文件上传绕过思路总结 - 先知社区

xz.aliyun.com/t/10515

才开始本来是项目碰到的，结果闹了乌龙。刚好有其他的平台，就总结一下用到的方法和思路。

参考链接：https://xz.aliyun.com/t/10459

绕waf的话，一般我的思路是硬怼，或者迂回打击。先说说两种思路

一 硬怼

硬怼的话，主要是从下面这些方法入手去操作。

（1）fuzz后缀名

看看有无漏网之鱼（针对开发自定义的过滤可能有机会，针对waf基本不可能。更多的情况是php的站寻找文件包含或者解析漏洞乃至传配置文件一类的，但是对于这种也大可不必fuzz后缀名了）

（2）http头变量改造

首先要明确waf的检测特征，一般是基于某种特定的情况下，去针对相应的拦截。几个例子，文件上传的时候，大多数Content-Type都是application/multipart-formdata这种，name对于waf来说，如果针对这种规则，对xxe ，sql注入，上传，命令执行，内容等所有都去做一波扫描是及其浪费内存的，所以有可能针对不同的类型，做了不同的校验规则。此时通过对Content-Type进行修改，可能会绕过waf。其他的http头添加删除等也是类似。

（3）文件后缀构造

这个和第一个有相似的就是都针对后缀名进行改造，不同的在于这里可能会利用waf的截取特征，比如回车换行绕过waf的检测，但是对于后端来说接收了所有的传入数据，导致了绕过waf。

