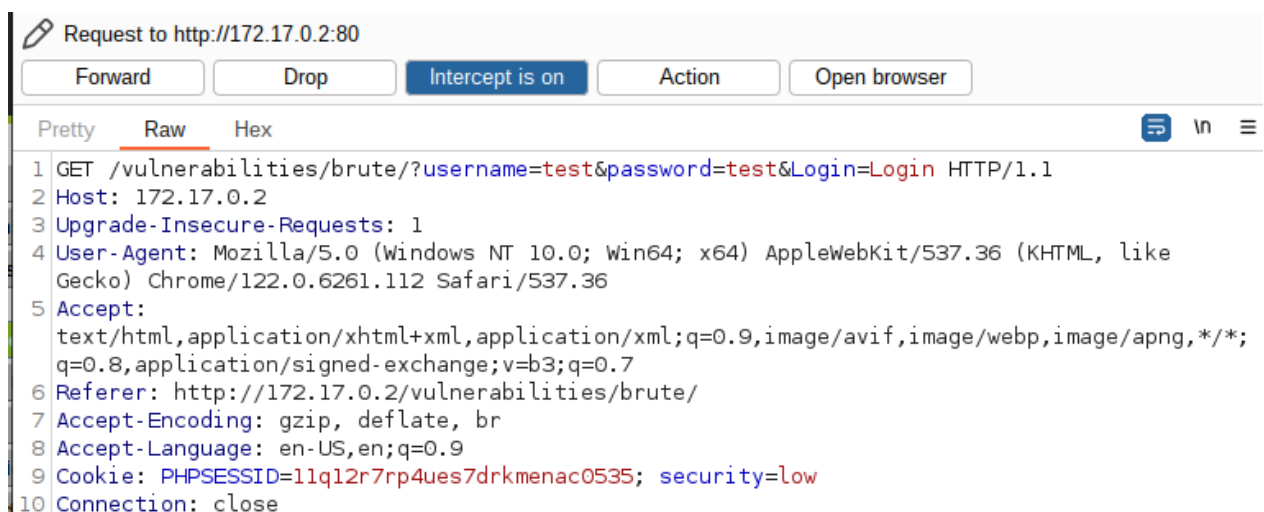


Figura 4: Consulta a replicar

En la Figura 4 podemos ver que es un request tipo GET, y podemos ver en texto plano el username y la password sin cifrado, además se puede ver el token de la cookie correspondiente a la conexión.

2.4. Identificación de campos a modificar (burp)

Copiamos la consulta que queremos replicar y la enviamos a la pestaña Repeater que se encargará de su réplica, desde Repeater se envía a Intruder y definimos como variables o payloads a las credenciales que identificamos en el paso anterior (valores de username y password), seleccionándolas y haciendo click en el botón .add en el menú lateral derecho como se puede ver en la Figura 5.



Figura 5: Variabilización de Payload

2.5. Obtención de diccionarios para el ataque (burp)

Luego ya teniendo variabilizado nuestro payload, debemos cargar un diccionario de usuarios y contraseñas para intentar lograr iniciar sesión utilizando el método de fuerza bruta (cluster bomb) que consiste en enviar todas las combinaciones posibles entre estas variables, hasta que por lo menos 2 sean correctas.

Este diccionario lo obtenemos del siguiente repositorio:

<https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/top-passwords-shortlist.txt>

Que consiste en una lista de los 8 usuarios y 8 contraseñas comunes que consisten en los siguientes valores de la Tabla 1:

Para poder cargar estos datos dentro de Burp Suite ingresamos en la pestaña de "Payload" dentro de "Intruder", seleccionamos la opción de 2 sets de payload y cargamos los archivos txt directamente como se ve en la Figura 6.

Usuario	Contraseña
root	password
admin	123456
test	12345678
guest	abc123
user	querty
adm	111111
administrator	password1
gordonb	1234567

Tabla 1: Usuarios y Contraseñas

Payload sets

You can define one or more payload sets. The number of payload sets depends on the number of payload types available for each payload set, and each payload type can be customized in the Payload settings section.

Payload set: Payload count: 17

Payload type: Request count: 425

Payload settings [Simple list]

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

```

root
admin
test
guest
info
adm
mysql
user

```

22

Figura 6: Carga de archivo a payload set

Correspondiendo el set 1 a los usuarios y el set 2 a las contraseñas.

2.6. Obtención de al menos 2 pares (burp)

Teniendo cargado los payload sets, podemos iniciar el ataque de fuerza bruta haciendo click en "Start Attack." Burp Suite, en donde realiza todas las combinaciones de usuarios y contraseñas. Una vez terminado el ataque podemos ver los resultados en una lista que po-

2.6 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

see las credenciales utilizadas, un tipo de respuesta y el largo de la consulta. Inicialmente observamos la primera respuesta correspondiente a root/password en la Figura 6:

Request ^	Payload 1	Payload 2	Status code	Respons...	Error	Timeout	Length
0			200	1			4703
1	root	password	200	1			4702
2	admin	password	200	3			4741
3	test	password	200	3			4702
4	guest	password	200	4			4703
5	adm	password	200	3			4702
6	user	password	200	3			4703
7	administrator	password	200	3			4702
8	gordonb	password	200	4			4703
9	root	123456	200	4			4702
10	admin	123456	200	3			4703
11	test	123456	200	3			4702
12	guest	123456	200	4			4703
13	adm	123456	200	4			4702
14	user	123456	200	4			4703
15	administrator	123456	200	3			4702

Request	Response
Pretty	Raw
36	</form>
37	<pre>
	Username and/or password incorrect.
	</pre>
38	</div>

Figura 7: Respuesta Incorrecta

En donde se puede ver en Response que el login fue incorrecto. Viendo el largo de las respuestas podemos ver un patrón, que en todas las respuestas incorrectas poseen un largo similar, por lo que si observamos las respuestas con un largo diferente a 4702 - 4703, como por ejemplo la combinación admin / password con un largo de 4741.

