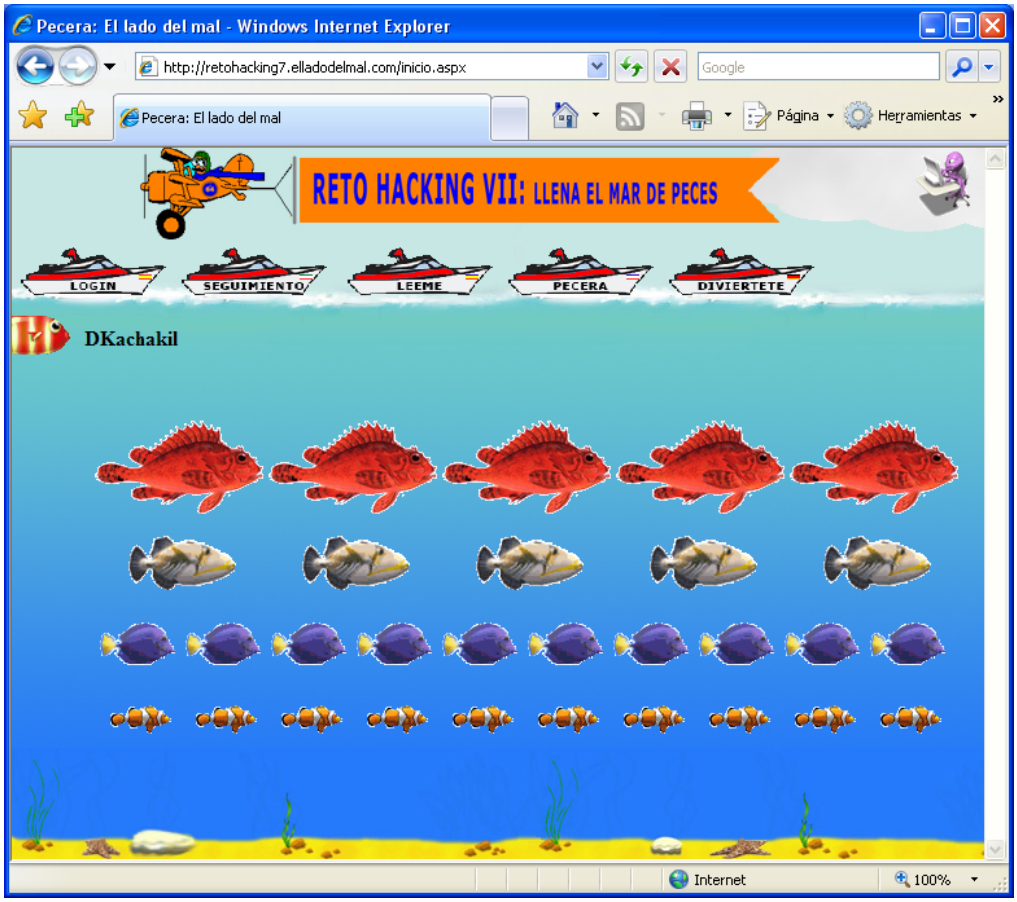
# Solución al Reto Hacking VII de Informática 64

***Abril 2008 © Daniel Kachakil***

## *Introducción*

Este documento describe una solución al Reto Hacking VII de Informática 64, el segundo de la segunda temporada, que se publicó el 4 de abril de 2008 en la siguiente dirección web:

`http://retohacking7.elladodelmal.com`




## *Pistas*

En esta ocasión no teníamos pistas a nuestra disposición, a no ser que a alguien le sirviera la frase que aparecía en el post que anunciaba el comienzo del reto:

> *La fase 1, será como una boda y la fase 2 como un mal sueño, así que nada, salud y suerte a los toreros!*

## *Fase 1: Llenando la pecera*

Como viene siendo habitual en los últimos retos, para acceder a la primera fase teníamos que registrarnos con un nombre de usuario y una dirección de correo válida, donde recibiremos la contraseña correspondiente generada aleatoriamente. Accediendo con estos datos en la página de login, aparecían los siguientes elementos en el menú:

