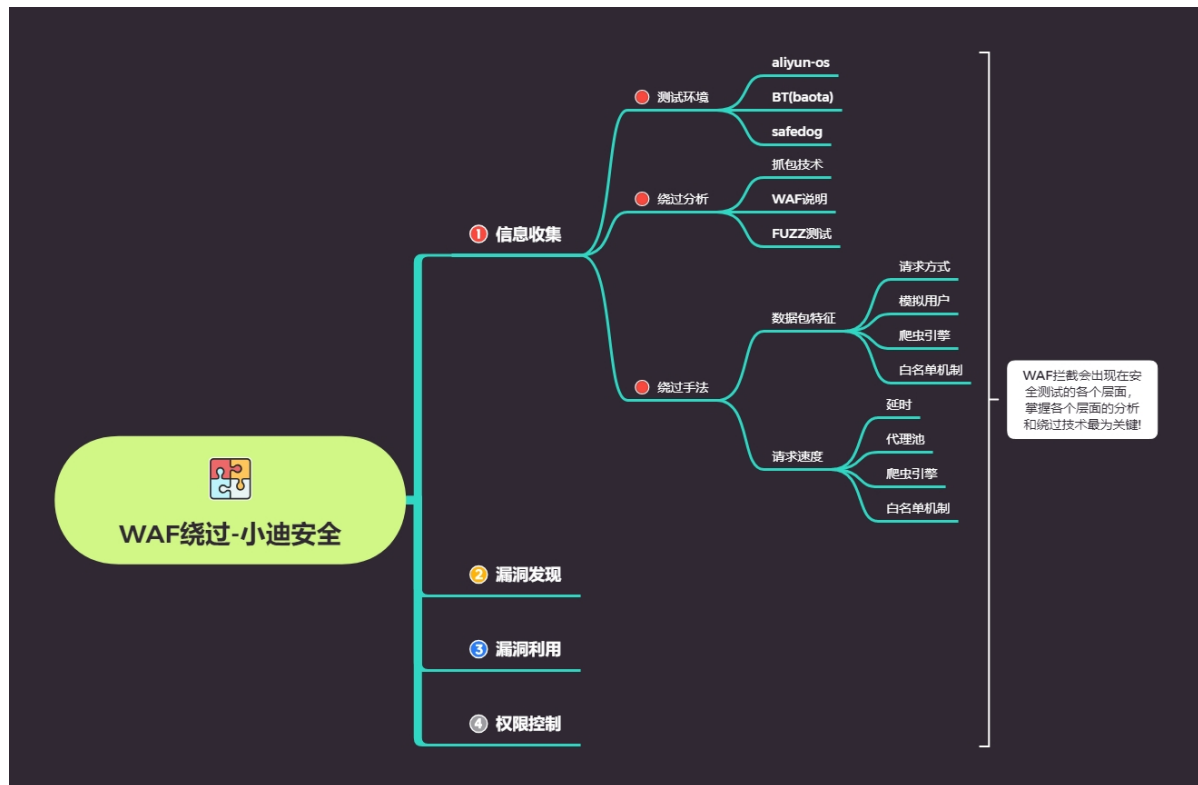


# Day46 WAF绕过-信息收集之反爬虫延时代理池技术

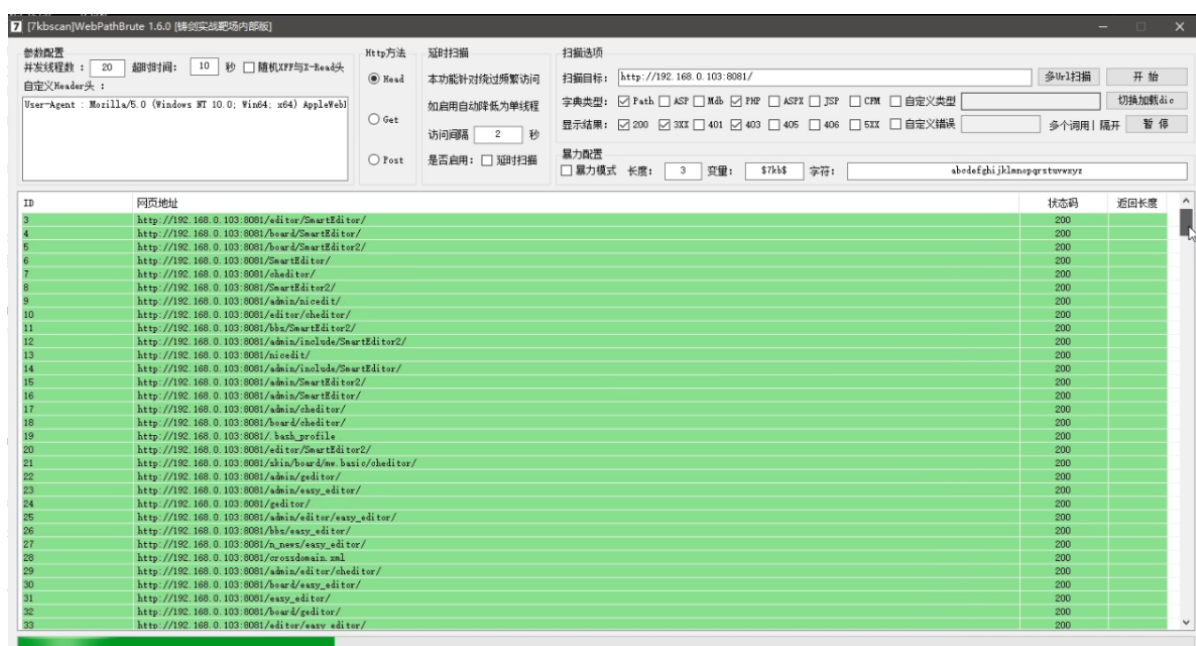


## 46.1 Safedog-默认拦截机制分析绕过-未开 CC

safedog ——CC攻击防护默认是关闭的:



用目录扫描工具扫:



字典的所有的结果都有，明显错误！抓取进程的抓包工具抓包，与浏览器数据包对比，明显请求方式不同：

```

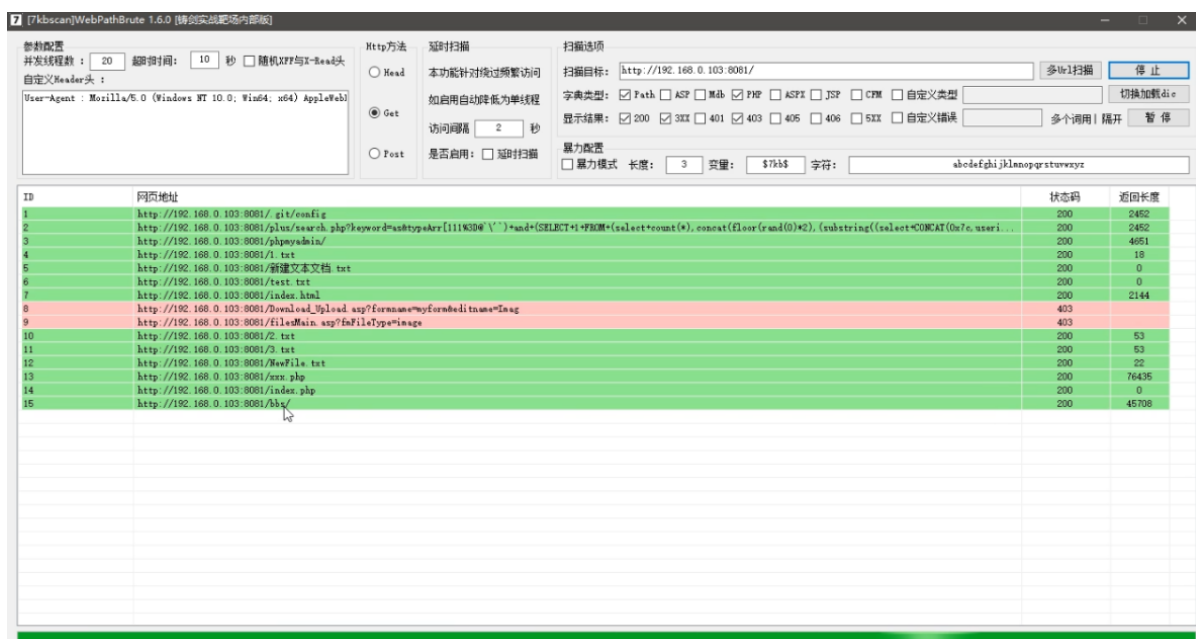
1 HEAD /bbs/easy_editor/ HTTP/1.1
2 Accept: text/html, application/xhtml+xml, image/jxr, */*
3 Accept-Language: en-US, en; q=0.8, zh-Hans-CN; q=0.5, zh-Hans; q=0.3
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
5 Host: 192.168.0.103:8081
6 Connection: Close
7
8
9 GET /sssada HTTP/1.1
10 Host: 192.168.0.103:8081
11 Connection: keep-alive
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: zh-CN,zh;q=0.9
17 Cookie: UM_distinctid=1723ba091ce6a-030410c3773dff-3f6b490f-384000-1723ba091cfb3f; CNZZ