2.6 Obtención del Desarrollo de las ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Request	Payload 1	Payload 2	Status code	Respons...	Error	Timeout	Length
0			200	1			4703
1	root	password	200	1			4702
2	admin	password	200	3			4741
3	test	password	200	3			4702
4	guest	password	200	4			4703
5	adm	password	200	3			4702
6	user	password	200	3			4703
7	administrator	password	200	3			4702
8	gordonb	password	200	4			4703
9	root	123456	200	4			4702
10	admin	123456	200	3			4703
11	test	123456	200	3			4702
12	guest	123456	200	4			4703
13	adm	123456	200	4			4702
14	user	123456	200	4			4703
15	administrator	123456	200	2			4702

Request	Response			
Pretty	Raw	Hex	Render	
86			</form>	
87			<p> Welcome to the password protected area admin </p> 	
88			</div>	

Figura 8: Respuesta Correcta admin / password

Podemos ver en la Figura 7 que el login fue exitoso para el caso de admin /password, así que buscamos otra respuesta con un largo similar y encontramos la siguiente:

2.7 Obtención de DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

Filter: Showing all items							
Request ^	Payload 1	Payload 2	Status code	Respons...	Error	Timeout	Length
24	gordonb	12345678	200	6			4703
25	root	abc123	200	5			4703
26	admin	abc123	200	3			4703
27	test	abc123	200	4			4703
28	guest	abc123	200	4			4703
29	adm	abc123	200	4			4703
30	user	abc123	200	4			4703
31	administrator	abc123	200	3			4703
32	gordonb	abc123	200	3			4745
33	root	querty	200	4			4703
34	admin	querty	200	1			4703
35	test	querty	200	4			4703
36	guest	querty	200	3			4703
37	adm	querty	200	3			4703
38	user	querty	200	3			4703
39	administrator	querty	200	3			4703

Request	Response
Pretty	Raw Hex Render
86	</form>
87	<p> Welcome to the password protected area gordonb </p>
88	</div>

Figura 9: Respuesta Correcta gordonb / abc123

En la Figura 8 podemos ver nuevamente que para una combinación con un largo de respuesta diferente es exitoso el login, en este caso para la combinación de gordonb/abc123.

Logrando encontrar entre 72 combinaciones, 2 credenciales reales.

2.7. Obtención de código de inspect element (curl)

Nos conectamos nuevamente al sitio de DVWA mediante la IP obtenida por docker, y entramos a la vista de desarrollador, luego de manera similar a la realizada en el método anterior, ingresamos a la pestaña de fuerza bruta e ingresamos una credencial cualquiera (usamos nuevamente test / test).

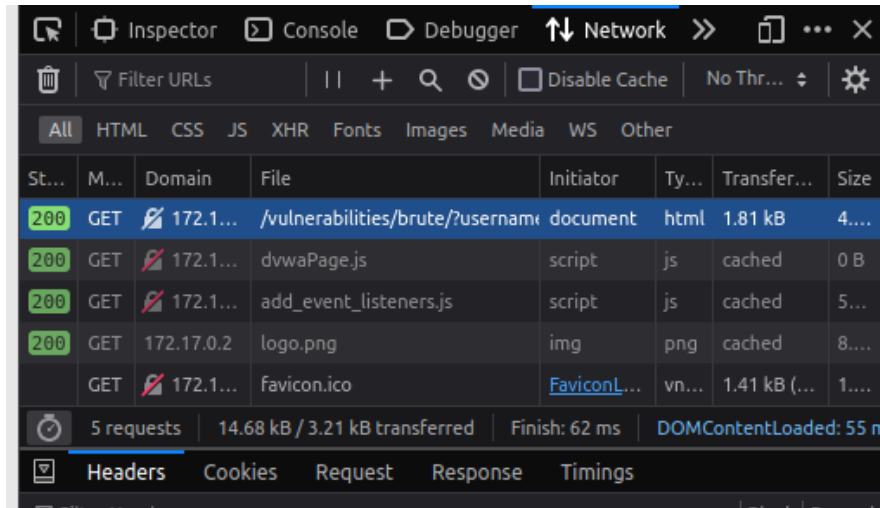


Figura 10: Obtención código curl

Luego como vemos en la Figura 9, al realizar el login fallido, en la pestaña de Network del modo desarrollador aparecen los requests que fueron invocados, y entre ellos esta el correspondiente al login, a este lo seleccionamos con click derecho y copiamos el código curl que podemos observar en la siguiente Figura 11

```
curl 'http://172.17.0.2/vulnerabilities/brute/?username=test&password=test&Login=Login#' --compressed
-H 'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Firefox/124.0'
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8'
-H 'Accept-Language: en-US,en;q=0.5'
-H 'Accept-Encoding: gzip, deflate'
-H 'Connection: keep-alive'
-H 'Referer: http://172.17.0.2/vulnerabilities/brute/'
-H 'Cookie: PHPSESSID=q8ncf0o7o50g9b4msd0duq3is4; security=low'
-H 'Upgrade-Insecure-Requests: 1'
```

Figura 11: Código curl

Que contiene toda la información del request, a que dirección se envía, las cookies utilizadas, las credenciales (test / test), el sistema operativo y el explorador y el tipo de escritura que recibe.

