

PHP y MySQL

Práctica 4

Gonzalo Tudela Chavero

ÍNDICE DE CONTENIDOS

ENUNCIADO	1
Ejercicio 1.....	1
Ejercicio 2	1
DISEÑO DE LA BASE DE DATOS.....	1
CREACION DE LA BASE DE DATOS	2
CREACION DE LA WEB	4

ÍNDICE DE FIGURAS

Ilustración 1 - Esquema relacional de la BD.	1
Ilustración 2 - Esquema relacional de la BD.	2
Ilustración 3 - Ventana Designer con el resultado de la base de datos.	2
Ilustración 4 - Generador de usuarios para nuestra base de datos.....	3
Ilustración 5 - Resultado de la inserción de 50 usuarios.	3
Ilustración 6 - Diagrama de flujo de la aplicación Web.	4

ENUNCIADO

Ejercicio 1

1. Se quiere programar una pequeña Intranet en PHP con una base de datos MySQL para que los usuarios de una organización puedan consultar sus datos personales (nombre, apellido1, apellido2, dirección, DNI, número de teléfono, etc.).
2. El sistema permitirá también cambiar la contraseña cuando el usuario quiera.
3. (Optativo) De manera opcional, cuando el usuario quiera cambiar la contraseña, el sistema le mostrará todas las contraseñas utilizadas para que no las repita.
4. (Optativo) De manera opcional, cuando el usuario cambie de contraseña, el sistema usará la API de la herramienta <https://haveibeenpwned.com/> para no permitir que el usuario utilice contraseñas expuestas en Internet.

Ejercicio 2

Realiza una pequeña prueba de intrusión a la aplicación anterior. El alcance del proyecto será:

- Comprobar si la aplicación presenta vulnerabilidades de inyección de código SQL.
- Comprobar que la aplicación hace un buen uso de las sesiones.

DISEÑO DE LA BASE DE DATOS

El ejercicio nos solicita almacenar los datos personales de cada usuario, así como su login y password y datos personales. Se han omitido las condiciones not null para los apellidos, dni y telf, con el fin de ahorrar tiempo en las pruebas.

El esquema relacional y el entidad-relación son como siguen:

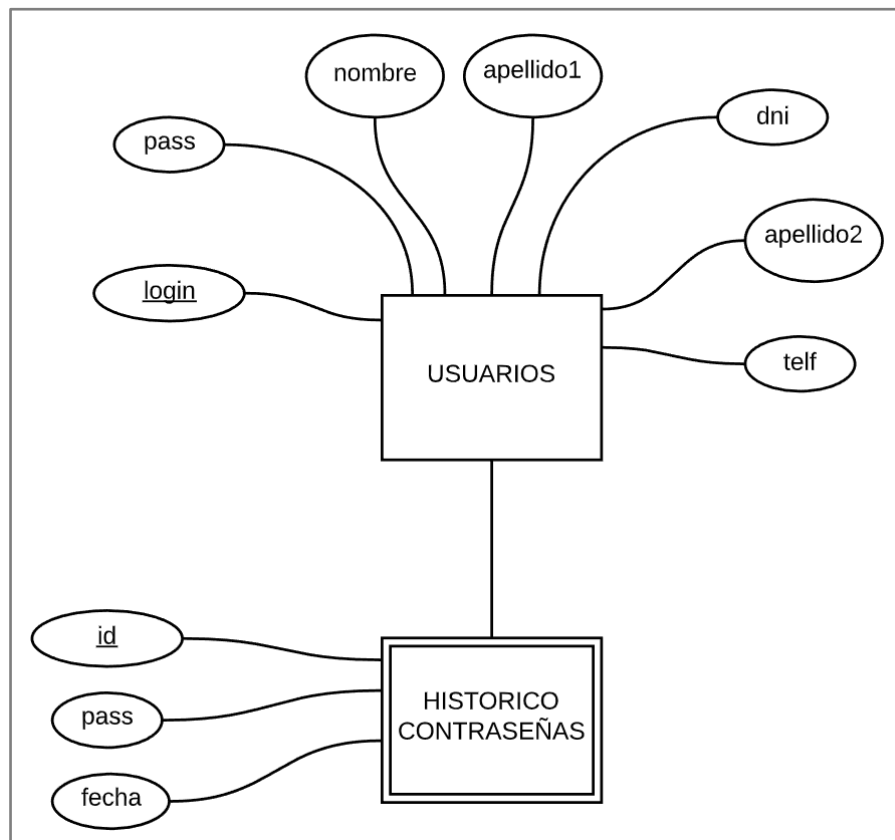


Ilustración 1 - Esquema relacional de la BD.