```

工具扫描数据包

浏览器访问数据包

改为GET请求方式，就是正常的扫描结果：



## 46.2 Safedog-默认拦截机制分析绕过-开启 CC

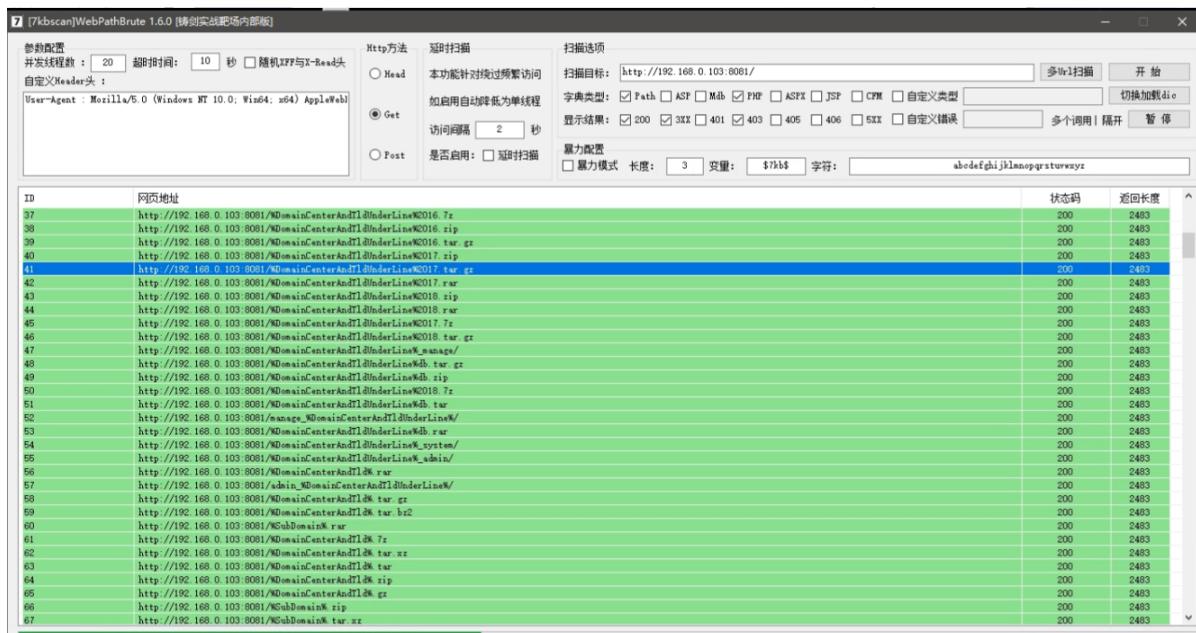
开启CC防护：



查看防护规则:



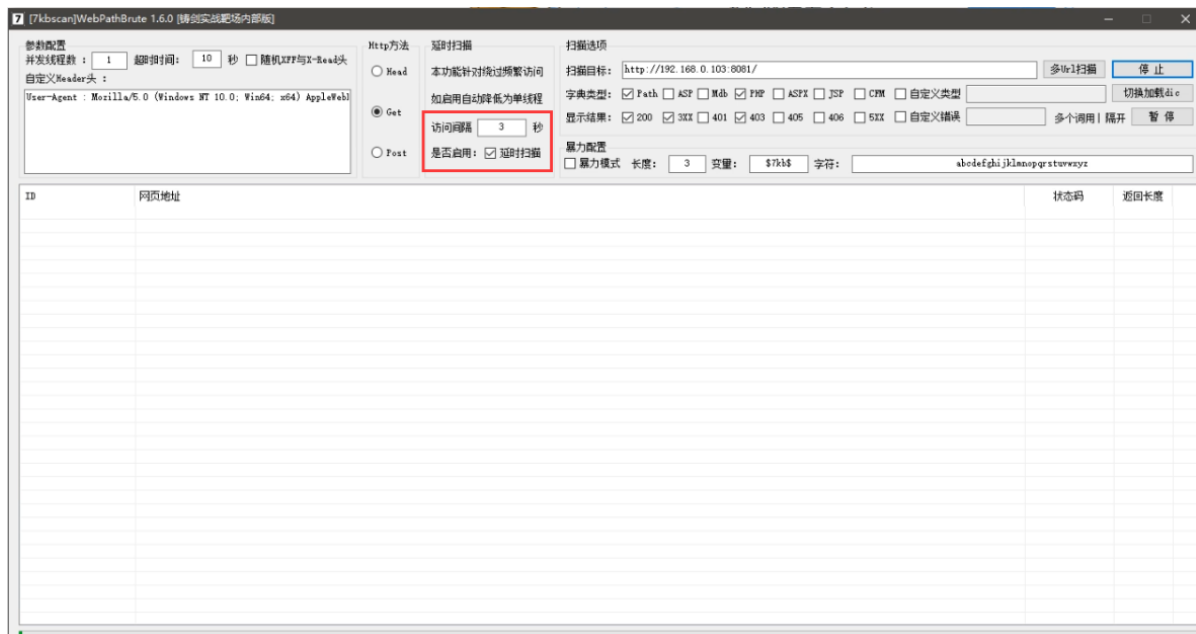
开启CC之后，再次使用扫描工具，并且访问网站:



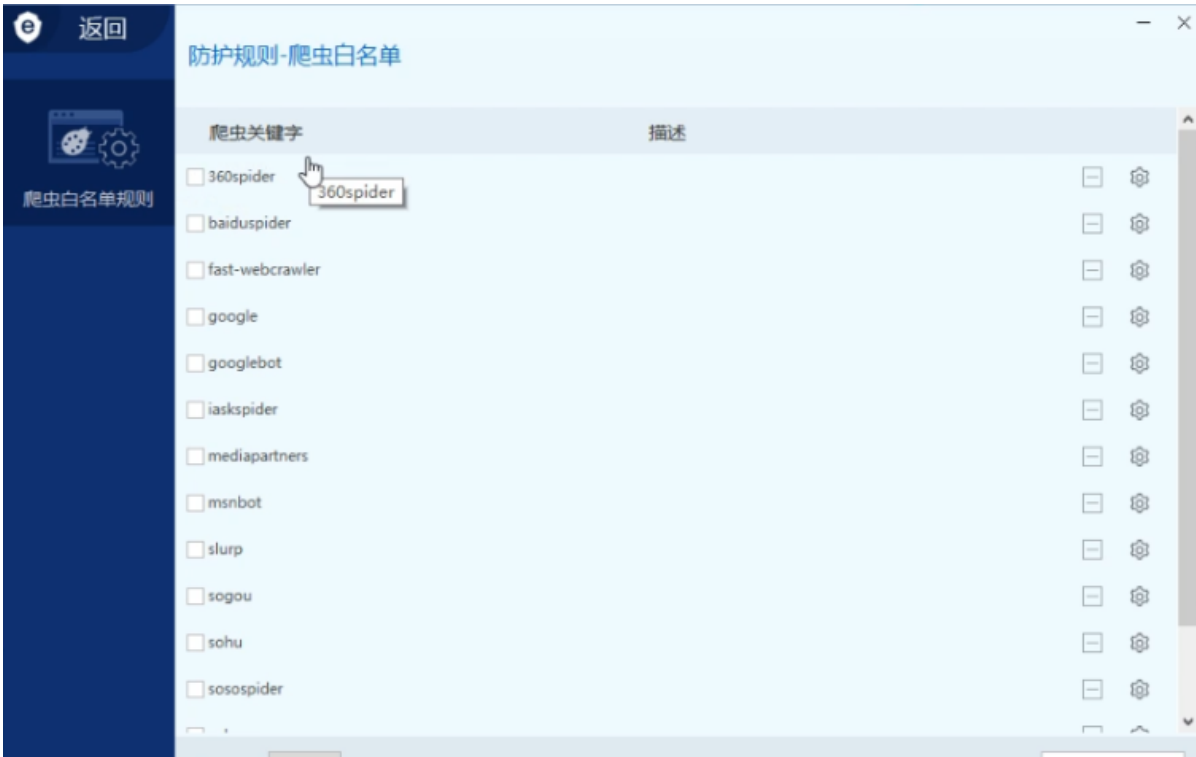
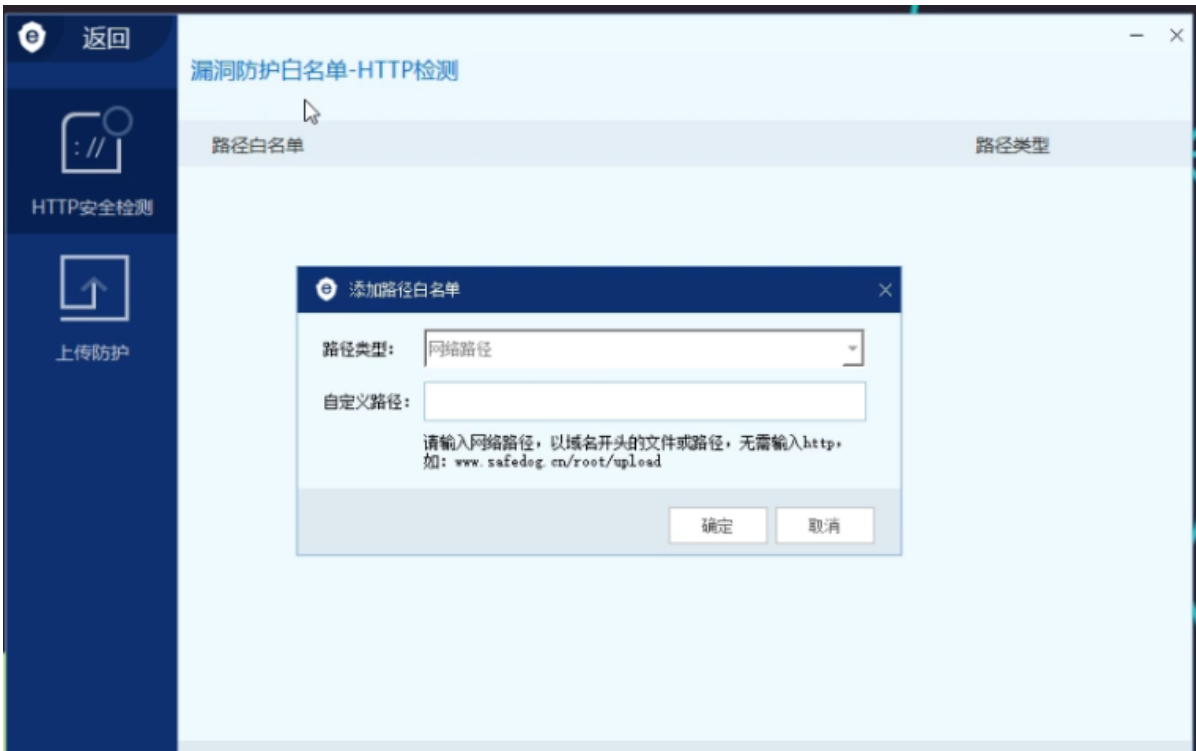
频繁访问（访问频率过快，绝对是工具），触发安全狗防护机制，被拦截：