2.8. Utilización de curl por terminal (curl)

Luego enviamos este código curl obtenido mediante la terminal de Linux directamente como se ve a continuación:

```
alejandro@omen:~$ curl 'http://172.17.0.2/vulnerabilities/brute/?username=
test&password=test&Login=Login#' --compressed -H 'User-Agent: Mozilla
/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Firefox
/124.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q
=0.9,image/avif,image/webp,*/*;q=0.8' -H 'Accept-Language: en-US,en;q
```

```
=0.5' -H 'Accept-Encoding: gzip, deflate' -H 'Connection: keep-alive' -
H 'Referer: http://172.17.0.2/vulnerabilities/brute/' -H 'Cookie:
PHPSESSID=q8ncf0o7o50g9b4msd0duq3is4; security=low' -H 'Upgrade-
Insecure-Requests: 1'

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
      charset=UTF-8" />
    <title>Vulnerability: Brute Force :: Damn Vulnerable Web
      Application (DVWA) v1.10 *Development*</title>
    <link rel="stylesheet" type="text/css" href="../../dvwa/
      css/main.css" />
    <link rel="icon" type="image/ico" href="../../favicon.ico
      " />
    <script type="text/javascript" src="../../dvwa/js/dvwaPage
      .js"></script>
  </head>
  <body class="home">
    <div id="container">
      <div id="header">
        
      </div>
      <div id="main_menu">
        <div id="main_menu_padded">
          <ul class="menuBlocks"><li class=""><a
            href="../../">Home</a></li>
          <li class=""><a href="../../instructions.php">Instructions</a></li>
          <li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
          </ul><ul class="menuBlocks"><li class="selected"><a href="../../
            vulnerabilities/brute/">Brute Force</a></li>
          <li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></
            li>
          <li class=""><a href="../../vulnerabilities/csrf/">CSRF</a></li>
          <li class=""><a href="../../vulnerabilities/fi/?page=include.php">File
            Inclusion</a></li>
          <li class=""><a href="../../vulnerabilities/upload/">File Upload</a></li>
          <li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a
            ></li>
          <li class=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>
          <li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (
            Blind)</a></li>
          <li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a
            ></li>
          <li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
          <li class=""><a href="../../vulnerabilities/xss_r/">XSS (Reflected)</a></
            li>
          <li class=""><a href="../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
          <li class=""><a href="../../vulnerabilities/csp/">CSP Bypass</a></li>
        </div>
      </div>
    </div>
  </body>
</html>
```

```

<li class=""><a href="../../../vulnerabilities/javascript/">JavaScript</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../../security.php">DVWA Security</a></li>
<li class=""><a href="../../../phpinfo.php">PHP Info</a></li>
<li class=""><a href="../../../about.php">About</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../../logout.php">Logout</a></li>
</ul>
</div>
</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>
    <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">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>
    <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>

  <br /><br />
</div>
<div class="clear">
</div>
<div id="system_info">
  <input type="button" value="View Help"

```

```

class="popup_button" id='help_button
data-help-url='../vulnerabilities/
view_help.php?id=brute&security=low' )
"> <input type="button" value="View
Source" class="popup_button" id='
source_button' data-source-url='../vulnerabilities/view_source.php?id=
brute&security=low' )"> <div align="
left"><em>Username:</em> admin<br /><em>
Security Level:</em> low<br /><em>
PHPIDS:</em> disabled</div>

</div>
<div id="footer">
<p>Damn Vulnerable Web Application (DVWA)
v1.10 *Development*</p>
<script src='/dvwa/js/add_event_listeners.
js'></script>

</div>
</div>
</body>

```

Listing 1: Respuesta del servidor al comando curl (login incorrecto).

En la respuesta anterior correspondiente al listado 1, corresponde al inicio de sesión fallido debido a que las credenciales no eran las correctas, luego procedemos a realizar el mismo código curl, pero utilizando unas credenciales correctas conocidas de admin / password.

```

alejandro@omen:~$ curl 'http://172.17.0.2/vulnerabilities/brute/?username=
admin&password=password&Login=Login#' --compressed -H 'User-Agent:
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101
Firefox/124.0' -H 'Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/avif,image/webp,*/*;q=0.8' -H 'Accept-Language: en-US,
en;q=0.5' -H 'Accept-Encoding: gzip, deflate' -H 'Connection: keep-
alive' -H 'Referer: http://172.17.0.2/vulnerabilities/brute/' -H '
Cookie: PHPSESSID=q8ncf0o7o50g9b4msd0duq3is4; security=low' -H 'Upgrade
-Insecure-Requests: 1'
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
      charset=UTF-8" />
    <title>Vulnerability: Brute Force :: Damn Vulnerable Web
      Application (DVWA) v1.10 *Development*</title>
    <link rel="stylesheet" type="text/css" href="../../../dvwa/
      css/main.css" />
    <link rel="icon" type="image/ico" href="../../../favicon.ico
      " />
    <script type="text/javascript" src="../../../dvwa/js/dvwaPage
      .js"></script>
  </head>
  <body class="home">

