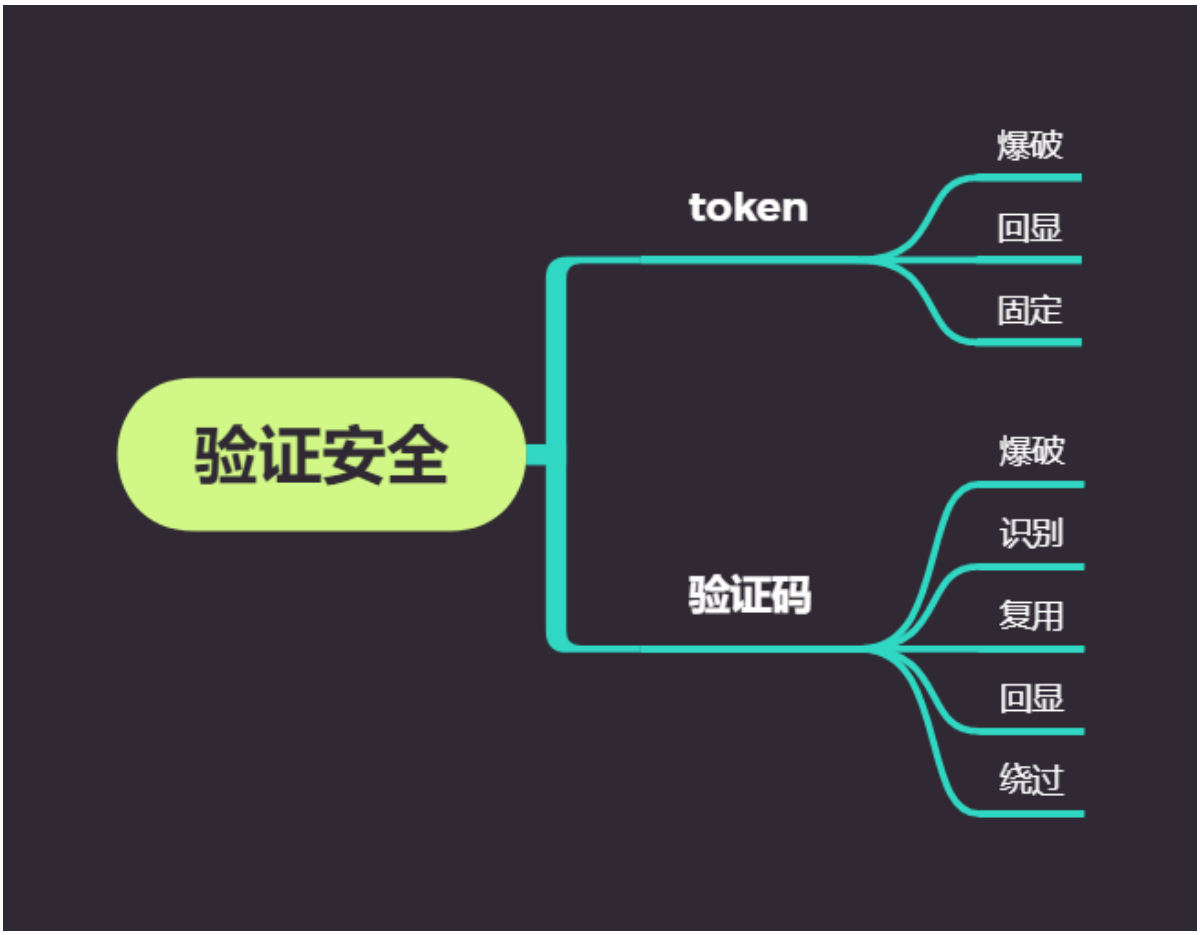
