# Una solución al 1er Reto Hacking Web de Informática 64

***Enero 2007 © Daniel Kachakil***

## *Introducción*

Este documento describe una posible solución al primer Reto Hacking Web de Informática 64 que se publicó a finales de 2006 en la siguiente dirección web:

http://www.informatica64.com/retohacking/

El reto consistía en acceder a una hipotética zona privada a través de una página de autentificación. Para ello se podía utilizar cualquier elemento de la web, respetando siempre las reglas establecidas para ello.


## *Reglas del juego*

Entrando en la página principal del reto nos encontramos con el siguiente texto que contiene la descripción y las reglas. Literalmente teníamos este texto:

> *Lo que os propongo aquí es un pequeño reto, sobre técnicas de hacking en aplicaciones web. La idea es sencilla, para entrar en una zona privada se nos pide una contraseña, no hay gestión de usuarios, solo se necesita una contraseña. Esta contraseña podría ser cambiada cada semana y enviada a los que tienen privilegios por mail cifrado o dicha de voz, o como nos de la gana, sólo es un juego.*
>
> *La zona privada a la que se tiene acceso con la contraseña está ubicada dentro de una página web que tiene más información y que en este caso está representado por todos*

*los elementos que se ven en la Web. Cualquier elemento que esté en ese mini-web puede ser utilizado para encontrar la palabrita de paso. La Web está en un servidor aislado detrás de un firewall en Informática64, asín que, cualquier escaneo con scanners de vulnerabilidades o similar provocará que se bloqueen ips. Si la tocada de webs es mayor, entonces cortaré el juego y listo calisto.*

*Para pasar el reto solo se necesita "la herramienta más poderosa jamás creada" y... un juguetito que sabréis cual es cuando lleguéis a ese punto, observar, repasar un poco los deberes y pensar. Pensar es la clave para este juego. No es trivial pero ni mucho menos es imposible. No me creería que no lo sacase NADIE. El juego está montado a partir de varias auditorias de seguridad que hemos realizado, de dos en concreto que me gustó mucho resolver y que las he fusionado en una sola, así que si Rodol y yo pudimos, vosotros también.*

*Iré regalando cosas a los que lo vayan sacando (gall..., camisetas, etc...) pero si alguien detecto que se lo han chivoteado (y lo hace solo por ganar la camiseta) entonces le daré la camiseta y lo pondré en la lista de "copiotas".*

*Maligno.*

*PD: Gracias a Alex que se curró la programación del juego.*

### *Pistas*

Los participantes del reto teníamos a nuestra disposición dos pistas accesibles directamente a través de la página principal. Este era el texto de las pistas:

**PISTA 1**
*Lázaro, engañado me has. Juraré yo a Dios que has tú comido las uvas tres a tres.*
*No comí -dije yo-; mas ¿por qué sospecháis eso?*
*Respondió el sagacísimo ciego:*
*¿Sabes en qué veo que las comiste tres a tres? En que comía yo dos a dos y callabas*

**PISTA 2**
*Access denied. Insert contraseØa*

## Solución al reto

Ahora que estamos plenamente ubicados en el contexto, vamos a ponernos manos a la obra, teniendo siempre presentes las reglas y las dos pistas del juego.

### *Análisis inicial*

Basta con echarle un vistazo a la página principal para diferenciar cuatro zonas bastante típicas en la web. Arriba vemos una cabecera, a la izquierda un menú de navegación, en medio está el contenido de la página en la que estamos y a la derecha un poco de publicidad con los próximos eventos.

Lo primero es lo primero, así que le echamos un vistazo al código HTML de la página y observamos que las zonas comentadas anteriormente coinciden con cuatro marcos (frames).

```
<FRAMESET rows="184px, *%" border="0">
  <FRAME src="titulo.htm" border="0">
  <FRAMESET cols="200px, *%, 246px" border="0">
      <FRAME src="indice.htm" border="0">
      <FRAME id="contenido" src="inicio.htm" border="0">
      <FRAME src="publicidad.htm" border="0">
  </FRAMESET>
</FRAMESET>
```

Al estar programada con marcos, normalmente al navegar por la web no veremos la URL de la página en la que estamos, así que tendremos que analizar un poco más a ver dónde nos lleva cada uno de los enlaces del índice.