设置延时扫描时间：



或者也可以通过白名单:



模拟搜索引擎请求头User-Agent就可以了,进行扫描:



- 1 谷歌搜索引擎蜘蛛爬虫:
- 2 Mozilla/5.0 (compatible; Googlebot/2.1;  
+http://www.google.com/bot.html)
- 3 Googlebot/2.1  
(+http://www.googlebot.com/bot.html)
- 4 Googlebot/2.1 (+http://www.google.com/bot.html)
- 5 Googlebot-Image/1.0



- 1 百度搜索引擎蜘蛛爬虫:
- 2
- 3 移动 UA:
- 4 Mozilla/5.0 (Linux;u;Android 4.2.2;zh-cn;)  
AppleWebKit/534.46 (KHTML,likeGecko) Version/5.1  
Mobile Safari/10600.6.3 (compatible;  
Baiduspider/2.0;  
+http://www.baidu.com/search/spider.html)
- 5 PC UA: Mozilla/5.0 (compatible; Baiduspider/2.0;  
+http://www.baidu.com/search/spider.html)
- 6
- 7 移动 UA:
- 8 Mozilla/5.0 (iPhone; CPU iPhone OS 9\_1 likeMac  
OS X) AppleWebKit/601.1.46 (KHTML, like Gecko)  
Version/9.0 Mobile/13B143 Safari/601.1  
(compatible; Baiduspider-render/2.0;  
+http://www.baidu.com/search/spider.html)
- 9
- 10 PC UA:
- 11 Mozilla/5.0 (compatible; Baiduspider-render/2.0;  
+http://www.baidu.com/search/spider.html)



- 1 搜狗搜索引擎蜘蛛爬虫:
- 2 Sogou web  
spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)
- 3
- 4 图片蜘蛛:
- 5 Sogou Pic  
spider/3.0(+http://www.sogou.com/docs/help/webmasters.htm#07)



- 1 360 搜狗搜索引擎蜘蛛爬虫:
- 2 Mozilla/5.0 (Windows NT 6.1; WOW64)  
AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/50.0.2661.102 Safari/537.36; 360Spider