（4）其他方法

这种就比较杂了，但是又不属于迂回打击的一类，比如重写等方法。接下来就实战来试试

第一步，先来对waf的规则做一个简单的判断。这里我的习惯是从内容，后缀两个方向进行判断。简单来说，基本分为这几种情况

（1）只判断后缀（基本碰到的比较少了，因为很多时候白名单开发都可以完成）

（2）只判断内容（也比较少，因为一般的waf都会带后缀的判断）

（3）内容后缀同时判断（这种情况比较多，相对于来说会安全一点）

（4）根据文件后缀来判断内容是否需要检测（较多）

（5）根据Content-Type来判断文件内容是否需要检测

暂时只想到这么多，以后碰到了再单独记吧。

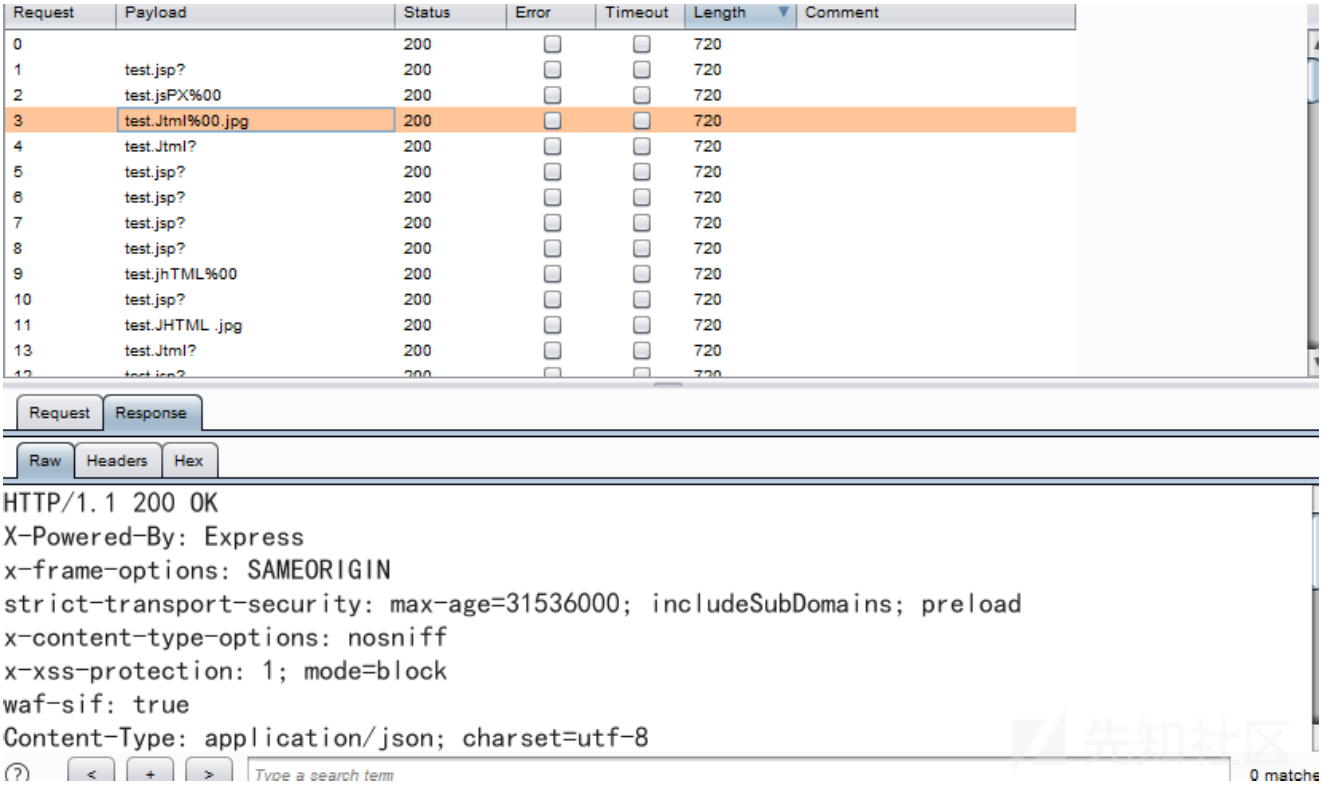
有了思路，那么接下来就好说了。举个例子我这里的情况

（1）传脚本后缀（被拦截，判断了后缀）

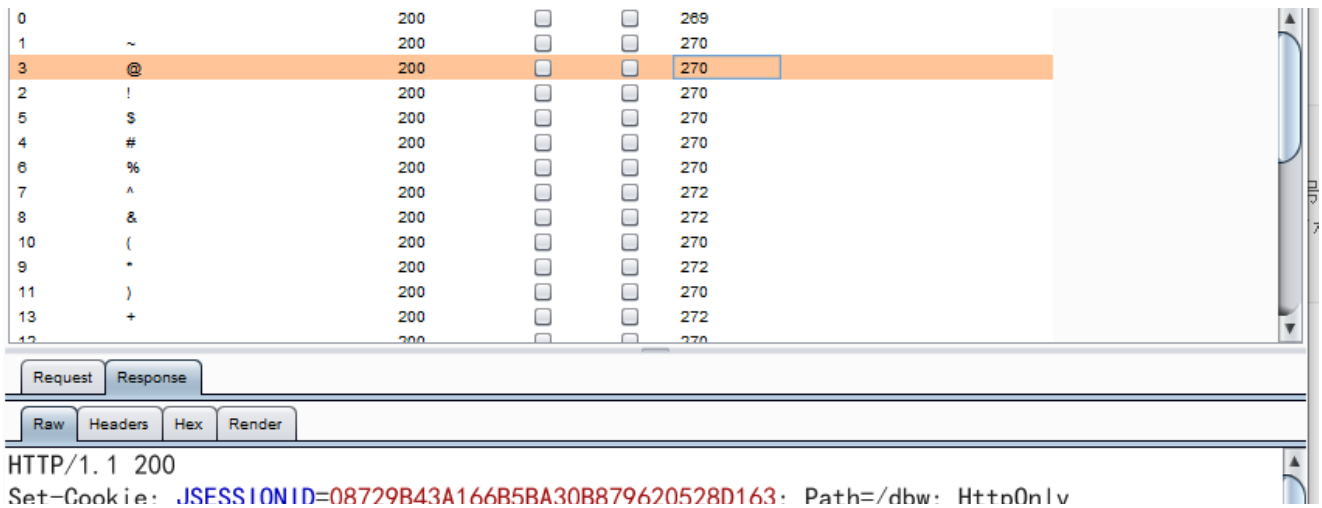
（2）传脚本后缀加不免杀代码（被拦截，可能后缀内容同时拦截）

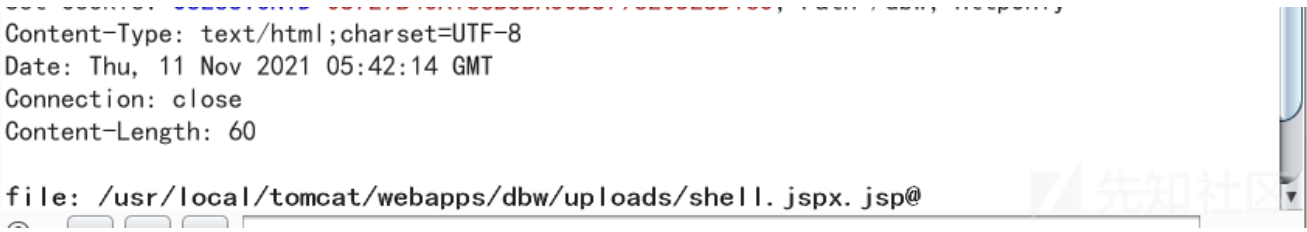
（3）传非脚本名（可自己fuzz一个能过waf的任意后缀，里面加恶意内容，被拦截。也就是说同时会对内容和后缀进行判断）

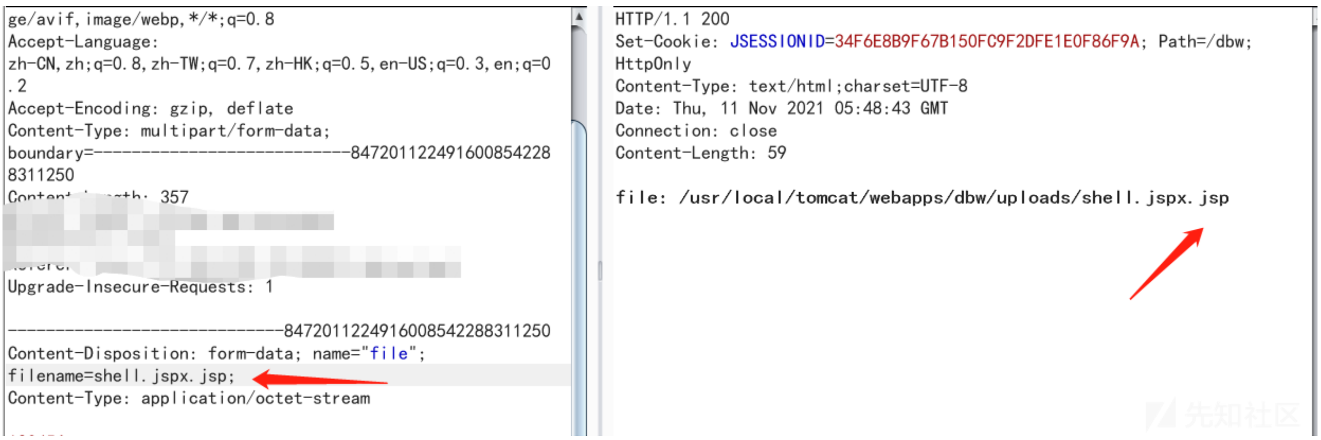
说说我这里的情况，会对内容和后缀进行拦截。检测到上传jsp文件，任意内容都会被拦截。先来fuzz一波能利用的后缀名，这里可以包括中间件的一些配置文件。希望不大，一点都不出意外，全部被拦截了。



既然我们需要对后缀名进行改造，就对后缀名后面加特殊符号做一个fuzz试试，测试了一下，在没有恶意内容的情况下，只有'被过滤了。所以如果有机会，我们看看能不能试试系统特殊，比如;去做截断。先记下来。因为最终还是需要免杀马的，jsp免杀又不会，先不考虑这个，先考虑把waf绕过。（这里我对filename做了换行，然后去掉了引号，加了一个;做截断绕过了waf，但是内容被查杀了，尴尬。）







接下来对http头部进行改造一下尝试绕过

一.filename改造

（2） 名字特殊符号替换以及构造异常闭合（符号方法很多自己天马星空，我这里就写几个就行了，但是要注意你改造了得让后端识别到，乱改造识别不到等于白搭）

filename='shell.jspx.jsp'

filename= `shell.jspx.jsp`

filename=shell.jspx.jsp'

"filename"=shell.jspx;

（3）重写

filename=shell.jpg;filename=shell.jspx;

filename=shell.jspx;filename=shell.jpg;

（4）大小写变化

FileName=shell.jspx.jsp'

（5）参数污染

FileName=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaashell.jspx.jsp'