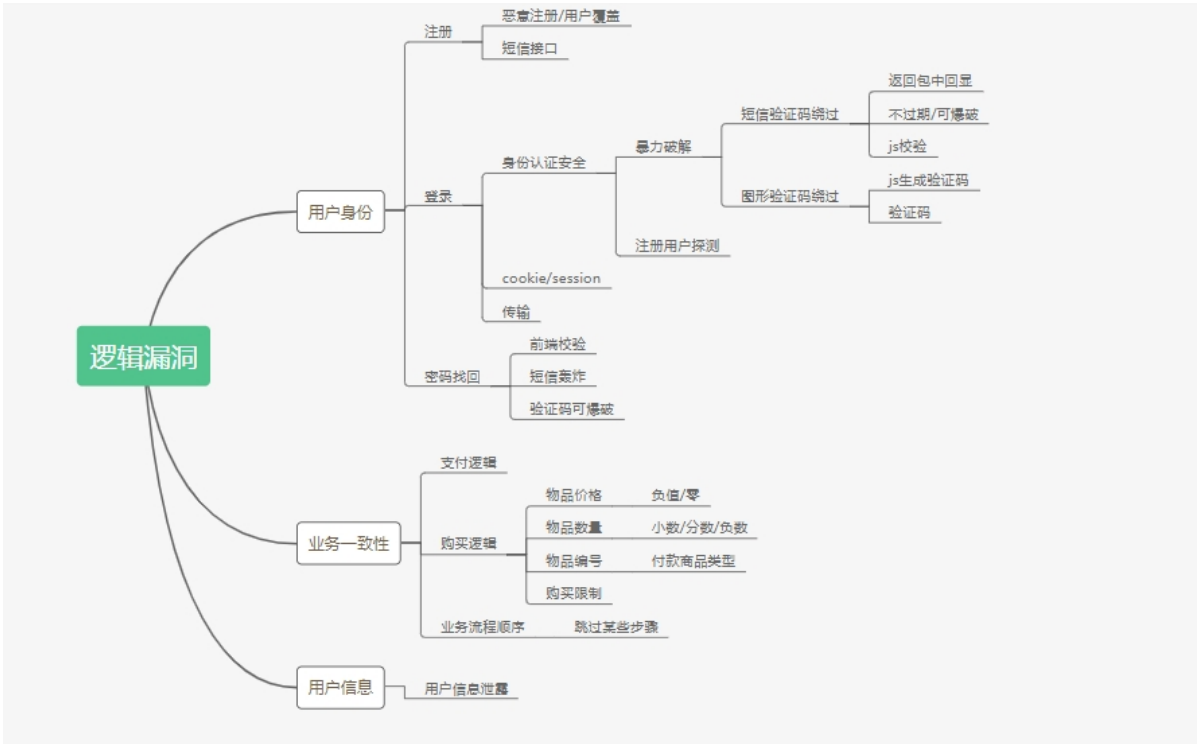
Day36　WEB漏洞–逻辑越权之验证码与 Token 及接口





