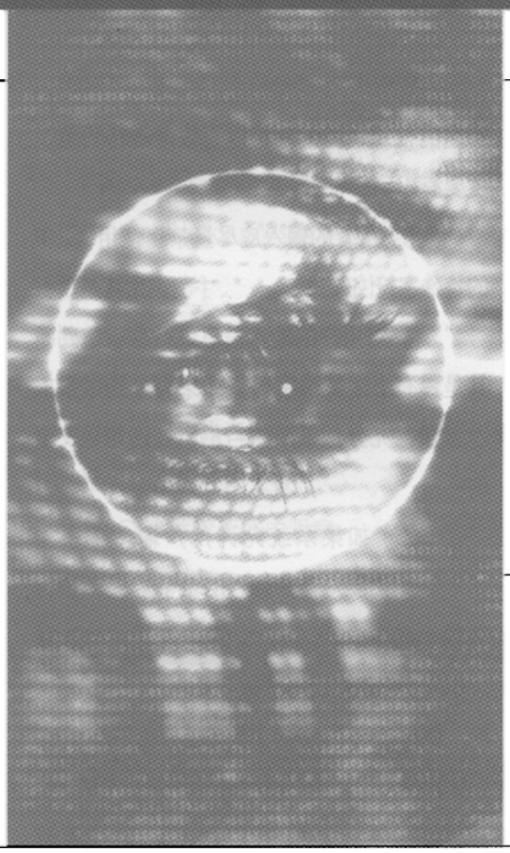

防范合法SQL的入侵和破坏

Keeping Away from Legal SQL's Inbreak and Damage

摘　要： 在数据库应用系统中，由于对某些SQL特殊用法缺乏了解，容易忽略由此而引起的严重安全隐患。本文通过示例演示了合法SQL对系统的欺骗入侵和破坏，同时对该问题进行了分析，最后给出了相应的防范措施。

关键词： SQL 存储过程

石 敏　金 辉（北京装备指挥技术学院软件中心　101416）

1 问题的提出

安全问题存在于系统的方方面面，除了有坚不可摧的系统、网络安全防范手段外，还应该有鲁棒的应用安全防范措施，俗话说“千里之堤，毁于蚁穴”，任何疏忽都有可能对系统造成不可挽回的损失。下面，我们通过一个Delphi示例，来演示由于对SQL使用不当而造成的应用安全漏洞。

图1　用户登录窗口

图1为数据库应用中常见的用户登录管理窗口。在本应用中，系统通过ADO方式(TADOConnection+ TADOQuery)来访问后台SQLServer数据库；UserName和Password为文本输入框(长度限制为不超过数据库中相应字段长度)，用户在此输入用户名和密码，点击‘　　’按钮，进行系统登录。系统内部通过用户输入信息构造数据库查询所用SQL，以判断数据库中是否有与用户输入一致的记录存在，处理函数如下：

```
procedure TfrmGetServerName.bOkClick(Sender: TObject);
  var
    sSQL:string;
  begin
    with DM.ADOQuery do    //数据库访问控件ADOQuery放在DataModule DM中
      begin
        Active:=False;
        SQL.Clear;                    //清空ADOQuery原有SQL
    //构造新的SQL：从users数据表中检索是否有与用户输入一致的记录
    sSQL:='select * from users where username='''+UserName.Text+''' and password='''+PassWord.Text+'''';
        SQL.Add(sSQL);
        Active:=True;                 //执行数据库查询
        if (RecordCount<>0) then      //有记录存在，登录成功
          Application.MessageBox('Logon succeed.', 'Logon', MB_OK);
    ...
      end;
  end;
```

以上程序中：

*'select *from users where username='''+UserName.Text+'''and password='''+PassWord.Text+'''*