- **Login:** Para darnos de alta o validarnos con nuestro usuario y contraseña.
- **Seguimiento:** Lista de participantes, desde la cual se puede acceder al progreso de cada uno, incluso sin necesidad de autentificarnos.
- **Léeme:** En este punto se nos explica un procedimiento para superar la primera fase, llenando nuestra pecera de peces (eso sí, armados de paciencia).
- **Pecera:** Aquí vemos los peces que tenemos en nuestra pecera, pudiendo añadir un nuevo pez cada 5 minutos. En total "solo" tenemos que añadir 3960 peces…
- **Diviértete:** De momento no tenemos acceso a esta sección. Una vez superada la primera fase, en este punto nos aparecerá la segunda.

Lógicamente, esta primera fase del reto consiste en buscar una forma alternativa de llenar la pecera sin tener que añadir los 3960 peces uno por uno, esperando a que pasen los 5 minutos de rigor entre un pez y el siguiente. De entrada esto nos lleva a pensar en dos posibles alternativas: o bien buscar alguna forma de evitar el retardo y automatizar las casi 4000 peticiones necesarias, o bien encontrar una forma de añadir un número arbitrario de peces en un solo paso.

Resulta curioso que en nuestra pecera aparezca un desplegable que solo contiene el nombre de nuestro usuario, por lo que no sería tan ilógico crear otro usuario para añadirle peces al nuestro, pero vemos que de esta forma no evitamos el retardo y no llegamos a ningún sitio. Por otro lado, las primeras pruebas parecen indicar que dicho campo no es vulnerable a inyección SQL (aunque más adelante veremos que sí lo era).

Por tanto, en principio nos decantaremos por la segunda opción, ya que parece más viable si conseguimos ejecutar alguna instrucción UPDATE o algo por el estilo con el fin de actualizar a nuestro antojo el contador de peces.

Desde la opción ***Seguimiento***, podemos visualizar la pecera de cualquiera de los participantes. Llama la atención que para ello tengamos que superar un pequeño mecanismo de CAPTCHA, escuchando un fichero de audio y escribiendo el texto que contiene. ¿Por qué nos querrán dificultar el acceso a esta información tan irrelevante?

Observando la URL vemos un parámetro de tipo texto (*user*), que se le pasa a la página por GET. Hacemos un par de pruebas típicas para comprobar si dicho parámetro es vulnerable a inyección SQL:

```
verPecera.aspx?user=UsuarioExistente
verPecera.aspx?user=UsuarioExistente' and '1'='1
verPecera.aspx?user=UsuarioExistente' and '0'='1
```

Las dos primeras peticiones muestran la pecera del usuario especificado, mientras que la tercera petición muestra siempre una pecera en blanco. Este pequeño detalle debería ser suficiente para poder extraer toda la información de la base de datos, siempre que tengamos acceso a su catálogo. Aunque todavía no está claro que este sea el camino que vayamos a tomar, ya que para conseguirlo serían necesarias muchas

peticiones al servidor, por lo que tendríamos que buscar la forma de evitar el CAPTCHA. Antes de automatizar nada, por el momento vamos a ver lo que conseguimos averiguar de forma más artesanal.

En primer lugar, intentamos determinar el gestor de bases de datos subyacente, inyectando diferentes instrucciones con variaciones en la sintaxis, objetos de sistema y funciones características de cada uno de los gestores más extendidos. Observamos, por ejemplo, que no fallan funciones como *substr* o uniones con la tabla *dual*, que no se ejecutan varias instrucciones separadas por punto y coma, que la comparación de cadenas diferencia entre mayúsculas y minúsculas, que el doble guión comenta el resto de la consulta, etc. Todo esto nos lleva a pensar que estamos ante una base de datos de Oracle, así que nos documentaremos un poco sobre su catálogo y su sintaxis.

