

# Web 安全测试常规流程

## 一、关于

本文介绍 web 安全测试中的常规流程 ,以尽量覆盖常见 web 安全漏洞的测试为目的 ,且以中高危漏洞的检测为主 ,低危漏洞暂不考虑。建议测试的漏洞类型包含 :

任意文件上传、SQL 注入、命令注入、代码执行、存储型 XSS、SSRF、XXE、任意文件读取。

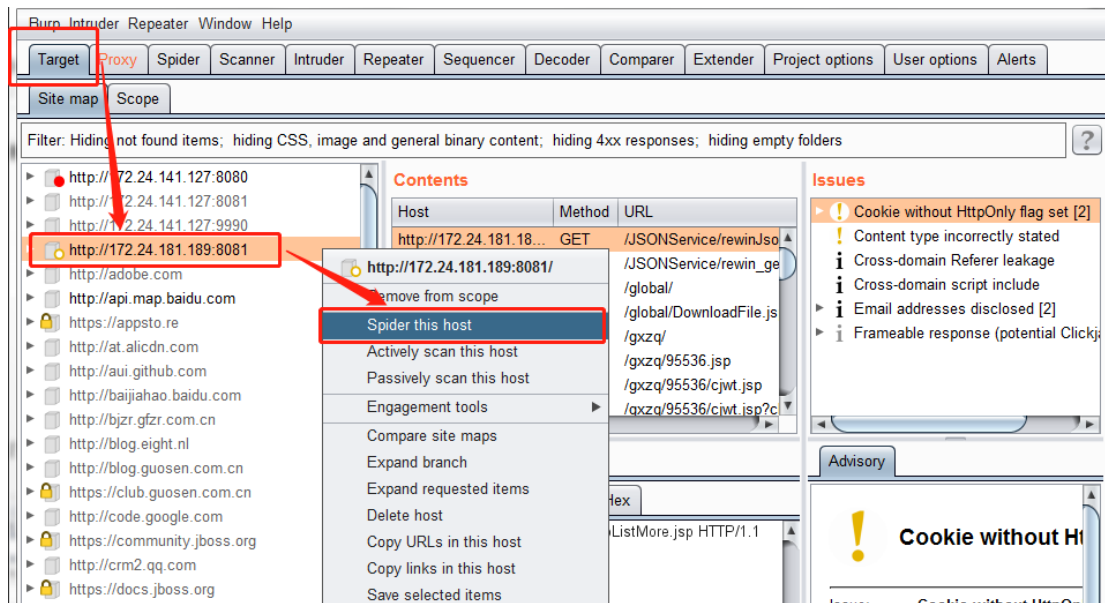
## 二、流程

测试流程一般分为 2 步 ,扫描器测试和人工测试 ,如下分别介绍。

### 1、扫描器测试

扫描器选择 Burpsuite ( 或者其他工具 ) ,Burpsuite 可覆盖所有常见 web 漏洞的漏洞 ,使用流程如下 :