Si pasamos con el ratón sobre el menú de navegación, observamos que cambia la barra de estado del navegador, mostrando una serie de funciones JavaScript, lo que nos incita a ver el código HTML de la página "indice.htm". De esta forma, vemos lo que pasa cuando pulsamos sobre cualquier elemento del índice:

```
<script type="text/javascript" language="javascript">
    function Entrada() {
        parent.frames[2].location = 'login.aspx';
    }

    function Pista(num) {
        parent.frames[2].location = 'pista.aspx?id_pista='+num;
    }

    function Ganadores() {
        parent.frames[2].location = 'ganadoresReto.aspx';
    }
</script>
```

| ENLACE | ACCIÓN | DESTINO |
|---|---|---|
| Entrada | javaScript:Entrada() | login.aspx |
| Pista 1 | javaScript:Pista(1) | pista.aspx?id_pista=1 |
| Pista 2 | javaScript:Pista(2) | pista.aspx?id_pista=2 |
| Ganadores | javaScript:Ganadores() | ganadoresReto.aspx |

Llegados a este punto solamente nos queda ir analizando cada una de las páginas, a ver si encontramos algún "despiste" que nos permita dar con la solución. Siguiendo con el planteamiento, lo lógico sería intentar buscar un punto de entrada con el que se pueda hacer algo. En principio parece que tenemos dos sitios:

- En la página "**login.aspx**", el campo de texto "**contraseña**"
- En la página "**pista.aspx**", el parámetro "**id_pista**"

## *Página de autentificación*

Empezaremos por lo que parece más evidente, es decir, por la página de autentificación.

Observamos el código HTML y vemos que aparenta ser un tanto complejo a primera vista. Varias funciones y varios ficheros de script con unos parámetros bastante crípticos. Estoy seguro de que más de uno le habrá dedicado unas cuantas horas a analizar esos ficheros sin llegar a ninguna conclusión. Es lógico, puesto que son scripts generados automáticamente por la plataforma ASP.NET 2.0 y no parece muy razonable que alguien se haya dedicado a cambiarlos. La única conclusión que podríamos extraer de ellos es que el campo de la contraseña es obligatorio para enviar el formulario, lo cual no nos aporta gran cosa, aunque también observamos que tampoco hay más validaciones sobre dicho campo (ni longitud, ni rango de caracteres, etc).

Podemos ir haciendo pruebas con ese campo, a ver si logramos entrar directamente, o al menos que nos aparezca algún mensaje de error que nos ayude a conseguir nuestro objetivo. Podemos jugar a adivinar la contraseña, a desbordar el campo, colarnos usando inyección SQL o meter caracteres extraños, pero hagamos lo que hagamos al final parece que siempre nos encontramos con el mismo mensaje: *"Contraseña Incorrecta. Vuelva a intentarlo"*.

Todo indica que la puerta principal parece bastante bien protegida, así que tendremos que intentar colarnos por alguna ventana, es decir, la página de las pistas.

## *Página de pistas*

Recordemos que teníamos un parámetro llamado "id_pista" al que se le pasaba un valor numérico (1 ó 2). ¿Y si metemos 3, 4, 5, etc? ¿nos aparecerá la pista secreta que nos permita superar la prueba? Jeje, más quisiéramos, pero nada más lejos de la realidad… simplemente nos aparece el texto de la pista 1.

Parece bastante razonable deducir que las pistas salen de alguna tabla de una base de datos. También sería lógico deducir que dicha base de datos podría ser la misma que almacena la tabla de usuarios o la contraseña, así que de momento mantendremos estas suposiciones.

Asumiendo que estamos ante una consulta SQL, hacemos algunas pruebas más con el parámetro, por ver si aparece algún mensaje de error que nos ayude:

```
http://www.informatica64.com/retohacking/pista.aspx?id_pista=3
http://www.informatica64.com/retohacking/pista.aspx?id_pista=1'
http://www.informatica64.com/retohacking/pista.aspx?id_pista=2'
http://www.informatica64.com/retohacking/pista.aspx?id_pista=abc
```

En todas las pruebas anteriores (y en muchas otras que se nos podrían ocurrir), aparece el texto de la pista 1. Esto nos lleva a pensar que existe algún tratamiento de errores que establezca por defecto el valor 1 en caso de fallo. Pero cuidado que aún no podemos asegurar que se trate de una sentencia SQL, ya que podría ser un simple bloque IF-ELSE metido directamente en el ASPX.