Por cierto, como curiosidad, comentar que era muy fácil llenar la pecera (al menos de forma visual), aunque con ello no conseguíamos nada útil:

```
verPecera.aspx?user=UsuarioInexistente' union select 3960 from dual--
```

En todo caso, no era difícil adivinar algunos de los campos involucrados por simple deducción, aplicando el método de prueba y error. Por ejemplo, probando inyecciones como estas:

```
verPecera.aspx?user=UsuarioInexistente' or usuario='UsuarioExistente'--
verPecera.aspx?user=UsuarioInexistente' or numeropeces>20--
```

Ahora que sabemos el nombre de dos columnas, en lugar de extraer todo el catálogo iremos directamente a localizar el nombre de las tablas que contengan alguna columna llamada *numeropeces* desde ALL_TAB_COLUMNS. Para ello podemos ejecutar la siguiente consulta para determinar el número de tablas involucradas:

```
verPecera.aspx?user=' and 0=1 union select count(*) from ALL_TAB_COLUMNS where
lower(COLUMN_NAME)='numeropeces'--
```

Como era de esperar, vemos que aparece un único pez, por lo que solo nos falta determinar el nombre de la tabla que contiene dicha columna, aprovechando la pecera como si se tratara de un visualizador numérico. Para determinar cualquier valor (que no supere el máximo de 3960), lo haremos simplemente contando el número de peces de cada tipo y multiplicando cada uno por el valor correspondiente a su tamaño (según la leyenda que aparece en la sección ***Léeme*** del menú principal). Ahora veamos cómo obtener la primera letra el nombre de la tabla mediante este procedimiento:

```
verPecera.aspx?user=' and 0=1 union select ascii(substr(TABLE_NAME,1,1)) from
ALL_TAB_COLUMNS where lower(COLUMN_NAME)='numeropeces'--
```

Al inyectar esta consulta vemos que en la pecera resultante aparecen 7 peces de color morado y 10 peces de color naranja, valor que interpretamos como 80, que según las tablas ASCII corresponde a la letra "P". Procedemos entonces con el siguiente valor:

```
verPecera.aspx?user=' and 0=1 union select ascii(substr(TABLE_NAME,2,1)) from
ALL_TAB_COLUMNS where lower(COLUMN_NAME)='numeropeces'--
```

En este caso, contamos 6 peces morados y 9 naranjas, valor que se interpreta como 69, o lo que es lo mismo, la letra "E" en ASCII. Si continuamos con el proceso

hasta el final, determinamos que el nombre de la tabla que buscamos es *"PECES"*. Sin embargo, si intentamos ejecutar subconsultas que involucren dicha tabla veremos que no obtenemos los resultados esperados. Por ejemplo, intentaremos determinar el número de registros que contiene dicha tabla con la siguiente inyección SQL:

```
verPecera.aspx?user=' and 0=1 union select count(*) from PECES--
```

Parece que algo no va bien, ya que a pesar de que la consulta aparenta no tener ningún problema sintáctico, su ejecución falla y, por tanto, la pecera aparece vacía. Esto es debido a que falta un pequeño detalle: prefijar el nombre de la tabla con el propietario de la misma. Ya que tenemos acceso al catálogo, podemos obtener el nombre de dicho propietario utilizando el mismo procedimiento que en el caso anterior, simplemente modificando la columna a la que estamos accediendo:

```
verPecera.aspx?user=' and 0=1 union select ascii(substr(OWNER,1,1)) from
ALL_TAB_COLUMNS where lower(COLUMN_NAME)='numeropeces'--
```

Esta consulta devuelve el valor 83 ("S"). Continuando con el proceso determinamos que el propietario que estamos buscando es *"SYS"*, por lo que esta consulta ya es válida y devolverá el número de participantes registrados en el reto:

```
verPecera.aspx?user=' and 0=1 union select count(*) from SYS.PECES--
```

La verdad es que con un poco de suerte nos podríamos haber ahorrado todo el procedimiento descrito anteriormente, ya que los nombres de todos los objetos que hemos obtenido eran bastante lógicos y perfectamente deducibles. De cualquier forma, ahora que ya tenemos comprobados todos los detalles relativos a la tabla, solamente nos falta encontrar la forma de ejecutar la consulta de actualización.