为判断用户Login的核心SQL语句，它有安全隐患吗？！

分析

在以下输入过程中，我们将使用Microsoft SQL Server软件套件中的“SQL事件探查器”来进行观察应用实际运行的SQL：

（1）用户在Password文本输入框中输入：' or '1'='1

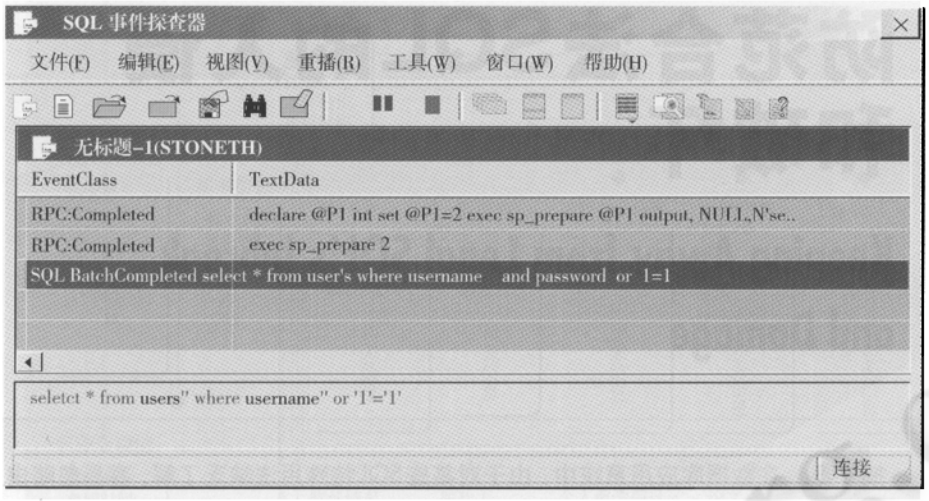

图 2 SQL 事件探查器观察窗口

用户通过输入单引号“'”，使得实际运行的SQL语句变为：

select * from users where username='' and password='' or '1'='1'

结果永真

结果：避开数据库用户表的查询，成功强行登录系统！SQL解释：

λ ANSI/ISO标准规定[1]，SQL字符常数放在单引号'...'中，如果字符串中出现单引号，则用连续两个单引号表示，如' I can''t ';

（2）用户在Password文本输入框中输入：';drop table authors;-

实际运行的SQL语句变为：

select * from users where username='' and password='';drop table authors;-'

结果：成功删除数据表authors!

SQL解释：

λ 一行中可以出现多个SQL语句，SQL语句之间以“；”作为分隔；

λ 两个连字符“-”是 SQL-92 标准的注释指示符，适用于单行或嵌套的注释；

（3）用户在Password文本输入框中输入：';xp_cmdshell('format d:/q/s');-

实际运行的SQL语句变为：

select * from users where username='' and password='';xp_cmdshell('format d:/q/s');-'

结果：成功格式化d盘!

SQL解释：

λ xp_cmdshell为SQLServer下以操作系统命令行解释器方式执行给定的命令字符串。

2 解决办法

以上安全隐患的关键在于，用户通过输入单引号“'”，使得原SQL语句中已有的单引号“'”在语法结构上成为一个合法的字符常数'...'，然后只要保证SQL语法结构的完整性和正确性，通过添加其它SQL，如“' or '1'='1”、“drop table authors;”、“xp_cmdshell('format d:/q/s')”等等来改变原有SQL的语义，就可以为所欲为地入侵系统或进行各种破坏活动。找到了问题的根源，我们就可以有效地进行防范，主要有以下三种方法：

（1）所有的输入应尽量禁止使用SQL中的特殊字符(如'、%、_、*、[等)，尤其是单引号“'”。对用户的输入信息进行检验，判断其是否有上述特殊字符，如果有，则报警提示错误，充分保证所执行的SQL语句是安全的；

（2）如果系统允许用户输入单引号“'”，则需要对用户输入的单引号“'”进行转换，转换为两个连续的单引号“''”，Delphi下的变换程序函数如下：

UserName.Text:=StringReplace(UserName.Txt,'','',[rfReplaceAll]);