```

```

<div id="container">
  <div id="header">
    
  </div>
  <div id="main_menu">
    <div id="main_menu_padded">
      <ul class="menuBlocks"><li class=""><a
        href="../../../">Home</a></li>
<li class=""><a href="../../../instructions.php">Instructions</a></li>
<li class=""><a href="../../../setup.php">Setup / Reset DB</a></li>
</ul><ul class="menuBlocks"><li class="selected"><a href="../../../
  vulnerabilities/brute/">Brute Force</a></li>
<li class=""><a href="../../../vulnerabilities/exec/">Command Injection</a></
  li>
<li class=""><a href="../../../vulnerabilities/csrf/">CSRF</a></li>
<li class=""><a href="../../../vulnerabilities/fi/?page=include.php">File
  Inclusion</a></li>
<li class=""><a href="../../../vulnerabilities/upload/">File Upload</a></li>
<li class=""><a href="../../../vulnerabilities/captcha/">Insecure CAPTCHA</a
  ></li>
<li class=""><a href="../../../vulnerabilities/sqli/">SQL Injection</a></li>
<li class=""><a href="../../../vulnerabilities/sqli_blind/">SQL Injection (
  Blind)</a></li>
<li class=""><a href="../../../vulnerabilities/weak_id/">Weak Session IDs</a
  ></li>
<li class=""><a href="../../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
<li class=""><a href="../../../vulnerabilities/xss_r/">XSS (Reflected)</a></
  li>
<li class=""><a href="../../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
<li class=""><a href="../../../vulnerabilities/csp/">CSP Bypass</a></li>
<li class=""><a href="../../../vulnerabilities/javascript/">JavaScript</a></
  li>
</ul><ul class="menuBlocks"><li class=""><a href="../../../security.php">DVWA
  Security</a></li>
<li class=""><a href="../../../phpinfo.php">PHP Info</a></li>
<li class=""><a href="../../../about.php">About</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../../logout.php">Logout
  </a></li>
</ul>
    </div>
  </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 />
    </form>
  </div>
</div>

```

```

        <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>
    <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>

        <br /><br />
    </div>
    <div class="clear">
    </div>
    <div id="system_info">
        <input type="button" value="View Help"
            class="popup_button" id='help_button'
            data-help-url='../vulnerabilities/
            view_help.php?id=brute&security=low' )
        "> <input type="button" value="View
            Source" class="popup_button" id='
            source_button' data-source-url='../vulnerabilities/view_source.php?id=
            brute&security=low' )"> <div align="
            left"><em>Username:</em> admin<br /><em>
            >Security Level:</em> low<br /><em>
            PHPIDS:</em> disabled</div>
    </div>
    <div id="footer">
        <p>Damn Vulnerable Web Application (DVWA)
            v1.10 *Development*</p>
        <script src='/dvwa/js/add_event_listeners.
            js'></script>
    </div>
</div>
</body>

```

Listing 2: Respuesta del servidor al comando curl (login exitoso).

Podemos ver en el Listado 2 la respuesta obtenida cuando se utilizan las credenciales correctas al sistema de DVWA.

2.9. Demuestra 4 diferencias (curl)

Entre ambas respuestas se pueden observar algunas claras diferencias entre ellas podemos nombrar:

1)

```
<p>Welcome to the password protected area admin</p>
<pre><br />Username and/or password incorrect.</pre>
```

Dependiendo si es correcto o no el login, el mensaje en el header 1 (h1) es diferente, dandonos una bienvenida o indicando que el usuario es incorrecto.

2)

```

```

Seguido de esto, en el mismo header, en caso de ser correcto el login, muestra una imagen jpg asociada al usuario admin.

3)

```
<p> o <pre>
```

Los mensajes que mostramos en la primera diferencia, también poseen otra peculiaridad, el exitoso esta envuelto en "p" que corresponde a un párrafo de texto normal pero particular para el usuario, mientras que el incorrecto está envuelto en "pre" que corresponde a un mensaje preformateado utilizado para todos los casos incorrectos por igual.

4)

Largo de mensaje

Otra diferencia es el largo o el peso del mensaje, ya que cuando es un login exitoso se le agrega la imagen, el tamaño en bytes debería ser mayor al login incorrecto.