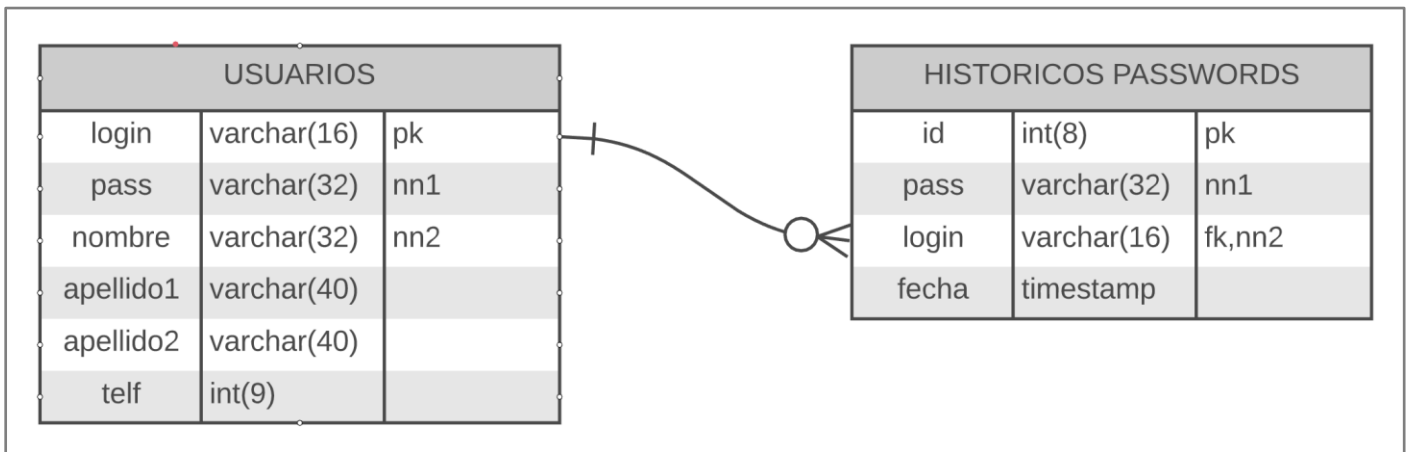


Ilustración 2 - Esquema relacional de la BD.

CREACION DE LA BASE DE DATOS

Para la creación de la base de datos utilizaremos el paquete XAMPP. <https://www.apachefriends.org/es/index.html>
 Para agilizar la creación de la base de datos utilizaremos el entorno grafico PHPMyAdmin.