- 1 Bing 搜索引擎蜘蛛爬虫:
- 2 Mozilla/5.0 (compatible; bingbot/2.0;  
+http://www.bing.com/bingbot.htm)



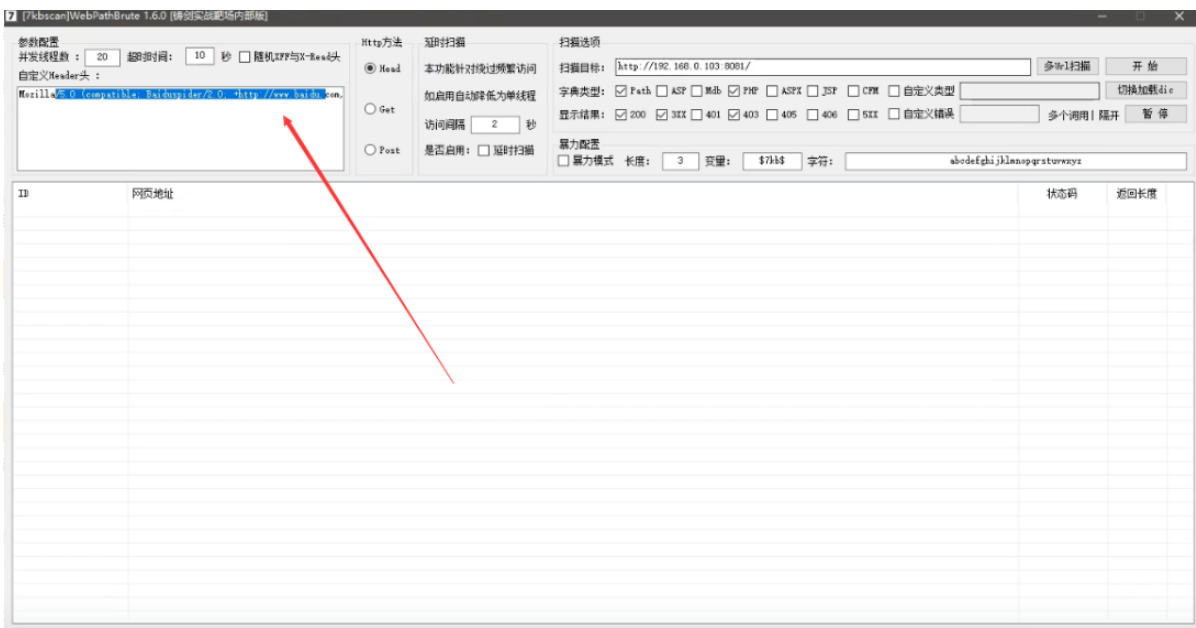
- 1 Yandex 搜索引擎蜘蛛爬虫:
- 2 Mozilla/5.0 (compatible; YandexBot/3.0;  
+http://yandex.com/bots)



- 1 <https://www.imtqy.com/amp/spider-ua-ip.html>
- 2 <https://ziyuan.baidu.com/college/articleinfo?id=1002>
- 3 <http://www.webkaka.com/tutorial/zhanzhang/2017/061068/>
- 4 <https://www.cnblogs.com/iack/p/3557371.html>



此时更换完user-agent后发现并没有任何结果。因为单单只修改了user-agent，如果对方对你请求包的内容进行检测的话，还是会被发现是工具进行的访问：



1.可以使用python脚本自定义请求包来访问：

可以通过python自定义来模拟正常用户的请求数据包，以及更改为百度谷歌等搜索引擎的user-agent来对网站进行访问请求，此时对方不会拦截。

抓包分析：此处为python脚本自定义的请求数据包

```
1 GET http://192.168.0.103:8081/administrator/components/com_contenthistory/models/history.php HTTP/1.1
2 Host: 192.168.0.103:8081
3 User-Agent: Mozilla/5.0 (compatible; Baiduspider-render/2.0; +http://www.baidu.com/search/spider.html)
4 Accept-Encoding: gzip, deflate
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Connection: close
7 Cache-Control: max-age=0
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
14 Cookie: bdshare_firsttime=1581597934650; PHPSESSID=ncsajdvh39qse0qlsgqokshuc4; yx_auth=dc4fq8FAEkyiAU254b5z19GGStCxXoRb1TFaAaozygM1Sc5uZYHjR3gCQm*2BtRNz3bcjbTi8BRgcd*2F7LvR01HN1j319CI6x2922QDI38
```