Toda la documentación que podemos encontrar sobre inyección SQL en Oracle coincide en que de forma general no se pueden ejecutar dos instrucciones diferentes en un solo paso (como sí se podría hacer en SQL Server, separándolas por un punto y coma). Esto solamente es factible para casos muy concretos. Si dejamos de lado las técnicas que involucran otro tipo de vulnerabilidades (como desbordamientos de buffer, etc.), solamente nos quedará la posibilidad de inyectar dicha instrucción dentro de un bloque PL/SQL y todo apunta a que las consultas que hemos estado ejecutando hasta el momento no se encuentran dentro de ninguno de esos bloques.

Aquí es cuando entra en juego ese misterioso desplegable que especifica el usuario destinatario del pez a añadir. Si intentamos las típicas pruebas para determinar si el parámetro es vulnerable, veremos que algo falla y que no conseguimos añadir el pez:

```
UsuarioExistente' and '1'='1
```

Sin embargo, si modificamos el contenido del desplegable con otro tipo de instrucciones observaremos que sí conseguiremos añadir el pez, como por ejemplo el nombre de nuestro usuario concatenado de esta manera (o de cualquier otra que se nos hubiera ocurrido con la finalidad de comprobar si se están procesando las funciones propias del gestor de bases de datos subyacente):

```
'||CHR(ASCII('U'))||CHR(ASCII('s'))|| CHR(ASCII('u'))||'arioExistente
```

Por cierto, para modificar el contenido del cuadro de lista desplegable podemos optar por varias alternativas. Por ejemplo, podemos intercalar un proxy (como Odysseus), o usar algún complemento para el navegador (como Web Developer para FireFox, o la Developer Toolbar para Internet Explorer), etc.

Ahora que hemos asumido que estamos dentro de un bloque de PL/SQL y que hemos conseguido que funcionen ciertas inyecciones propias de Oracle, vamos a intentar inyectar una consulta de actualización. Para ello podemos ponernos en la piel del programador, imaginando que tenemos que implementar la invocación de algún procedimiento dentro de un bloque PL/SQL. En el caso más simple, lo primero que tendremos que hacer es declarar el bloque con instrucciones "BEGIN" y "END" y lo segundo es la invocación del procedimiento dentro de dicho bloque. Por ejemplo:

```
"BEGIN procedimiento('" + parámetro + "'); END;"
```

Por tanto, nuestra inyección deberá cerrar la cadena, el paréntesis y terminar el bloque PL/SQL, comentando el resto de la instrucción con un guión doble (sin olvidar la confirmación de la actualización con la instrucción COMMIT):

```
UsuarioExistente'); update SYS.PECES set NUMEROPECES=10; COMMIT; END;--
```

La inyección mostrada establece en 10 el número de peces de todos los usuarios (de ahí que apareciera ese mensaje de aviso de la sección ***Léeme*** que decía *"… y ten cuidado no te quiten peces"*). Para actualizar únicamente el número de peces de nuestro usuario, evidentemente tendremos que añadir una cláusula WHERE a la UPDATE.

Pues hasta aquí llegaba la primera fase del reto. ¿A que no era tan difícil? Bueno, en realidad tampoco era tan difícil dejarse algún pequeño detalle en cualquiera de los pasos, que es lo que nos suele hacer perder más tiempo del necesario a todos.