Para poder seguir con el razonamiento, tenemos que asegurarnos que detrás de ese parámetro hay una instrucción que suponemos similar a esta:

```
"SELECT * FROM pistas WHERE id_pista=" + [parámetro]
```

Por ahora también asumiremos que todo lo que pasamos en el parámetro está concatenándose con la consulta anterior, así que ahora solo nos falta demostrar ambas suposiciones. Para ello vamos a probar añadirle al parámetro una condición tautológica (que siempre es cierta). Como hemos comprobado que cualquier error nos lleva por defecto a la pista 1, probaremos con la pista 2:

```
SELECT * FROM pistas WHERE id_pista=2 AND 1=1
```

Vemos que haciéndolo de esta forma, efectivamente nos aparece el texto de la pista 2, por lo que no se ha producido ningún error y parece que podemos concluir que las suposiciones anteriores eran correctas. De todas formas, comprobaremos que añadiendo una condición falsa, que no devuelva ningún resultado, nos encontraremos de nuevo con el texto de la pista 1:

```
SELECT * FROM pistas WHERE id_pista=2 AND 1=0
```

Estamos ante un caso típico en el que podemos aplicar técnicas de inyección ciega de SQL (blind SQL injection). ¿Ahora se entiende la pista de Lázaro y el ciego?

Ahora el juego consiste en construir una sentencia SQL que nos permita deducir más información, asumiendo que las preguntas que hagamos solo pueden tener dos posibles respuestas: verdadero o falso (es decir, pista 2 o pista 1, respectivamente).

## *Determinando la estructura de la tabla*

Ya sabemos que detrás de todo esto se esconde un motor de base de datos que está ejecutando las sentencias SQL, por lo que ahora trataremos de averiguar cual es, basándonos en las peculiaridades de cada uno de los más extendidos. Para ello usaremos la siguiente sentencia, que nos permitirá saber si existe una determinada tabla (de aquí en adelante sustituiré la parte inicial de la URL por unos puntos suspensivos):

```
http://www.informatica64.com/retohacking/pista.aspx?id_pista=2
… AND (SELECT Count(*) FROM Tabla) >= 0
```

En caso de existir la tabla, no se producirá ningún error y la condición que hemos añadido será verdadera, por lo que aparecerá el texto de la pista 2. En otro caso, se habrá producido un error (la tabla no existe o no tenemos permisos para leerla), por lo que veremos el texto de la pista 1.

La mayoría de los gestores de bases de datos representan la información acerca de las bases de datos que contienen en unas tablas y vistas especiales que son bien conocidas. Por ejemplo, en SQL Server tendríamos "sysobjects", en Access "MSysObjects", etc.

Desgraciadamente, ninguna de las pruebas para acceder al catálogo de la base de datos nos lleva a ningún sitio. Parece que técnicamente es imposible determinar el

nombre de la tabla que buscamos, salvo que usemos la fuerza bruta probando todas las posibles combinaciones. Es una opción que está ahí y podría ser factible, al igual que podríamos haber hecho con la contraseña en la página de entrada, pero de momento vamos a dejarla como ultimísimo recurso. Sigamos con razonamientos más lógicos…

Ya que nos encontramos en un punto muerto, intentaremos sacar más información sobre la tabla de pistas, a ver si tuviera algo interesante. Pongámonos en la mente del programador o diseñador de la base de datos. ¿Cómo llamaríamos a la tabla que contendrá las pistas? Probamos con "pistas" sin éxito y "pista" con éxito. De momento ya tenemos una pequeña pista más: parece que el que diseñó la base de datos opta por poner los nombres de las tablas en singular y no utiliza ningún prefijo.

Por curiosidad, determinamos que la tabla de pistas solamente contiene dos filas, por lo que no vale la pena seguir buscando más por ahí.

```
… AND (SELECT Count(*) FROM pista) = 2
```

Aplicando el mismo criterio, vamos a ver si somos capaces de determinar más información útil. ¿Cómo llamaríamos la tabla de usuarios? Aunque sabemos que no hay gestión de usuarios, probamos con "usuarios", "usuario", "login" y lo que se nos ocurra hasta probar con "contraseña", momento en el que nos salta un error 500 de validación de filtro HTTP del ISA Server. Parece que no le ha gustado la "ñ", así que probamos sustituyéndola por otras letras hasta que casualmente damos con una tabla existente llamada "contrasena". Vale, ha sido cuestión de suerte (basada en la lógica), sí, pero ¡¡ahora ya tenemos el nombre la tabla!!.