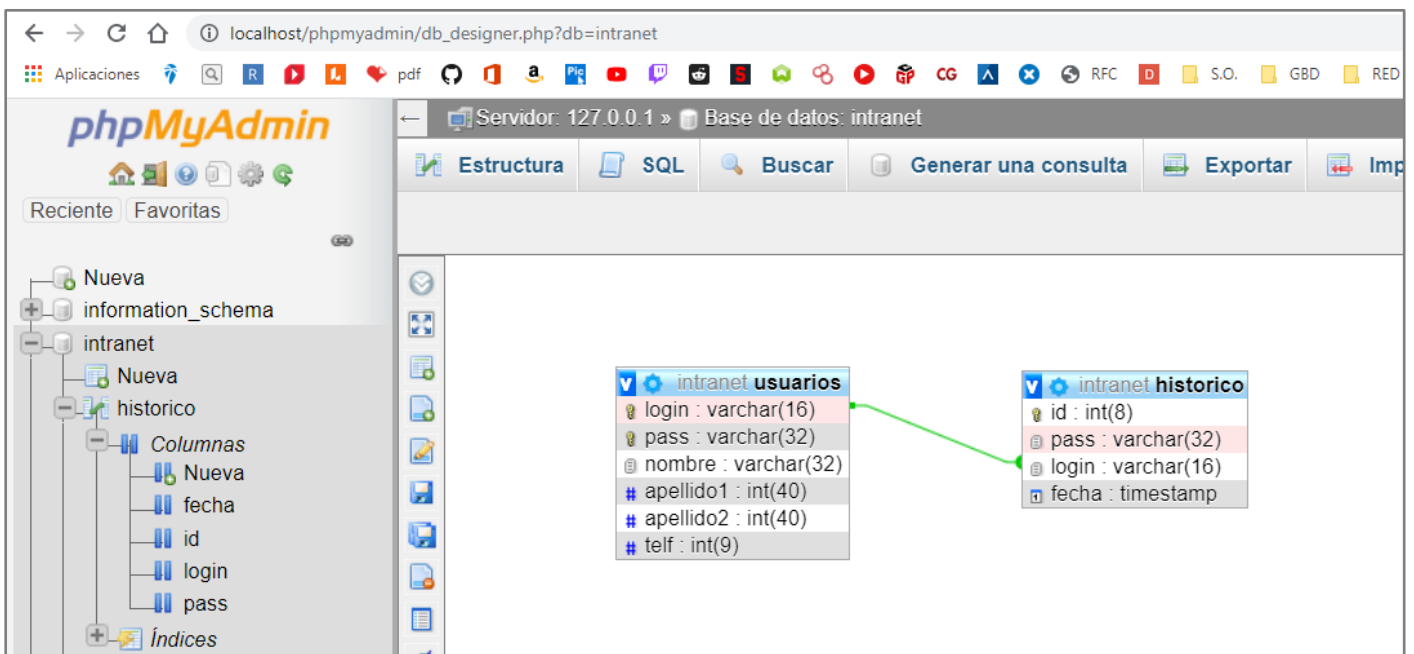


Ilustración 3 - Ventana Designer con el resultado de la base de datos.

Tras la creación de las tablas y relaciones procedemos a introducir varios usuarios mediante una herramienta online en el siguiente enlace <https://mockaroo.com/> la cual nos permitirá crear usuarios de forma aleatoria adaptándolos a la estructura de nuestra base de datos.

The screenshot shows the Mockaroo website interface. At the top, there's a green header with the Mockaroo logo and navigation links. Below the header, a light gray box contains introductory text about generating realistic test data. The main area is a configuration form with three columns: Field Name, Type, and Options. Fields include login (Username), pass (Password), nombre (First Name), apellido1 (Last Name), apellido2 (Last Name), and telf (Phone). At the bottom, there are settings for the number of rows (50), format (SQL), table name (usuarios), and a checkbox to include the create table statement. A green 'Download Data' button is visible.

Ilustración 4 – Generador de usuarios para nuestra base de datos.

La web nos generara un archivo con instrucciones SQL, copiamos su contenido en una consola SQL en PHPMyAdmin y tendremos 50 usuarios listos.

The screenshot shows the PHPMyAdmin interface. On the left, the database structure is visible, including 'information_schema', 'intranet', and 'usuarios'. The main area displays the 'usuarios' table with 50 rows of data. The columns are login, pass, nombre, apellido1, apellido2, and telf. The data is shown in a table format with options to edit, copy, or delete each row.

login	pass	nombre	apellido1	apellido2	telf
aacomb17	F0xPTiO	Ailey	Bertelmot	Acomb	2147483647
bcollingwoodl	m7rYmOYS	Barn	Spowart	Collingwood	2147483647
bfenix1b	qDYM8K8gtOn	Brittni	Shearman	Fenix	2147483647
bpieruccik	7dDNqARt	Braden	Deek	Pierucci	2147483647
clygoew	4qWpzPFX	Cornelius	Tasseler	Lygoe	1043107845
ctullett9	DuQ5feMUF	Chaunce	Goodbarne	Tullett	2147483647
dstrong11	Dbrpp7Wzr	Daniel	Rowatt	Strong	2147483647
ejanssensx	odQtbaAQ	Elke	Furse	Janssens	2147483647
eolesen4	oJQMXDs8	Elmo	Lorant	Olesen	2147483647
fbathoe8	6HYnUm5D7a	Fred	Sissland	Bathoe	1443407060
fdebruinb	JF5GiYxX	Fairfax	Semarke	De Bruin	2147483647
felement3	DlgnMcE	Ferris	Gunson	Element	2147483647
gcloss2	EkD20DvG	Grissel	McKew	Closs	1613282935
gearyi	zOcm5Uc	Giustino	Danilewicz	Eary	2147483647