FileName =shell.jspx（加空格）

filename===="shell.jspx.jsp1"（加等号）

FileName =shell.jspx（前后加空格，中间也可以加特殊符号fuzz）

（6）文件名字编码（filename一般为后端接收参数，编码了可能识别不到，这个就看情况）

filename=\u0073\u0068\u0065\u006c\u006c\u002e\u006a\u0073\u0070

（7）回车换行（有时候确实挺好用的，任意位置都可以试一下）

1.FileName=shell.jspx.
jsp

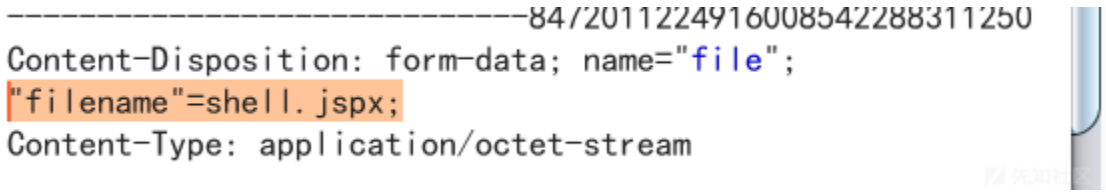
2.File

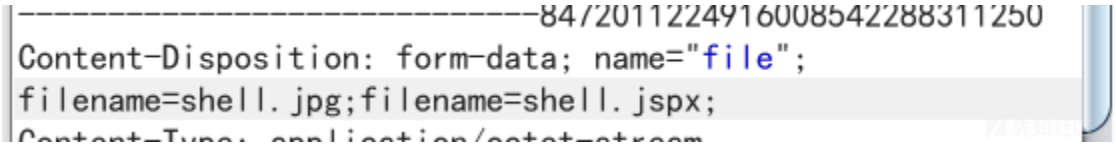
Name=shell.jspx.jsp'

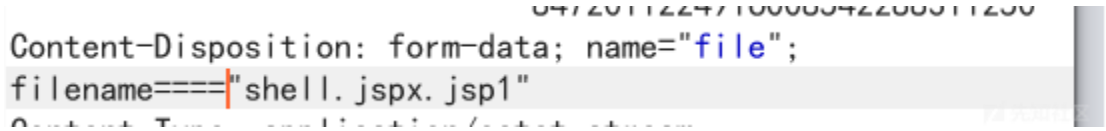
二 name改造

name也可以任意改造，改造的方法和filename差不多，就不重复发了，主要是思路重要。

其他的比如奇奇怪怪的正则需要用到的特殊字符都可以在文件名中fuzz一下，看看能否打断waf规则，也就是把我们fuzz后缀的再跑一次，或者再找点其他的正则字母，这里就不重复写了。







http头部格式上传相关绕过

有一些用畸形相关的，不太推荐一来就试，fuzz的可以带一下，这种属于天时地利人和占据才用，毕竟底层的规定好的合规变了就不能识别，但是也说不准fuzz出问题了呢。fuzz本来就是一个天马行空的过程，好了，继续来看。

（1）Content-Disposition

溢出绕过

Content-Disposition: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa form-data; name="file"; filename=shell.jpg;filename=shell.jspx;

回车换行绕过（注意不要把固定字段打散了，）

Content-Disposition:

form-data; name="file"; filename=shell.jpg;filename=shell.jspx;

双写绕过（写两次）

Content-Disposition: form-data; name="file"; filename=shell.jpg;filename=shell.jspx;

Content-Disposition: form-data; name="file"; filename=shell.jpg;filename=shell.jspx.jpg;

还有一些参数污染加减空格啥的，和上面filename类似，就不重复写了。

（2）boundary

加减空格或者前面加恶意的参数

boundary =----------------------------847201122491600854228831125O

&boundary =----------------------------847201122491600854228831125O

123& boundary =----------------------------847201122491600854228831125O

多个污染（他是用来分割的，他变了下面的也要变一下）

boundary =--------------------------847201122491600854228831125 1
boundary =--------------------------847201122491600854228831125 2
回车换行污染
分割污染（简单来说就是他自定义了一些分割部分，我们可以把我们的恶意参数提交到其他的分割部分）见下图第一个，视情况而定。其他的常用方式和上面都可以重复的
（3）Content-Type
直接删除
修改类型为application/text或者 image/jpeg等等
回车换行
溢出
参数污染
重复传入Content-Type
大小写变换
设置charset
Content-Type: multipart/form-data;charset=iso-8859-13
列举几个
ibm869
ibm870
ibm871
ibm918
iso-2022-cn
iso-2022-jp
iso-2022-jp-2
iso-2022-kr
iso-8859-1
iso-8859-13
iso-8859-15
还有其他的方式，其实和上面的思路差不多

```
Content-Disposition: form-data; name="file";
filename=shell.jspx.jsp#
Content-Type: application/octet-stream

1234567
-----------------------------847201122491600854228831125 0
Content-Disposition: form-data; name="file";
filename=shell.jspx.jsp#
Content-Type: application/octet-stream

1234567
-----------------------------847201122491600854228831125 0
Content-Disposition: form-data; name="submit"
```

```
Submit
```