¿Cuál es el siguiente paso? Bueno, digo yo que habrá que conocer la estructura los campos y filas que tiene la tabla, así que adelante con ello. Ya sabemos cómo determinar el número de filas que contiene así que vamos a probar con esto:

```
… AND (SELECT Count(*) FROM contrasena) = 1
```

Vemos que solamente hay un registro, lo cual es razonable teniendo en cuenta que en la página de login no había nombre de usuario ni nada por el estilo. Bien, ahora podemos aplicar otras técnicas de inyección de SQL para determinar el número de campos de la tabla y el tipo de datos de cada uno de ellos, aunque tampoco nos servirá de mucho. Otra opción es ir probando nombres de campos, sustituyendo el asterisco de la consulta anterior. Como siempre, si el campo no existe veremos el texto de la pista 1.

```
… AND (SELECT Count(campo) FROM contrasena) >= 0
```

Hacemos varias pruebas sin éxito hasta dar con el campo, que resulta llamarse también "contrasena". Vaya, parece que hoy la suerte está de nuestro lado, aunque tampoco debemos olvidar que los pasos que hemos seguido son bastante lógicos y que el factor humano siempre suele ser importante en estos casos.

## *Obtención de la contraseña*

Ahora que tenemos el nombre de la tabla y el campo, solamente nos queda leer el campo, meterlo en la página de login y ya está (bueno, o eso parece). Para ello, añadiremos una condición que nos permita ir adivinando la contraseña. Asumiendo que la contraseña es de tipo texto, podríamos empezar por algo como esto:

```
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘%’) > 0
```

Lógicamente, la condición anterior se evalúa a cierto. Ahora es cuestión de paciencia ir sacándola haciendo preguntas como un niño pesado: ¿la contraseña empieza por A? ¿y por B? ¿y por C? ¿y por D? Al final daremos con el primer carácter, que resulta ser un cero.

```
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘a%’) > 0
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘b%’) > 0
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘c%’) > 0
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘0%’) > 0
```

Ahora que tenemos el primer carácter, vamos a por el segundo con el mismo procedimiento:

```
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘0a%’) > 0
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘0b%’) > 0
… AND (SELECT Count(*) FROM contrasena WHERE contrasena LIKE ‘0c%’) > 0
```

Acertamos a la tercera, ya que el segundo carácter es una “c”, así que continuamos con el procedimiento, obteniendo un “9”, luego un “4”, etc.

Como parece que la cosa se alarga y hacer esto a mano es un poco pesado, vamos a automatizar el proceso, asumiendo que la contraseña está en formato hexadecimal. Para ello utilizaremos un pequeño programa desarrollado en Visual Basic.NET (en este caso la versión 2.0 de .NET Framework).

```
Dim url As String = "http://www.informatica64.com/retohacking/pista.aspx?id_pista=2"
url += " and (select count(*) from contrasena where contrasena like'{0}%')=1"
Dim resultado As String = "0c94"

For i As Integer = 1 To 30
    For Each caracter As Char In "0123456789abcdef".ToCharArray
        Dim pruebaUrl As String = String.Format(url, resultado + caracter)
        My.Computer.Network.DownloadFile(pruebaUrl, "c:\temp.txt")

        Dim fichero As New IO.FileInfo("c:\temp.txt")
        Dim tamaño As Long = fichero.Length
        fichero.Delete()
        If tamaño < 800 Then
            resultado = resultado + caracter
            Continue For
        End If
    Next
Next
MessageBox.Show(resultado)
```

La idea no era desarrollar un algoritmo eficiente, sino que nos cueste menos escribir el programa que ir haciendo las pruebas a mano. De ahí que esta solución no sea muy elegante, aunque resulta efectiva. Como el texto de la pista 2 es más corto que el de la pista 1, una forma muy rápida de obtener la respuesta era descargarla como archivo y comprobar su tamaño, sin necesidad de leer el contenido ni usar streams.

Un último comentario a tener en cuenta es que esta solución no obtiene el último carácter, ya que si ponemos la contraseña completa y el comodín "%" detrás, no se cumplirá la condición. Por tanto, este último carácter lo sacaremos a mano o modificando ligeramente el código anterior.