Ilustración 5 – Resultado de la inserción de 50 usuarios.

CREACION DE LA WEB

El diagrama de flujo inicial para la Web fue el siguiente, pero se decidió eliminar la parte de creación de nuevos usuarios ya que esta debería ser enfocada de otra forma y con otras prestaciones, tal vez acceder desde una cuenta admin o si quisiéramos simular un servicio en el que nos damos de alta, requerir otros datos como el correo...etc.

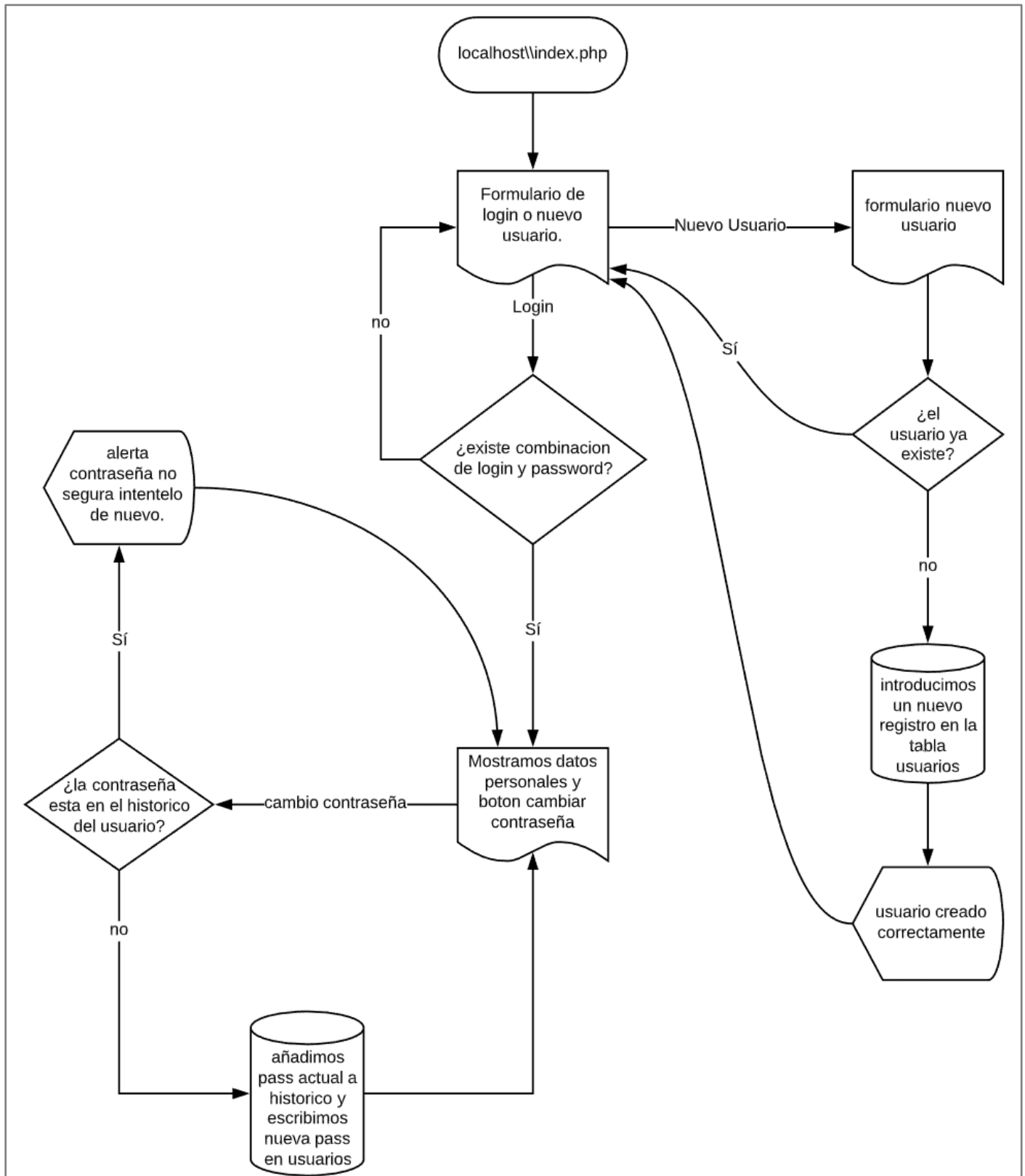


Ilustración 6 - Diagrama de flujo de la aplicación Web.

MODIFICACIONES:

Para cumplir con ciertos objetivos de la rúbrica se modificó el formato de la columna pass por VARBINARY para admitir el formato de los datos que produce la función de MySQL AES_ENCRYPT();

Se eliminó la pagina de registro de usuarios presente en el diagrama de flujo ya que esta característica necesita de un razonamiento sobre el uso que se va a dar, si solo será utilizada por cuentas admin o si tan solo es una herramienta para generar cuentas con las que poder realizar pruebas durante la práctica.

SQLMAP

Para realizar pruebas de inyección de código en los parámetros de nuestra aplicación utilizaremos esta conocida herramienta, antes de empezar vamos a actualizarla con:

```
sqlmap --update
```

Para las pruebas utilizaremos los siguientes parámetros:

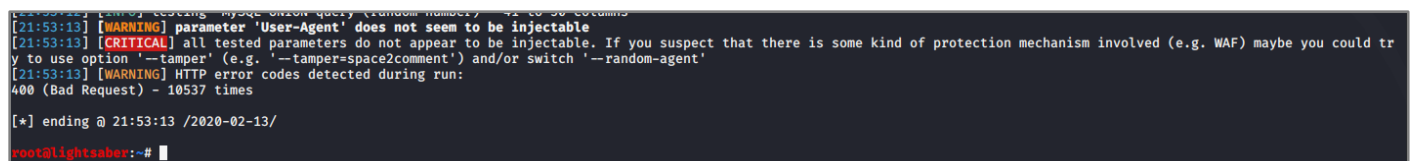
```
--data que indica los parámetros a revisar
--dbms que indica el sistema gestor de base de datos
--cookie indica una cookie para suplantar
```

Las mayoría de pruebas que realiza SQLmap de forma exhaustiva en parámetros como "user agent" en el "header" HTTP u otros que se pueden ver durante la ejecución tienen sentido en entornos reales donde la pagina puede utilizar dichos parámetros de forma controlada a través de una base de datos, pero en esta práctica ninguno de ellos es necesario ya que solo existen un puñado de parámetros por GET para el control de errores, POST para el envío de credenciales y SESSION para mantener la información del usuario que se encuentra "logueado" en la aplicación.

Ahora procedemos a realizar las pruebas en los parámetros de index.php que recoge error mediante (GET).

```
sqlmap http://192.168.1.10/Practica04/web/index.php --data "error=captcha"
--dbms="mysql" --level=5 --risk=3 --dump
```

Los resultados son negativos, detecta que posiblemente tenga un mecanismo de captcha y un WAF protegiendo el sitio.




```
[21:53:13] [WARNING] parameter 'User-Agent' does not seem to be injectable
[21:53:13] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could tr
y to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[21:53:13] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 10537 times
[*] ending @ 21:53:13 /2020-02-13/
root@lightsaber:~#
```