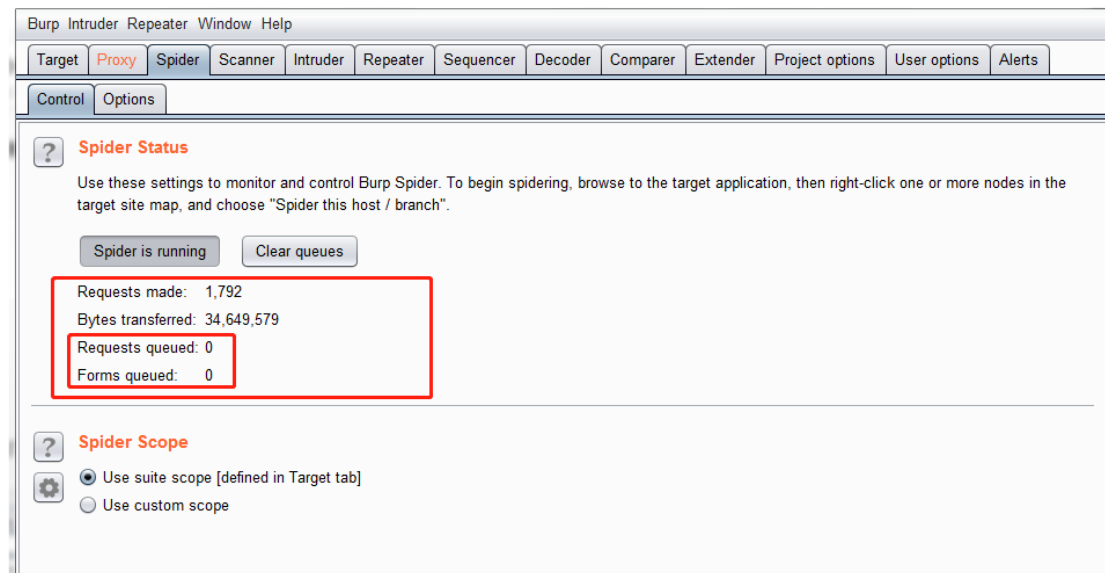
1 ) 选择 Target 功能标签



2) 在需要扫描的目标 url 上右键弹出菜单

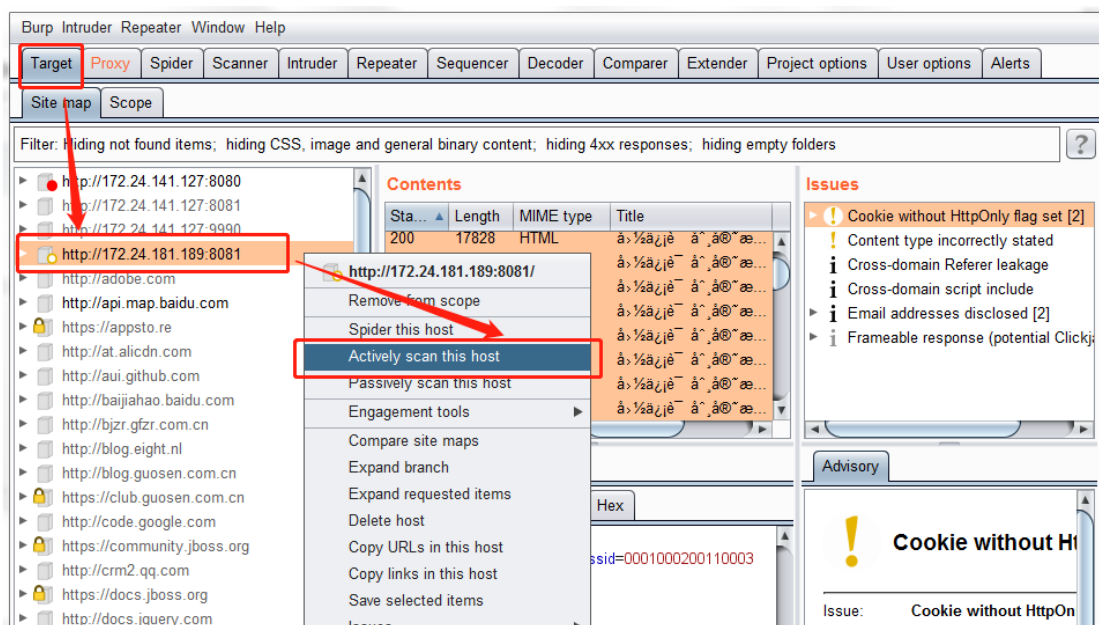
3) 在弹出的菜单上选择 Spider this host

4) 过一段时间后查看 Spider 功能标签查看是否完成爬虫



如果 Requests queued 和 Forms queued 都是 0 则代表完成了爬虫。

5) 在 Target 功能标签里右键单击目标域名，并选择 Actively scan this host



## 6) 查看 Scanner 功能标签中的扫描情况，如下图所示

The screenshot shows the Burp Suite Scanner tab. It displays a table of scanned URLs and their status, issues, requests, and errors. The table has columns for '#', 'Host', 'URL', 'Status', 'Issues', 'Requests', and 'Errors'. The data is as follows:

#	Host	URL	Status	Issues	Requests	Errors
35	http://172.24.181.189:8081	/gxzq/gxyw/index_rzrq.jsp	finished	13	431	
36	http://172.24.181.189:8081	/gxzq/gxyw/jhlcJzData.jsp	finished		371	
37	http://172.24.181.189:8081	/gxzq/gxyw/jhlc_list.jsp	finished	4	396	
38	http://172.24.181.189:8081	/gxzq/gxyw/jhlc_xx.jsp	finished	8	461	
39	http://172.24.181.189:8081	/gxzq/gxyw/jht.jsp	finished	13	429	
40	http://172.24.181.189:8081	/gxzq/gxyw/jtl.jsp	finished	4	396	
41	http://172.24.181.189:8081	/gxzq/gxyw/lsyw_index.jsp	finished	4	396	
42	http://172.24.181.189:8081	/gxzq/gxyw/product/index.jsp	finished	3	397	
43	http://172.24.181.189:8081	/gxzq/gxyw/tzyh.jsp	finished	4	402	
44	http://172.24.181.189:8081	/gxzq/gxyw/ywbl.jsp	finished	4	394	
45	http://172.24.181.189:8081	/gxzq/gxyw/ywbl.jsp	finished	4	467	
46	http://172.24.181.189:8081	/gxzq/gxyw/ywbl.jsp	finished	4	540	
47	http://172.24.181.189:8081	/gxzq/gxyw/index.jsp	finished	6	426	
48	http://172.24.181.189:8081	/gxzq/jty/channelDetail.jsp	finished	5	397	
49	http://172.24.181.189:8081	/gxzq/lcxx/dfzt.jsp	finished	6	461	
50	http://172.24.181.189:8081	/gxzq/lcxx/fzqhd.jsp	finished	6	426	
51	http://172.24.181.189:8081	/gxzq/lcxx/videoList.jsp	finished	6	426	
52	http://172.24.181.189:8081	/gxzq/public/detail.html	finished	4	439	
53	http://172.24.181.189:8081	/gxzq/public/detail_jty.html	finished	1	448	