5)

Acceso y/o cookies

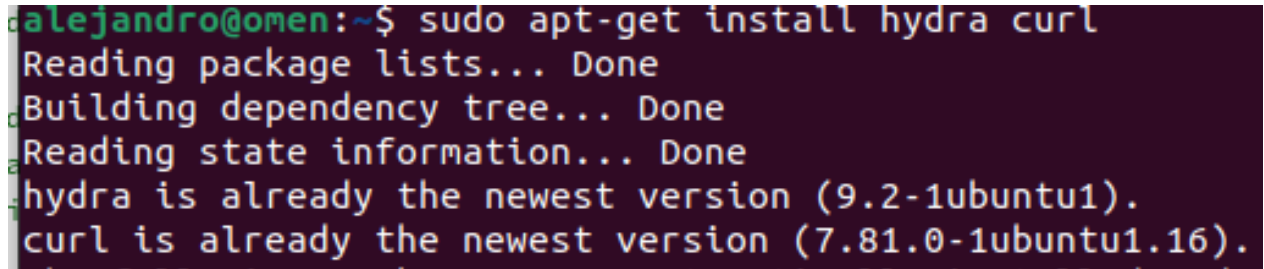
Si bien esta diferencia no sale explícitamente en el código, al ser una sesión correcta podemos suponer que el sitio actualiza sus cookies para mantener la sesión iniciada, mientras que en una sesión incorrecta, la credencial es deshechada.

2.10. Instalación y versión a utilizar (hydra)

Procedemos a instalar Hydra utilizando en terminal el comando:

```
sudo apt-get install hydra curl
```

En nuestro caso, ya teníamos hydra instalada, pero se puede observar en la Figura 12 lo obtenido en la terminal.



```
alejandro@omen:~$ sudo apt-get install hydra curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
hydra is already the newest version (9.2-1ubuntu1).
curl is already the newest version (7.81.0-1ubuntu1.16).
```

Figura 12: Instalación Hydra

En donde podemos observar que ya se encontraba instalada y esta en la última versión 9.2 (c).

2.11. Explicación de comando a utilizar (hydra)

Hydra funciona de manera similar a Burp Suite en cuanto a que ambas aplicaciones pueden realizar un ataque de fuerza bruta, para realizar esto, el esquema básico de un comando de Hydra corresponde a:

```
hydra -L <archivo 1> -P <archivo 2> 'Parámetros URL, Cookies, Filtros'
```

En donde el archivo 1 correspondería a los usuarios o el primer campo del formulario (-L), el archivo 2 a la contraseña o el segundo campo (-P), luego en los parámetros podemos indicar el tipo de request que se quiere hacer (en este caso consisten en GET requests), la ruta URL que corresponde al IP obtenido en docker en localhost, definimos las variables a las cuales vamos a enviar los datos de los archivos para hacer el ataque, agregamos el ID de las cookies para que Hydra pueda llegar hasta el login de "fuerza brutaz no intente ingresar las credenciales en el login principal de DVWA y finalmente agregamos un filtro para que nos muestre en consola sólo las combinaciones correctas, ignorando en las que se recibe el mensaje de login incorrecto que ya conocemos.

Para este caso vamos a utilizar los mismos archivos de texto que se utilizaron para el ataque de fuerza bruta en Burp Suite de 8 usuarios y 8 contraseñas.

2.14 Explicación de Actividades SEGÚN CRITERIO DE RÚBRICA

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.17.0.2	TCP	74	32952 → 80 [SYN] Seq=0 Win=64240 Len=0
2	0.000028133	172.17.0.2	172.17.0.1	TCP	74	80 → 32952 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
3	0.000035815	172.17.0.1	172.17.0.2	TCP	66	32952 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.000055322	172.17.0.1	172.17.0.2	HTTP	586	GET /vulnerabilities/brute/?username=admin
5	0.000067150	172.17.0.2	172.17.0.1	TCP	66	80 → 32952 [ACK] Seq=1 Ack=521 Win=64240 Len=0
6	0.001236780	172.17.0.2	172.17.0.1	HTTP	1888	HTTP/1.1 200 OK (text/html)
7	0.001241988	172.17.0.1	172.17.0.2	TCP	66	32952 → 80 [ACK] Seq=521 Ack=1823 Win=64240 Len=0
8	0.001454067	172.17.0.1	172.17.0.2	TCP	66	32952 → 80 [FIN, ACK] Seq=521 Ack=1823 Win=64240 Len=0
9	0.001469617	172.17.0.2	172.17.0.1	TCP	66	80 → 32952 [FIN, ACK] Seq=1823 Ack=521 Win=64240 Len=0
10	0.001472101	172.17.0.1	172.17.0.2	TCP	66	32952 → 80 [ACK] Seq=522 Ack=1824 Win=64240 Len=0
11	5.077561807	02:42:a4:81:61:0c	02:42:ac:11:00:02	ARP	42	Who has 172.17.0.2? Tell 172.17.0.1
12	5.077539006	02:42:ac:11:00:02	02:42:a4:81:61:0c	ARP	42	Who has 172.17.0.1? Tell 172.17.0.2


```

\t\t\t