此时语义并没有因为上述变换而改变，因为含有两个连续单引号“''”的字符常数在SQL标准中理解为含有单引号“'”的字符串；

（3）以上转换过程，也可以通过采用带参数的数据库查询方法，由高级语言数据库控件自动完成，此时，用户输入信息将作为查询参数。以下我们对原有程序进行了修改：

* 修改ADOQuery为带参数的查询：

将ADOQuery的SQL属性设置为“select * from users where username=:userName and password=:passWord”

其中，userName和passWord为查询参数

* 修改‘ ’处理函数为

```
procedure TfrmGetServerName.bOkClick(Sender: TObject);
  begin
    with DM.ADOQuery do
    begin
      Active:=False;
      //为查询参数赋值 Parameters.ParamByName('userName').Value:=UserName.Text;
          Parameters.ParamByName('passWord').Value:=PassWord.Text;
```

```
Active:=True;
if (RecordCount<>0) then
    Application.MessageBox('Logon succeed.', 'Logon', MB_OK);
        ...
 end;
end;
```

此时，我们再在Password文本输入框中输入： ' or '1'='1

结果：系统登录失败!

在SQL事件探查器中观察：

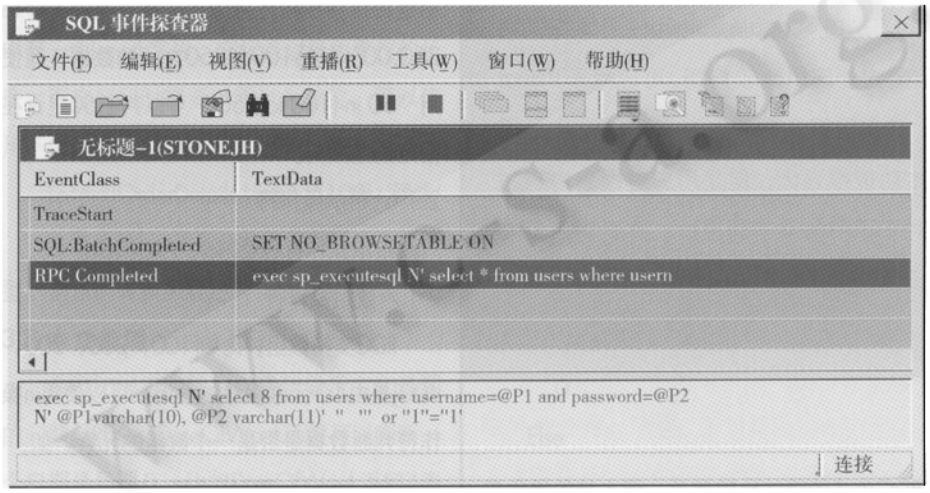

图 3 SQL 内部执行

实际运行的SQL语句变为：

exec sp_executesql N'select * from users where username=@P1 and password=@P2', N'@P1 varchar(10),@P2 varchar(11)', '', '' or ''1''='1'

解释：Delphi应用在向SQLServer数据库系统提交查询任务时，自动转换为相应的存储过程，这与Delphi的ADO控件内部实现有关；

在以上带参数的数据查询方法中，严格保证查询SQL的语义不随用户的输入信息而改变，同时将用户的输入信息与查询参数严格匹配，自动地将用户输入信息中的单引号“'”转换为两个单引号“''”，因此无论用户耍什么花招，如输入“' or '1'='1”，始终将其对应为一个内部含多个单引号信息的字符常数，不会再出现以往改变SQL语义的情况了。

3 结束语

本文所提及的SQL安全问题，不仅仅存在于传统CS结构应用中，而且也普遍存在于BS结构应用中，凡是那些直接通过用户输入信息拼装SQL的运行方式都有可能出现。“千里之堤，毁于蚁穴”，只有真正做好涉及安全问题的点点滴滴、方方面面，才能切实地保证系统安全稳定运行，这将是一个长期经验和教训积累的过程。

参考文献

1 James R.Groff 等著，关系数据库 SQL 使用指南，付增少等译，学苑出版社，1994:51~53。