http头部其他绕过

这一块就比较多了，编码，长度等等，都可以试一下，具体的方法和上面的差不多。这里就用参考链接pureqh老哥的几个东西了。

1.Accept-Encoding 改变编码类型

Accept-Encoding: gzip

Accept-Encoding: compress

Accept-Encoding: deflate

Accept-Encoding: br

Accept-Encoding: identity

Accept-Encoding: *

下面截取的图片是我本次的，就不弄其他的了，长度那一块，主要是说内容方面相关的。

```
Content-Disposition: form-data; name="file";
filename=shell.jspx.jsp#
Content-Type: application/octet-stream

1234567
-----------------------------847201122491600854228831125O
Content-Disposition: form-data; name="file";
filename=shell.jspx.jsp#
Content-Type: application/octet-stream

1234567
-----------------------------847201122491600854228831125O
Content-Disposition: form-data; name="submit"

Submit
```

2.修改请求方式绕过

post改为get put等其他的请求方式（这一块主要是针对waf的拦截特性）

3.host头部绕过

对host进行回车，换行

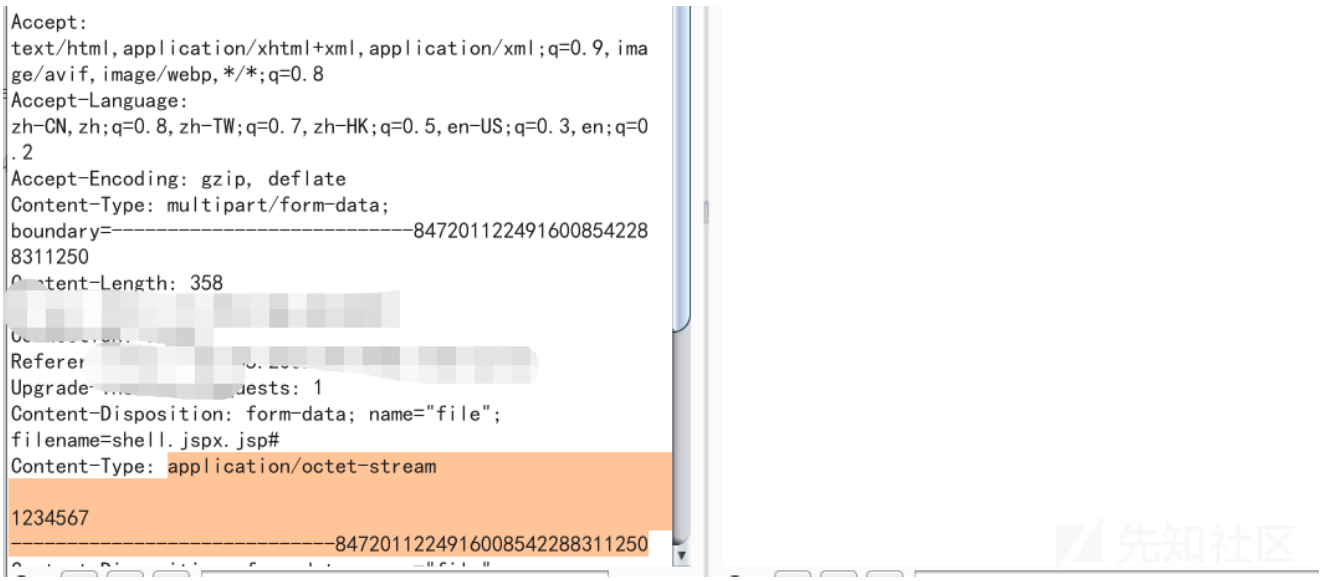
修改host头部

host跟链接

host改为127.0.0.1

删除host

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:94.0) Gecko/20100101 Firefox/94.0
```



到这里就差不多了，再来回头理一下我们的思路。借用露迅先生的一句话，你如果啥都不晓得就莽起整，一些都等求于零。所以我们总结一下我们的思路。
waf的特性大多数是写了很多的规则，基于截取的内容做规则匹配，匹配到了就不放行，未匹配到就认为是安全的放行，所以我们需要做的就是绕过waf对于规则的匹配。大概是这几个方向
（1）基于正则匹配的绕过（也就是参数污染，正则破坏等上面的方法，打乱waf的检测）
（2）基于正则匹配的缺失（类似于修改请求等，让waf根本不去检测这部分的内容）
（3）基于操作系统的特性（类似于后缀名加特殊符号让操作系统进行识别）
我们做一切的前提都是既绕过了waf，也能让后端识别，所以可以乱来，不要太乱。基本也就是污染，多写，绕过，添加删除几个方向。

二 迂回打击

说是迂回打击，但是其实就是利用一些通用的手段，或者中间件的特性去绕过waf，甚至说寻找到了真实ip去直接绕过云waf等方法。这里我就简单总结一些，不全面的话忘体谅。这一块主要是内容相关的了。