如果不使用自定义的请求数据包：

```
1 GET http://192.168.0.103:8081/administrator/components/com_contenthistory/models/history.php HTTP/1.1
2 Host: 192.168.0.103:8081
3 User-Agent: python-requests/2.23.0
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7
8
```

## 2.代理池

可通过python写的脚本不停更换代理ip访问进行测试,检测ip代理,一般是CC防护,但是使用代理,扫描目录,大概率是可行的:



### 46.3 Aliyun\_os-默认拦截机制分析绕过-简要界面

- 阿里云-无法模拟搜索引擎爬虫绕过,只能采用代理池或者延时。
- 一般访问频率过快,站点直接就访问不了,一个小时后才可以访问。

## 46.4 BT(防火墙插件)-默认拦截机制分析绕过-简要界面

The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom.

**File Explorer:**

- Myproject D:\Myproject
  - AngelSword-master
  - CMSRex-master
  - CVE-2018-7600-master
  - CVE-2019-12815-master
  - dirsearch-master
  - DnslogSqlj-master
  - Impacket-master
  - IPProxyPool-master
  - JSHell-master
  - LaZagne-master
  - NoSQLMap-master
  - OneForAll-master
  - SatanSword-master
  - sqlmap\_ip
  - subDomainsBrute-master
  - winExploit-master
  - Ypsocn
  - TPiscan-master
  - tv
  - venv library root
  - vulna-master
  - webshell-venom-master
  - wufuzi-master
  - wufuzi 文档、翻译
  - WindowsVulnScan-master

**Code Editor:**

```
21 paths=paths.replace('\n','')
22 urls=url+paths
23 proxy = {
24     'http': '127.0.0.1:7777',
25 }
26 try:
27     code=requests.get(urls,headers=headers).status_code
28     time.sleep(3)
29     print(urls+'|'+str(code))
30     if code==200 or code==403:
31         print(urls+'|'+str(code))
for paths in open('php_b.txt').e... : try
```

**Terminal:**

```
D:\Myproject\venv\Scripts\python.exe D:/Myproject/fuckscan.py
http://test.xiaodi8.com//administrator/components/com_contenthistory/models/history.php|404
http://test.xiaodi8.com//index.php?m=search&c=index&a=public_get_suggest_keyword&url=asdf&q=../../../../
http://test.xiaodi8.com//index.php?m=search&c=index&a=public_get_suggest_keyword&url=asdf&q=../../../../
connecting error
connecting error
connecting error
```