```

Figura 14: Tráfico cURL login exitoso

También podemos observar de manera más general el comportamiento muy similar al enviar diferentes ataques por cURL como se ve en la Figura 15, donde podemos ver los paquetes del 1 al 10 son el ataque fallido y del 11 al 20 el ataque exitoso.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.17.0.2	TCP	74	44536 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
2	0.000023027	172.17.0.2	172.17.0.1	TCP	74	80 → 44536 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
3	0.000031059	172.17.0.1	172.17.0.2	TCP	66	44536 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSva
4	0.000050852	172.17.0.1	172.17.0.2	HTTP	581	GET /vulnerabilities/brute/?username=test&passwor
5	0.000066749	172.17.0.2	172.17.0.1	TCP	66	80 → 44536 [ACK] Seq=1 Ack=516 Win=64768 Len=0 TS
6	0.002025903	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
7	0.002031578	172.17.0.1	172.17.0.2	TCP	66	44536 → 80 [ACK] Seq=516 Ack=1806 Win=64128 Len=0
8	0.002218476	172.17.0.1	172.17.0.2	TCP	66	44536 → 80 [FIN, ACK] Seq=516 Ack=1806 Win=64128 Len=0
9	0.002233723	172.17.0.2	172.17.0.1	TCP	66	80 → 44536 [FIN, ACK] Seq=1806 Ack=517 Win=64768 Len=0
10	0.002236026	172.17.0.1	172.17.0.2	TCP	66	44536 → 80 [ACK] Seq=517 Ack=1807 Win=64128 Len=0
11	3.408519446	172.17.0.1	172.17.0.2	TCP	74	44538 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
12	3.408540607	172.17.0.2	172.17.0.1	TCP	74	80 → 44538 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
13	3.408548373	172.17.0.1	172.17.0.2	TCP	66	44538 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSva
14	3.408568951	172.17.0.1	172.17.0.2	HTTP	586	GET /vulnerabilities/brute/?username=admin&passwo
15	3.408581817	172.17.0.2	172.17.0.1	TCP	66	80 → 44538 [ACK] Seq=1 Ack=521 Win=64640 Len=0 TS
16	3.410373334	172.17.0.2	172.17.0.1	HTTP	1888	HTTP/1.1 200 OK (text/html)
17	3.410380259	172.17.0.1	172.17.0.2	TCP	66	44538 → 80 [ACK] Seq=521 Ack=1823 Win=64128 Len=0
18	3.410647775	172.17.0.1	172.17.0.2	TCP	66	44538 → 80 [FIN, ACK] Seq=521 Ack=1823 Win=64128 Len=0
19	3.410694633	172.17.0.2	172.17.0.1	TCP	66	80 → 44538 [FIN, ACK] Seq=1823 Ack=522 Win=64640 Len=0
20	3.410698264	172.17.0.1	172.17.0.2	TCP	66	44538 → 80 [ACK] Seq=522 Ack=1824 Win=64128 Len=0

Figura 15: Comparación login fallido y exitoso wireshark

Son ambos ataques esencialmente iguales, excepto por el tamaño de los paquetes HTTP, que como analizamos con anterioridad, al ser exitoso son de mayor tamaño. Sería simple identificar estos ataques si son realizados en sucesión, ya que cumplen con un formato fijo y regular, y además las direcciones de fuente y destino son siempre las mismas, sólo cambiando la información de las credenciales dentro del mensaje.

2.14. Explicación paquete burp (tráfico)

El tráfico al realizar el ataque con Burp Suite es más complejo que el con cURL, sin embargo podemos identificar los siguientes elementos. Si ordenamos por tamaño de paquete

2.14 Explicación de las Actividades SEGÚN CRITERIO DE RÚBRICA

podemos observar que los dos más grandes corresponden a los ataques exitosos como mencionamos anteriormente.

No.	Time	Source	Destination	Protocol	Length	Info
491	55.934011075	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
500	57.615383183	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
509	59.326043483	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
518	61.067065336	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
527	62.822147687	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
536	64.608527714	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
545	66.424331390	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
554	68.255158419	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
563	70.116296366	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
572	72.382869458	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
581	74.298494114	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
590	76.239555515	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
599	78.205254805	172.17.0.2	172.17.0.1	HTTP	1871	HTTP/1.1 200 OK (text/html)
16	0.203628028	172.17.0.2	172.17.0.1	HTTP	1888	HTTP/1.1 200 OK (text/html)
233	18.292591123	172.17.0.2	172.17.0.1	HTTP	1894	HTTP/1.1 200 OK (text/html)


```

<input type="submit" value="Login" name="Login">\n
\r\n
</form>\r\n
<p>Welcome to the password protected area admin</p>\r\n
</div>\r\n
\r\n
<h2>More Information</h2>\r\n
<ul>\r\n
<li><a href="https://www.owasp.org/index.php/Testina_for_Brute_Force_(OWASP-AT-004)" target=" blank">https://www.owa:

```

Figura 16: Tráfico Burp login exitoso

En la Figura 16 podemos ver el mensaje de login exitoso para el usuario admin / password, que contiene la imagen jpg, y el otro paquete de tamaño 1894 corresponde al otro login exitoso con credenciales gordonb/abc123.