Ilustración 7 - Resultados de la inyección de código en index.php

Pruebas contra login.php:

```
sqlmap http://192.168.1.10/Practica04/web/login.php --data "login=tokio&pass=tokio"
--level=5 --risk=3 --dbms="mysql" --dump
```

```
root@lightsaber:~# sqlmap http://192.168.1.10/Practica04/web/login.php -data "user=tokio&pass=tokio" --level=5 --risk=3 --dbms="mysql" --dump
 {1.4.2.33#dev}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 21:57:13 /2020-02-13/

[21:57:13] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.1.10:80/Practica04/web/index.php?error=captcha'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
[21:57:34] [WARNING] potential CAPTCHA protection mechanism detected
[21:57:34] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS
[21:57:34] [INFO] testing if the target URL content is stable
[21:57:34] [WARNING] POST parameter 'user' does not appear to be dynamic
[21:57:35] [WARNING] heuristic (basic) test shows that POST parameter 'user' might not be injectable
[21:57:35] [INFO] testing for SQL injection on POST parameter 'user'
[21:57:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[21:57:42] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
```

Ilustración 8 - Resultados de la inyección de código en login.php

Los resultados tras el resto de las pruebas contra login.php fueron negativos.

```
[22:10:37] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment')
```

```
[22:10:37] [WARNING] HTTP error codes detected during run:
```

```
400 (Bad Request) - 10536 times
```

```
[*] ending @ 22:10:37 /2020-02-13/
```

```
root@lightsaber:~#
```

Ilustración 9 - Fin de las pruebas de la inyección de código en login.php

Pruebas contra datos.php:

```
sqlmap http://192.168.1.10/Practica04/web/datos.php --data "login=tokio"
--level=5 --risk=3 --dbms="mysql" --dump
```

```

root@lightsaber:~# sqlmap http://192.168.1.10/Practica04/web/datos.php -data "login=tokio" --level=5 --risk=3 --dbms="mysql" --dump --random-agent
[+] http://192.168.1.10/Practica04/web/datos.php
[+] {1.4.2.33#dev}
[+] http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 22:14:31 /2020-02-13/

[22:14:31] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 6.1; fr-FR) AppleWebKit/533.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[22:14:31] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=iri@topokmv...koenp4mth'). Do you want to use those [Y/n] y
[22:14:34] [INFO] testing if the target URL content is stable
[22:14:34] [INFO] target URL content is stable
[22:14:34] [INFO] testing if POST parameter 'login' is dynamic
[22:14:34] [WARNING] POST parameter 'login' does not appear to be dynamic
[22:14:34] [WARNING] heuristic (basic) test shows that POST parameter 'login' might not be injectable
[22:14:35] [INFO] testing for SQL injection on POST parameter 'login'

```

Ilustración 10 - SQLmap detecta que hay una cookie de sesión y solicita modificar el análisis para tener esto en cuenta.

Esto podría haberse hecho de forma manual con el parámetro `--cookie="PHPSESSID=..."`

El resultado tras el análisis es negativo:

```
[22:17:08] [WARNING] parameter 'User-Agent' does not seem to be injectable
[22:17:08] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment')
[22:17:08] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 3337 times

[*] ending @ 22:17:08 /2020-02-13/

root@lightsaber:~#
```

Ilustración 11 - Resultado negativo en las pruebas contra datos.php.

Podemos observar que con el nivel 5 y riesgo 3 se ha comprobado el parámetro User-Agent que es un parámetro que pasa la información del sistema operativo y el navegador que se está utilizando al sitio web.

Ejemplo de datos de user-agent:

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36

Ilustración 12 - Ejemplo de datos que se recoger en la cabecera user-agent.

Pruebas contra nuevopass.php:

```
sqlmap http://192.168.1.10/Practica04/web/datos.php --data
"login=tokio&pass=tokio&cpass=tokio" --level=5 --risk=3 --dbms="mysql" --dump
```

El resultado de las pruebas es que ninguno de los parámetros es inyectable.

```
[22:23:04] [WARNING] parameter 'User-Agent' does not seem to be injectable
[22:23:04] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could tr
y to use option '--tamper' (e.g. '--tamper=space2comment')
[22:23:04] [WARNING] HTTP error codes detected during run:
400 (Bad Request) - 3336 times
[*] ending @ 22:23:04 /2020-02-13/
```

Ilustración 13 - Resultado negativo para los parametros incluida la cookie de sesion.