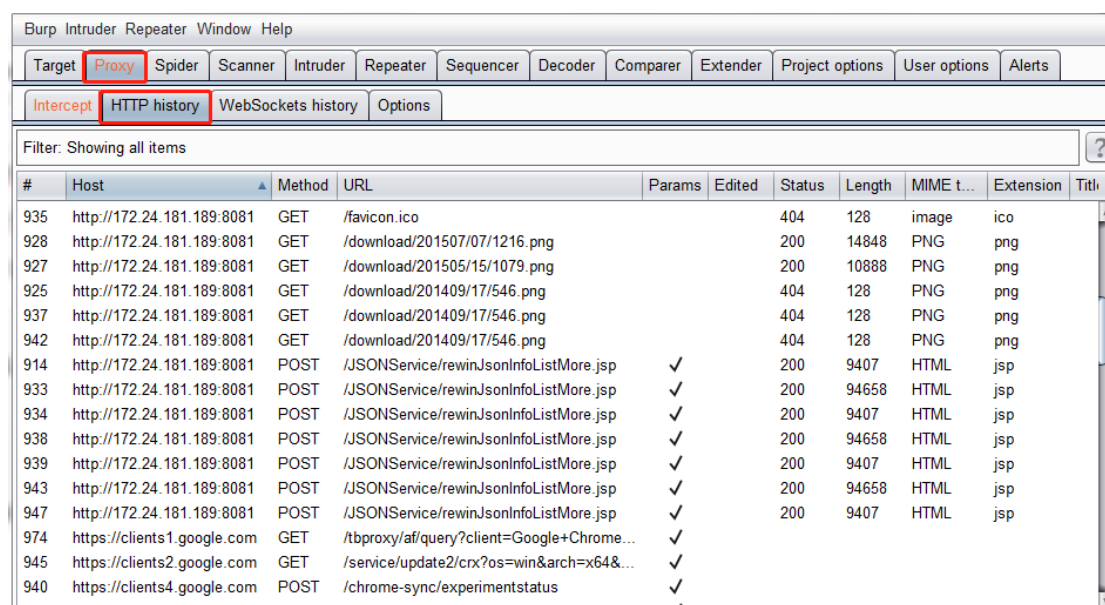
## 2、人工测试

扫描器的测试方法是污染参数的正常数据，通过服务器返回响应内容判断漏洞是否存在，可覆盖大部分的测试范围，但存在下面几种情况无法直接被扫描器测试

到：

- 1) 参数内容被加密 (如对称加密, 非对称加密)
- 2) 参数有其他限制 (如测试 url 有 csrftoken 等防御措施)
- 3) 参数的内容不是正常的格式 (如 json 格式, 序列化字符串)
- 4) 漏洞触发需要多个页面的联合操作 (如存储型 xss, 二次 sql 注入)
- 5) 逻辑漏洞 (如整数溢出, 越权)
- 6) 通过 js 生成的请求 (这种情况 js 生成 post 请求较多)
- 7) 服务器前面有 waf 防护

可根据 burpsuite 的 Proxy|HTTP history 中的 URL 来检查有没有满足以上条件的请求, 如下图：



The screenshot shows the Burp Suite interface with the 'HTTP history' tab selected. The table below represents the data shown in the history list.

#	Host	Method	URL	Params	Edited	Status	Length	MIME t...	Extension	Title
935	http://172.24.181.189:8081	GET	/favicon.ico			404	128	image	ico	
928	http://172.24.181.189:8081	GET	/download/201507/07/1216.png			200	14848	PNG	png	
927	http://172.24.181.189:8081	GET	/download/201505/15/1079.png			200	10888	PNG	png	
925	http://172.24.181.189:8081	GET	/download/201409/17/546.png			404	128	PNG	png	
937	http://172.24.181.189:8081	GET	/download/201409/17/546.png			404	128	PNG	png	
942	http://172.24.181.189:8081	GET	/download/201409/17/546.png			404	128	PNG	png	
914	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	9407	HTML	jsp	
933	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	94658	HTML	jsp	
934	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	9407	HTML	jsp	
938	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	94658	HTML	jsp	
939	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	9407	HTML	jsp	
943	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	94658	HTML	jsp	
947	http://172.24.181.189:8081	POST	/JSONService/rewindJsonInfoListMore.jsp		✓	200	9407	HTML	jsp	
974	https://clients1.google.com	GET	/tbproxy/af/query?client=Google+Chrome...		✓					
945	https://clients2.google.com	GET	/service/update2/crx?os=win&arch=x64&...		✓					
940	https://clients4.google.com	POST	/chrome-sync/experimentstatus		✓					

现对这几种情况分别简单举例：

## 1) 参数内容被加密

待测试 url 为：<https://www.baidu.com/?a=bmloYW8=>

这里参数 a 的值是 bmloYW8= 猜测是 base64 编码后的内容 经检验可 base64 解码成功，说明较大概率是服务器接收参数值后会先经过 base64 解码再响应，于是测试时需要将污染数据先经过 base64 编码再发送到服务器。可通过自行编码实现这个功能或通过 sqlmap 中的 eval 功能来实现测试。

## 2) 参数有其他限制

待测试 url 有 csrftoken 保护，这种情况也可以通过自行编码或者 sqlmap 的 csrf-token 和 csrf-url 功能来实现测试，详情可参考 `sqlmap -hh | grep csrf`

## 3) 参数的内容不是正常的格式

待测试 url 为：<https://www.baidu.com/?a={ 'p1' :v1, 'p2' :v2 }>

这里参数 a 的值是{ 'p1' :v1, 'p2' :v2 }，也即通过字典类型传参，这种情况可通过 sqlmap 检测或手动编码实现检测。

Sqlmap 检测命令：(这里检测 a 参数中的 p1 参数)

```
sqlmap -u "https://www.baidu.com/?a={ 'p1' :v1*, 'p2' :v2}"
```