基于http的绕过

这种属于硬怼，方法如下：
1.免杀马
这种是万能的，只要能免杀就能如履平地，但是现在的waf规则更新太快了，熬了一夜去弄了个免杀，第二天踩了蜜罐上去就被抓，蓝方产品支持加入规则，一点也不美滋滋，但是这也是一条YYDS的道路
2.分块传输
说实话这玩意儿我从来没有成功过，但是面试问的挺多的，有一次有个面试官还专门跟我提了这个所以我这里列举一下。但是分块参数+参数污染组合利用貌似效果还是不错
3.修改长度字段
和分块参数有点类似，作用是这样，有些时候做参数大数据污染的时候，waf判断数据过长直

接丢弃，有些判断长度和内容相差太多也直接丢弃。这时候可以把两者结合起来使用，达到超长数据绕过waf的检测，同时数据送到了后端

4.修改传输编码

和分块传输类似，自己手动去改，burp那个插件工具我是一次都没成功过

5.基于网站系统特性添加字段

比如ASP专属bypass-devcap-charset，添加这些字段去绕过waf的检测（这也是我看到但是没机会实战，记录一下）

6.修改头部+内容结合

修改头部为其他格式，再把内容头加其他格式，例如图片，中间插入恶意代码，类似图片马

7.增加多个boundary

这样子打乱了恶意内容，有点类似分开传输，欺骗waf的检测，逃逸后面的代码。

8.文件名写入文件

windows下利用多个<<<<去写入文件，详情可以看参考链接。

还有一些其他的方法，这一种也是类似于对waf欺骗，过着直接利用免杀硬过waf的。jsp免杀不会，就不献丑免杀了。

其他绕过

这种绕过就是一般适用于云waf了。咋说呢，这种我碰到的不怎么多，因为一般碰到的云waf基本都很强，注入上传类的绕过现在越来越难了，xss还好一点，但是不走钓鱼的话xss也没用太大的用处，毕竟可以一把梭最舒服。来看看吧，检测全球ping就行。

1.寻找真实ip

这个方法网上太多了，说下我常用的

（1）利用ssl证书寻找

（2）利用子域名寻找

（3）利用公司其他业务寻找（跑C端看运气，和子域名一样）

（4）利用信息泄露寻找（github，google，目录文件，js代码等）

（5）利用一些云网站或者专门查找cdn的网站，链接在家里电脑上，这电脑没有，就自己去找吧

（6）利用已知工具

（7）搜索引擎（fofa，夸克等，看以前收集的业务）

（8）利用http返回信息

（9）找邮箱弱口令，然后你懂的

（10）找朋友，你懂的。

2.利用子域名去打

有些网站，可能外面做了防护，子域名没加waf，而子域名又在白名单，迂回去锤就行了。

3.利用头部绕过

基本碰不到了，修改host为本地ip，现在已经绝迹了，突然想起来写一下。

4.找设备

找一些vpn一类的设备碰碰运气

其他的就不说了吧，头痛。总结下这个思路

（1）直接寻找waf保护后的目标地址，进行亲身拥抱（绕过waf去打）

（2）寻找waf后目标的子女子孙亲儿子（被waf加白的一些资产）去挑拨离间。

上一篇：Thymeleaf SSTI漏洞分析

下一篇：DedeCMS-5.8.1 SST...