18	0.329801330	172.17.0.1	172.17.0.2	HTTP	717	GET /vulnerabilities/brute/?username=test&password=password
19	0.332905175	172.17.0.2	172.17.0.1	HTTP	1870	HTTP/1.1 200 OK (text/html)
20	0.332922068	172.17.0.1	172.17.0.2	TCP	66	34574 → 80 [ACK] Seq=1952 Ack=5414 Win=64128 Len=0 TSval=1
21	0.485780711	172.17.0.1	172.17.0.2	TCP	74	34588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=
22	0.485843528	172.17.0.2	172.17.0.1	TCP	74	80 → 34588 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460


```

Transmission Control Protocol, Src Port: 34574, Dst Port: 80, Seq: 1301, Ack: 3610, Len: 651
Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=test&password=password&Login=Login HTTP/1.1\r\n
  Host: 172.17.0.2\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/122.0.6261.112 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
  Referer: http://172.17.0.2/vulnerabilities/brute/?username=test&password=test&Login=Login\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  Cookie: PHPSESSID=11q12r7rp4ues7drkmenac0535; security=low\r\n
  Connection: keep-alive\r\n
  \r\n
  [Full request URI: http://172.17.0.2/vulnerabilities/brute/?username=test&password=password&Login=Login]
  [HTTP request 3/5]
  [Prev request in frame: 8]
  [Response in frame: 19]
  [Next request in frame: 38]

```

Figura 17: Protocolo HTTP - Referer

Además podemos ver en la Figura 17, que dentro del protocolo HTTP se puede identificar el Referer: que contiene el username y password test/test que usamos como variables al momento de realizar el ataque y dentro del paquete éstas no varían y son iguales en todo el tráfico de HTTP enviado por Burp Suite.

2.15. Explicación paquete hydra (tráfico)

Vamos a identificar 3 características del tráfico al realizar el ataque de fuerza bruta mediante hydra, primero utilizando la lógica para diferenciar los paquetes exitosos del ataque de Bursp Suite, podemos inferir que los paquetes con un mayor tamaño deberían ser los login exitosos, por lo que si buscamos en el tráfico de Wireshark el paquete con mayor tamaño corresponde al de la Figura 18:

[illegible]

Figura 18: Paquete login exitoso en Wireshark

Donde podemos ver en el mensaje el mensaje de bienvenida junto a la ruta de la imagen jpg que se muestra en el sitio, este fue el correspondiente al usuario gordonb con contraseña abc123. En este caso corresponde a un paquete con protocolo HTTP ya que este maneja la transmisión de la página sin cifrado.

Además podemos obser que el ataque replica el funcionamiento de un paquete TCP común, haciendo un "Three-way handshake como se ve en la Figura 19

No.	Time	Source	Destination	Protocol	Length	Info
1371	1.164980331	172.17.0.2	172.17.0.1	HTTP	1784	HTTP/1.1 200 OK (text/html)
1372	1.164986706	172.17.0.1	172.17.0.2	TCP	66	39100 → 80 [ACK] Seq=152 Ack=4615 Win=62336 Len=0
1373	1.165065502	172.17.0.2	172.17.0.1	TCP	66	80 → 39100 [FIN, ACK] Seq=4615 Ack=152 Win=65024 Len=0
1374	1.165781804	172.17.0.1	172.17.0.2	TCP	66	39046 → 80 [FIN, ACK] Seq=152 Ack=4616 Win=64128 Len=0
1375	1.165810098	172.17.0.2	172.17.0.1	HTTP	4680	HTTP/1.1 200 OK (text/html)
1376	1.165815711	172.17.0.2	172.17.0.1	TCP	66	80 → 39046 [ACK] Seq=4616 Ack=153 Win=65024 Len=0
1377	1.165822318	172.17.0.1	172.17.0.2	TCP	66	39106 → 80 [ACK] Seq=152 Ack=4615 Win=62848 Len=0
1378	1.165871044	172.17.0.2	172.17.0.1	TCP	66	80 → 39106 [FIN, ACK] Seq=4615 Ack=152 Win=65024 Len=0
1379	1.165956366	172.17.0.1	172.17.0.2	TCP	74	39184 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
1380	1.165994778	172.17.0.2	172.17.0.1	TCP	74	80 → 39184 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
1381	1.166015694	172.17.0.1	172.17.0.2	TCP	66	39184 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TS=0
1382	1.166038256	172.17.0.1	172.17.0.2	HTTP	262	GET /vulnerabilities/brute/?username=gordonb&pass=

▶ Frame 1437: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface docker0, id 0
 ▶ Ethernet II, Src: 02:42:a4:81:61:0c (02:42:a4:81:61:0c), 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: 39150, Dst Port: 80, Seq: 199, Ack: 4668, Len: 0
 Source Port: 39150
 Destination Port: 80
 [Stream index: 119]
 [Conversation completeness: Complete, WITH_DATA (31)]

Figura 19: Three-way handshake

2.16 Mención de DESARROLLO (DE) ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

En donde se puede observar en rojo la comunicación entre el cliente y el servidor, el cliente envía un mensaje SYN para avisar que quiere enviar un mensaje, el servidor responde con un SYN+ACK otorgándole permiso, luego el cliente lo envía y el servidor lo confirma.

La tercera característica es que todos estos paquetes HTTP se envían en rápida sucesiones regulares, utilizando las mismas entradas y dirección de destino pero con diferentes combinaciones de credenciales.