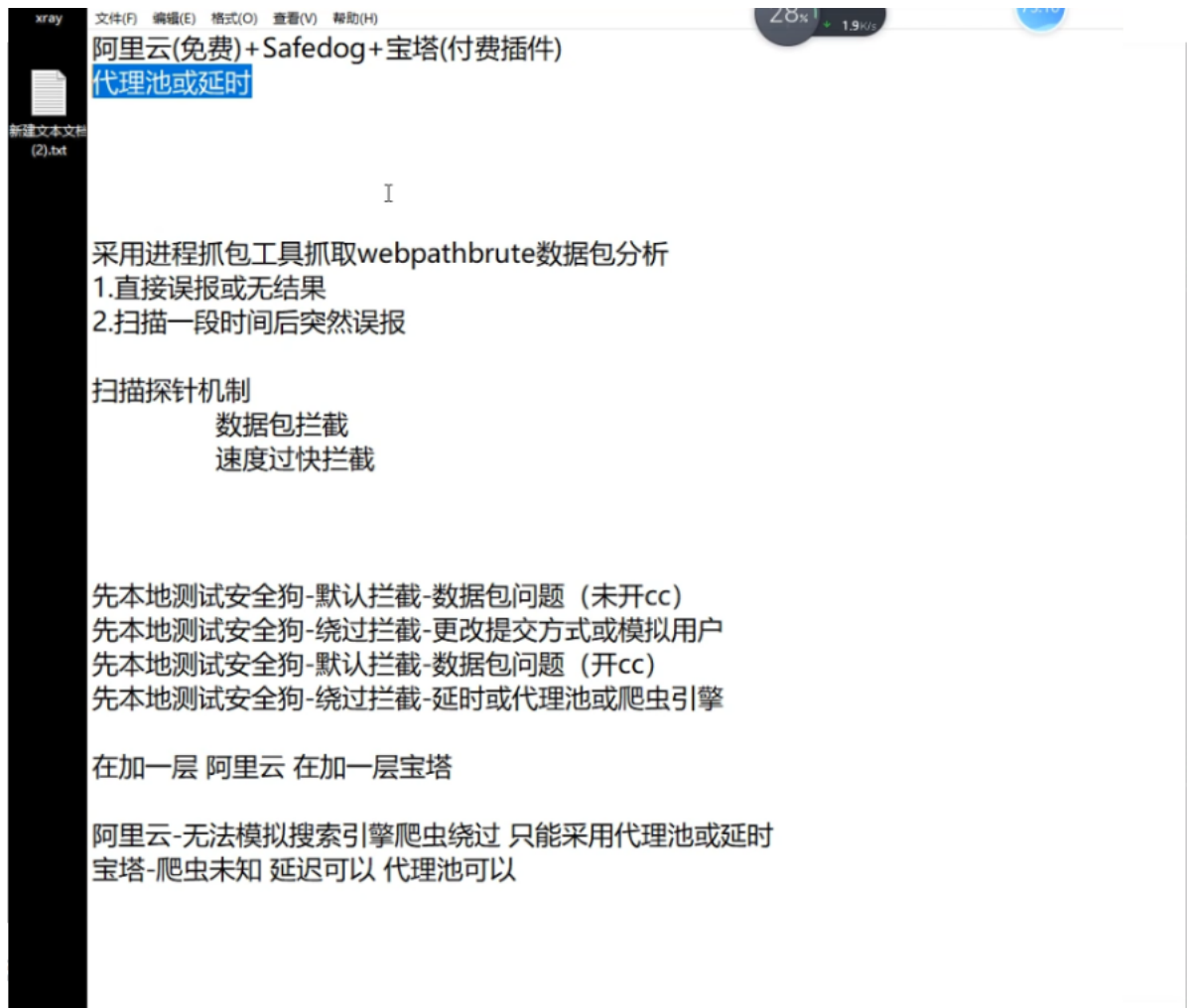
扫描3个后就不行，不是alliyun拦截的，而是BT的防火墙拦截的：

网站配置【test.xiaodi8.com】			
网站防火墙开关 <input checked="" type="checkbox"/>			
名称	描述	状态	操作
CC防御	60 秒内,请求同一URI累计超过 120 次,封锁IP 100 秒	<input checked="" type="checkbox"/>	<a href="#">设置</a>
恶意容忍设置	60 秒内,累计超过 6 次恶意请求,封锁IP 600 秒	--	<a href="#">设置</a>
GET-URI过滤	过滤uri、uri参数中常见sql注入、xss等攻击	<input checked="" type="checkbox"/>	<a href="#">规则</a>
GET-参数过滤	过滤uri、uri参数中常见sql注入、xss等攻击	<input checked="" type="checkbox"/>	<a href="#">规则</a>
POST过滤	过滤POST参数中常见sql注入、xss等攻击	<input checked="" type="checkbox"/>	<a href="#">规则</a>
User-Agent过滤	通常用于过滤浏览器、蜘蛛及一些自动扫描器	<input checked="" type="checkbox"/>	<a href="#">规则</a>
Cookie过滤	过滤利用Cookie发起的渗透攻击	<input checked="" type="checkbox"/>	<a href="#">规则</a>
禁止国外访问	禁止中国大陆以外的地区访问站点	<input type="checkbox"/>	<a href="#">设置</a>
常见扫描器	过滤常见扫描测试工具的渗透测试	<input checked="" type="checkbox"/>	<a href="#">设置</a>
使用CDN	该站点使用了CDN,启用后方可正确获取客户IP	<input type="checkbox"/>	<a href="#">设置</a>
禁止执行PHP的URL	禁止在指定URL运行PHP脚本	--	<a href="#">设置</a>
禁止访问的URL	禁止访问指定的URL	--	<a href="#">设置</a>
禁止扩展名	禁止访问指定扩展名	--	<a href="#">设置</a>
禁止上传的文件类型	禁止上传指定的文件类型	--	<a href="#">设置</a>
受保护的URL	通过自定义参数加密URL地址,参数错误将被拦截	--	<a href="#">设置</a>
URL专用过滤	为特定URL地址设置过滤规则	--	<a href="#">设置</a>

- 注意: 此处大部分配置, 仅对当前站点有效!

- 1 绕过:
- 2 字典拆开使用 , 绕过他的恶意容忍机制60秒内6次
- 3 字典优化
- 4 `index.php.bak` -----> `index.php.bak.`

## 46.5 总结



## 资源:

- 1 <https://www.bt.cn/>
- 2 [http://free.safedog.cn/website\\_safedog.html](http://free.safedog.cn/website_safedog.html)
- 3 <https://www.cnblogs.com/iack/p/3557371.html>