## *Fase 2: El mal sueño*

Habiendo superado la primera fase, accedemos a la segunda desde la opción ***Diviértete***. Nos encontramos con una pequeña foto de un girasol, dos campos de texto (primera y segunda clave) y con esta frase de Karl Weierstrass como pista:

> *"Un matematico que no es tambien algo de poeta nunca sera un matematico completo"*

Nada parece indicar el camino a seguir para superar esta fase, por lo que en principio podríamos suponer que se trata de esteganografía. Parece lógico que exista algún texto oculto en la imagen que indique lo que hay que rellenar en ambos campos, así que como primer paso nos descargaremos la imagen del girasol (*nivel2_girasol.jpg*) para ver lo que escondía.

Como no quiero aburrir con las mil pruebas inútiles que podríamos hacerle a la imagen (que contenía varias miniaturas y otros textos que despistaban mucho), pasaremos directamente a la solución, que tampoco era nada trivial a mi parecer.

Bastaba con acceder a los comentarios Exif de la imagen (que en principio son legibles desde el propio sistema operativo), para poder leer un conjunto de pistas que se supone que deberían llevarnos a deducir que teníamos que adivinar el nombre de un personaje y que se trataba de Fibonacci, quien murió en el año 1250 DC, es decir, el mismo año que Federico II, nieto de Federico I "Barbarroja", el mismo que aparecía en el campo del fabricante de la cámara (que son las pistas que aparecían).

A partir de ahí, teníamos que adivinar que **"Fibonacci"** era la palabra que se correspondía con la primera clave. Hecho esto, teníamos que aplicar la sucesión de Fibonacci a la frase que aparecía como pista en la página (el propio girasol también debería habernos llevado a pensar en esta sucesión, ya que sus semillas se ordenan según esa sucesión). Numerando los caracteres de la frase como si de un vector con base cero se tratara y tomando los índices correspondientes a la sucesión de Fibonacci, también empezando en cero (0, 1, 1, 2, 3, 5, 8,…), obteníamos la segunda clave, que como se puede ver, el resultado no esconde ninguna palabra o frase y no tiene ningún sentido, pero esto es lo que obteníamos: **"Unn mta ega"**. Y con eso, ¡reto superado!.

## *Agradecimientos y comentarios*

Hay que reconocer que la primera fase tenía un toque de originalidad y que su resolución era muy factible, ya que gracias a que obteníamos una retroalimentación, podía resolverse paso a paso. En mi caso esa fase me motivó bastante, ya que era la primera vez que aplicaba una inyección SQL en Oracle y eso me obligó a tener que leer todo tipo de documentación y a aprender las peculiaridades de este gestor de bases de datos, ya que a pesar de que la base es la misma, todos ellos tienen sus diferencias.

Sobre la segunda fase, la verdad es que me ha parecido la más chapucera de todos los retos, ya que tenía una complejidad excesiva, era demasiado rebuscada y su resolución no tenía mucha lógica, ni tampoco había ninguna ayuda que indicara si el camino era el correcto o no. Considero que la primera clave sobraba y así su resolución sería mucho más factible. Pero ahí no acabó la cosa, ya que lo peor era que esta fase estaba mal programada por un pequeño detalle. Sí, se trataba de una simple comparación entre dos cadenas de texto, porque no tenía más, pero una mayúscula donde no tocaba hacía que la fase fuera imposible de superar, lo cual demuestra que la programación del reto se terminó de forma precipitada y se publicó sin haberlo probado. Una vez arreglado ese fallo el reto ya se podía superar, pero en el momento de superarlo nuestro usuario desaparecía por completo de la lista debido a otro fallo…

En fin, dejando esos pequeños incidentes de lado (aunque tampoco es la primera vez que digo que las cosas hay que probarlas por el bien de todos), hay que reconocer el esfuerzo y la dedicación que requiere la programación de un reto de estas características, así que ante todo hay que agradecer el trabajo a los de siempre: Chema, Rodol, Alex y a todo aquél que haya participado de cualquier forma, ya que gracias a todos vosotros hemos tenido una nueva oportunidad de aprender jugando, que al final es la utilidad de estos retos. Después de leer esto, ¿te animas a participar en el siguiente?

Saludos,

**Daniel Kachakil**
dani@kachakil.com