2.16. Mención de las diferencias (tráfico)

Los tres tipos de ataque presentan el three-way handshake clásico del protocolo TCP, sin embargo en este punto también presentan una diferencia, en el caso del ataque por cURL, el ataque es singular, se realiza la conexión, se envía el request y finaliza la conexión. Para hydra en cambio, estos ataques los realiza en simultáneo y si bien funcionan de manera similar al cURL, cada ataque posee una conexión independiente, ocultando la relación User-Agent. Para el caso de Burp utiliza el mismo puerto del servidor para realizar más de un ataque, en vez de que cada ataque sea individual.

```
Transmission Control Protocol, Src Port: 44536, Dst Port: 80, Seq: 1, Ack: 1, Len: 515
Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /vulnerabilities/brute/?username=test&password=test&Login=Login
    Request Version: HTTP/1.1
    Host: 172.17.0.2\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:124.0) Gecko/20100101 Firefox/124.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
```

Figura 20: HTTP cURL

Otra diferencia es que en Burp Suite y cURL se puede identificar la línea en que creamos las variables de username y password para agregarlas al payload como se ve en la figura 20 y 21.

```
Frame 4: 715 bytes on wire (5720 bits), 715 bytes captured (5720 bits) on interface docker0, id 0
Ethernet II, Src: 02:42:a4:81:61:0c (02:42:a4:81:61:0c), 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: 34574, Dst Port: 80, Seq: 1, Ack: 1, Len: 649
Hypertext Transfer Protocol
  GET /vulnerabilities/brute/?username=prueba&password=test&Login=Login HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /vulnerabilities/brute/?username=prueba&password=test&Login=Login HTTP/1.1\r\n]
    [GET /vulnerabilities/brute/?username=prueba&password=test&Login=Login HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
```

Figura 21: HTTP Burp

Sin embargo esto no se pudo identificar dentro del protocolo HTTP para Hydra, ya que en

2.17 Detección de SW (tráfico) DE DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

éste simplemente se muestra el usuario y contraseña que se están enviando, no el parámetro de la variable como se ve en la Figura 21.

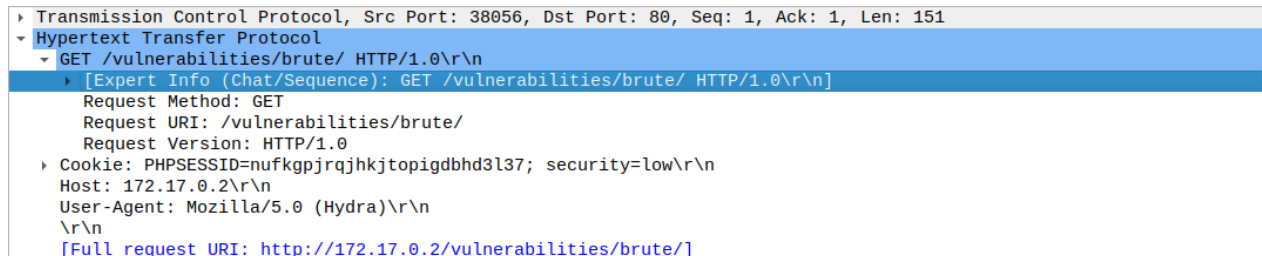


Figura 22: HTTP Hydra

Además podemos notar que para el ataque de Hydra se puede reconocer el navegador Mozilla en este caso, con el que se envió el request, pero en cURL se puede ver el explorador y el sistema operativo junto con su versión

2.17. Detección de SW (tráfico)

Estos diferentes ataques se pueden diferenciar usando la información comentada a través del informe, el paquete cURL siendo el más fácil de detectar ya que son series de 10 paquetes que se envían de manera constante y regular, en ataques individuales, además dentro de ellos se puede identificar el SO y explorador con el cual se realizaron los ataques.

Para el caso de Hydra, si bien son muchos mas paquetes ya que es un ataque masivo, es cuestión de identificar las diferentes conversaciones de TCP y encontrar los patrones ya que se envían utilizando el mismo formato con la dirección de origen y destino, a pesar de que se oculta el SO y el dispositivo por el cuál se está realizando el ataque.

Finalmente en el caso de Burp Suite, los ataques son similares en contenido a los de cURL, pero se envían sin un orden fijo, y a veces utilizando el mismo puerto de entrada para realizar más de un ataque. Sin embargo se puede identificar al Referer con los parámetros de las credenciales con las que se está atacando, además de identificar el origen de destino del ataque junto con su SO y explorador.

Conclusiones y comentarios

En esta experiencia se realizaron tres tipos de ataques de fuerza bruta a la plataforma DVWA abierta en un contenedor de docker.

Los ataques consistieron en un ataque individual por cURL, y dos ataques masivos utilizando una combinación de usuarios y contraseñas comunes mediante Hydra y Burp Suite.

Pudimos ver cómo se realizan estos ataques, identificando los parámetros que se deben enviar para simular las credenciales. En el caso de cURL y Hydra se conseguían en el modo de desarrollador del sitio, mientras que Burp Suite lo generaba automáticamente mediante su herramienta incluida de detección.

Además vimos cómo se pueden llegar a detectar éstos ataques, analizando el tráfico hacia su ruta usando Wireshark, identificando patrones generados en los ataques e información contendia dentro de los paquetes HTTP y TCP.

Estos ataques y detecciones de tráfico, sin embargo, serían completamente diferentes si se tratase de un sistema con una seguridad mayor y el contenido cifrado, además de que las herramientas de Hydra y Burp Suite se pueden configurar con mayor detalle para ocultar su detección si se posee un conocimiento más profundo de ellas.