#### 4) 漏洞触发需要多个页面的联合操作

待测试 url 为：<https://www.baidu.com/?a=value>

这里的 a 的内容会被存储到数据库中并在管理员后台显示，这种情况需要搭建 xss 平台来检测 a 参数是否存在存储型 xss 漏洞。

#### 5) 逻辑漏洞

待测试 url 为：<https://www.baidu.com/?a=value>

这里的 a 传入的值为付款的值，如在前端页面正常的值是 100 元，测试时可通过 burpsuite 的 proxy 功能将 100 修改成 0 或 -1 等，通过查看响应是否支付成功判断是否存在逻辑漏洞

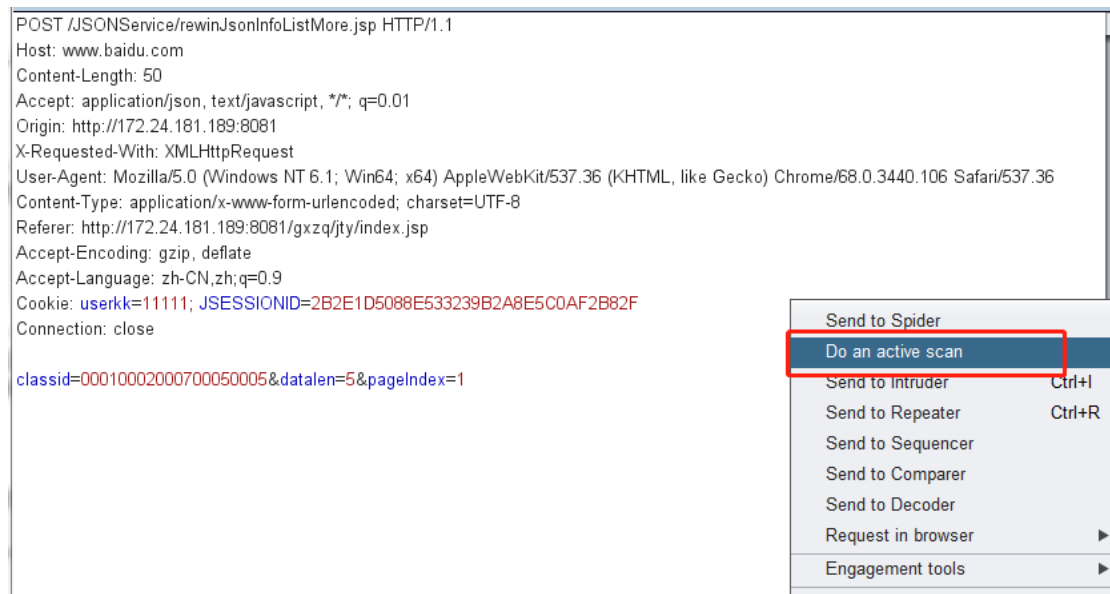
#### 6) 通过 js 生成的请求

GET 请求访问 <https://www.baidu.com/?a=value> 后发现响应内容中有 js，且 js 会自动生成一个如下的 POST 请求：

```
POST /JSONService/rewinJsonInfoListMore.jsp HTTP/1.1
Host: www.baidu.com
Content-Length: 50
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://172.24.181.189:8081
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://172.24.181.189:8081/gxzc/jty/index.jsp
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: userkk=11111; JSESSIONID=2B2E1D5088E533239B2A8E5C0AF2B82F
Connection: close

classid=000100020007000500005&datalen=5&pageIndex=1
```

这种通过 js 生成的请求无法通过 burpsuite 等扫描器测试到，需要自行捕获并测试。如在 burpsuite 中测试可通过右键 Do an active scan 功能实现。



## 7) 服务器前面有 waf 防护

这种情况下只能靠个人经验来测试了，可尝试不同的 waf 绕过手段来进行测试，常见 waf 绕过方法可参考：

<http://3xp10it.cc/web/2016/08/12/waf%E7%BB%95%E8%BF%87%E6%8A%80%E5%B7%A7%E5%BA%93/>