Por fin, después de todo este proceso, hemos obtenido el contenido del campo contraseña (omitiré los últimos caracteres):

```
0c9474f07a2a009e8594adde********
```

La contraseña que hemos obtenido parece un poco rara como para hacer esto, pero lo primero que haremos es probar a introducirla tal cual en la página de login, a ver si hay suerte, aunque ya adelanto que de poco sirve intentarlo. Siguiendo con el símil anterior, si la página de login era la puerta y la de pistas era la ventana, ésta última solamente nos ha servido para echarle un vistazo a lo que hay dentro (¡que no es poco!)

## *El paso final*

La contraseña tiene pinta de estar cifrada o codificada de alguna forma. Son exactamente 32 caracteres en hexadecimal, es decir 128 bits. ¿No estará almacenada también en forma de texto en claro? No lo mencioné antes, pero con una consulta de unión ya había comprobado que la tabla no tiene más que dos campos (id, contrasena), así que por ahora poco más podemos hacer.

Vamos a ver qué algoritmos típicos producen un resultado de 128 bits. Podría ser un algoritmo de hash o algún cifrado. Lo más típico sería aplicar una función de hash (algoritmo de dispersión), entre los que tenemos por ejemplo los siguientes algoritmos estándar: MD2, MD4, MD5 o SHA.

Descartamos la familia SHA porque ninguno de ellos genera 128 bits. Nos quedaría la familia de algoritmos Message Digest, en la que todos ellos generan precisamente los 128 bits que buscamos. De los mencionados, en .NET Framework solamente está implementado el MD5, así que asumiremos que se ha usado este.

Si tuviéramos detrás un SQL Server, podríamos pensar en inyectar una sentencia para insertar una nueva fila, aplicando el MD5 a cualquier palabra que usaríamos para entrar. Lo intentamos con una instrucción "insert" detrás de un punto y coma, pero no funciona. Ningún indicio nos ha llevado a pensar que es SQL Server, sino más bien Access (Microsoft Jet), que no permite ejecutar dos instrucciones a la vez.

Descartados todos los supuestos anteriores, solo nos queda intentar reventar la supuesta hash MD5, a ver si obtenemos nuestra tan deseada contraseña. Para ello tenemos a nuestra disposición bastantes programas que aplican algoritmos para romper una hash, como ataques de diccionario o de fuerza bruta. ¿Será este el juguetito que hace falta cuando lleguemos a este punto? Buscando en Google, no resulta complicado encontrar alguno. En mi caso el que utilicé está en el siguiente enlace:

http://www.milw0rm.com/cracker/

Vamos a intentarlo a ver lo que pasa, ¿qué podemos perder? Total, solamente se trata de meter la hash ahí y esperar. Con un poco de suerte, alguien habrá buscado esa cadena antes y nos dará directamente la contraseña. Si no tenemos esa suerte, no pasa nada, ya que podemos apagar el ordenador e irnos a dormir tranquilamente. De nuevo, con un poco de suerte, a la mañana siguiente tendremos nuestra preciada contraseña y así fue. Ahora metemos esa contraseña en la página de login y… ¡reto superado!

Como curiosidad final, si buscamos en Google la hash del reto encontraremos la contraseña directamente, ya que la URL anterior estaba bastante bien posicionada.

## *Agradecimientos*

No quería terminar este documento sin antes agradecer al lector el tiempo invertido en leerlo, esperando que haya resultado ameno y, sobre todo, útil y didáctico.

Por otro lado, quería también mostrar mi agradecimiento a todo el equipo de **Informática 64** que haya participado en el reto, especialmente al gran **Chema Alonso** por esas charlas tan divertidas a las que nos tiene acostumbrados y, cómo no, por el pedazo de libro “Writing Secure Code” (de Microsoft Press) que me ha regalado por haber superado el reto.

Saludos,

**Daniel Kachakil**
dani@kachakil.com

## *Más información*

Hoy en día resulta bastante fácil encontrar mucha información en Internet acerca de los temas aquí tratados, por ello mencionaré solo algunos documentos que podrían servir como guía de iniciación:

http://es.wikipedia.org/wiki/Inyección_SQL
http://es.wikipedia.org/wiki/Blind_SQL_inyection
http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf
http://www.spidynamics.com/whitepapers/Blind_SQLInjection.pdf
http://es.wikipedia.org/wiki/SHA
http://es.wikipedia.org/wiki/MD5