36.1 验证码安全

36.1.1 分类

图片，手机或邮箱，语音，视频，操作等

36.1.2 原理

验证生成或验证过程中的逻辑问题

36.1.3 危害

账户权限泄漏，短信轰炸，遍历，任意用户操作等

36.1.4 漏洞

客户端回显(已讲)，验证码复用，验证码爆破(已讲)，绕过等

1. 验证码爆破：没有次数限制，验证码有效时间内不变，爆破绕过验证码的方法：验证码的复用，只需要发送第一次验证码，然后可以通过数据包爆破密码
2. 验证码识别：用工具识别验证码
3. 复用：用上一次的验证码来绕过下一次的验证
4. 回显：验证码在前端数据包显示
5. 绕过：逻辑上的绕过，直接跳过验证码

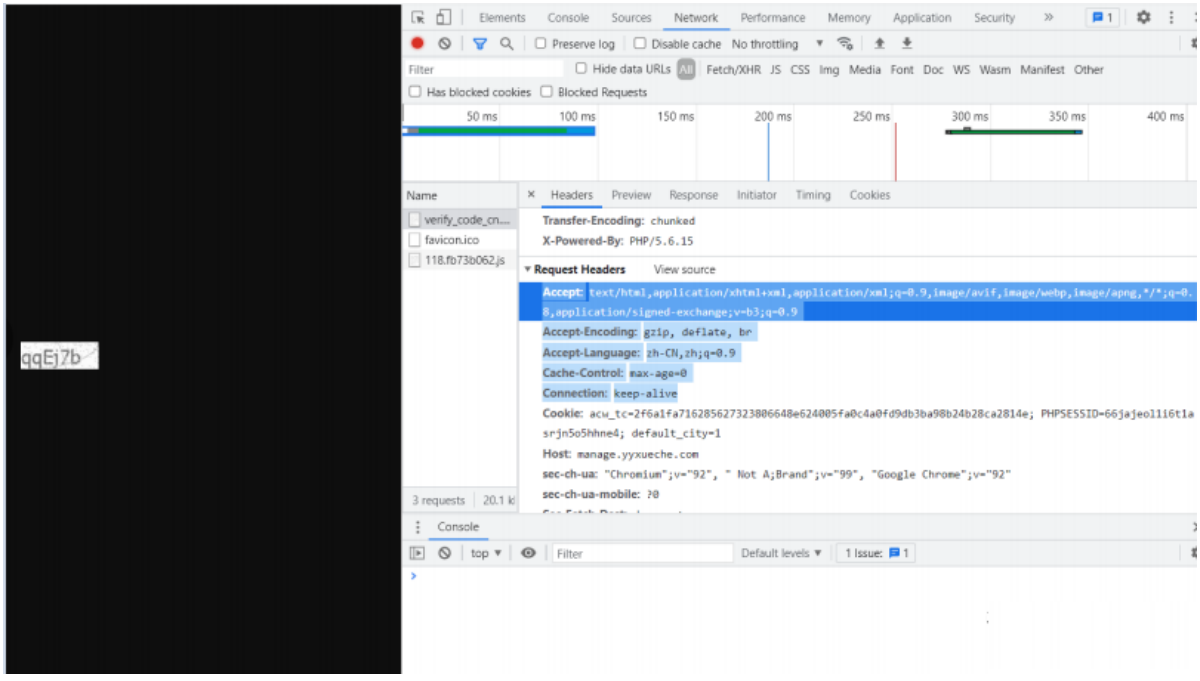
36.1.5 爆破防范措施

```
1. 如验证码输入3次就重新获取
2. 每输入一下变换一次
3. 使用特殊字符集
4. 特殊的验证码如图形，滑动，判断图形等
```

36.1.6 验证码案例分析

```
1  https://manage.yyxueche.com//panel/login.php
```

复制图片地址:



验证码地址，为验证码图片的链接地址:

Pkav HTTP Fuzzer 1.5.6 Verkey@Pkav安全团队 本程序仅供安全测试使用! 致谢PKAV全体成员! ©http://www.

◉ 图片型

验证码地址:

https://manage.yyxueche.com/verify_code_cn.php?103311

其他请求头部:

```
d-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
```

◉ 自带识别引擎

识别模式

◉ 单个文本统一块　○ 单一的文本行　○ 一个单词　○ 无OSD全自动页分割

○ 垂直对齐文本的统一块　○ 可变大小文本中的一列　○ 无OSD或OCR的自动页面分割

○ 仅OSD的定位及检测　○ OSD模式自动页面分割　○ 圈内的一个单词

识别范围

○ 不限定

○ 清晰的数字

◉ 限定为以下字符:

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

○ 第三方识别引擎

◉ 亦思验证码识别引擎　○ 次世代验证码识别引擎

识别库:

○ 第三方识别引擎

○ 亦思验证码识别引擎　◉ 次世代验证码识别引擎

识别库:

加载..

识别测试:

验证码图片:

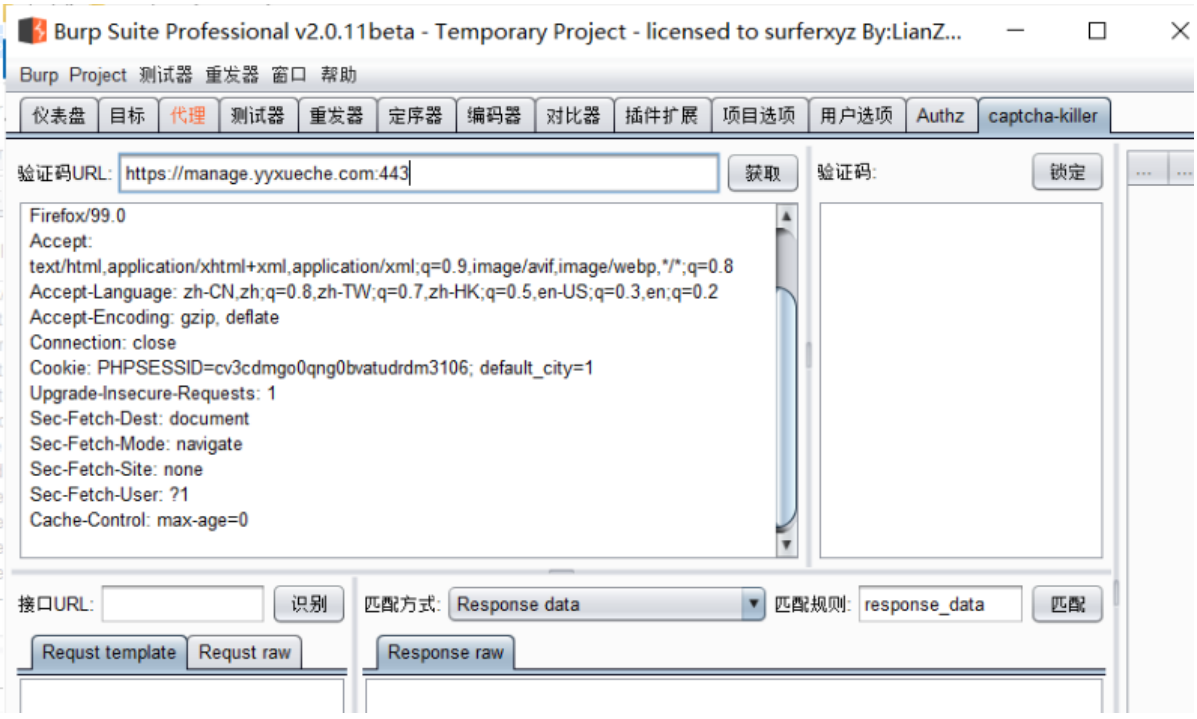
获取到的验证码为:

wPAeWF

```
总结：
右键选择验证码“复制图片地址”，访问图片地址
抓取图片地址的数据包：右键“审查元素”“Network”，刷新一下
找到请求数据包“Request Headers”复制GET，Host和Cookie之间的部分
打开工具，选择“图片型验证码识别”
将复制内容添加到“其他请求头部”，在填写“验证码地址”
根据需要调整参数、引擎等，点击“识别测试”
缺点：
识别精度有限
缺少接口，只能加载能够下载的识别库
只能识别简单的验证码
只能使用自带的发包器，不能结合到burpsuite
```

36.1.7 captcha-killer插件

浏览器访问验证数据包，抓包，右键点击captch-killer：

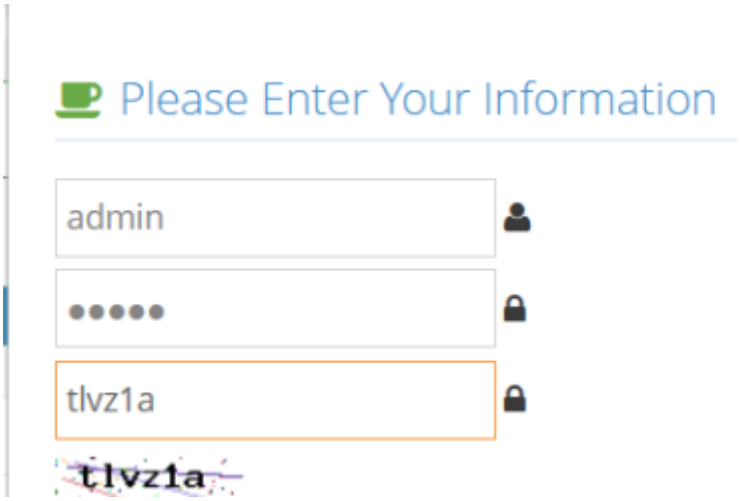


```
使用方法：
使用burpsuite抓包，右键发送到“captcha-killer”“Send to captcha panel”
接口URL填识别平台的接口地址
开始识别
爆破时使用该插件：“Attack type”选择“Pitchfork”，“Payload type”选择“Extension-generated”，“Extension payload generator”选择“captcha-killer”。不要多线程
```

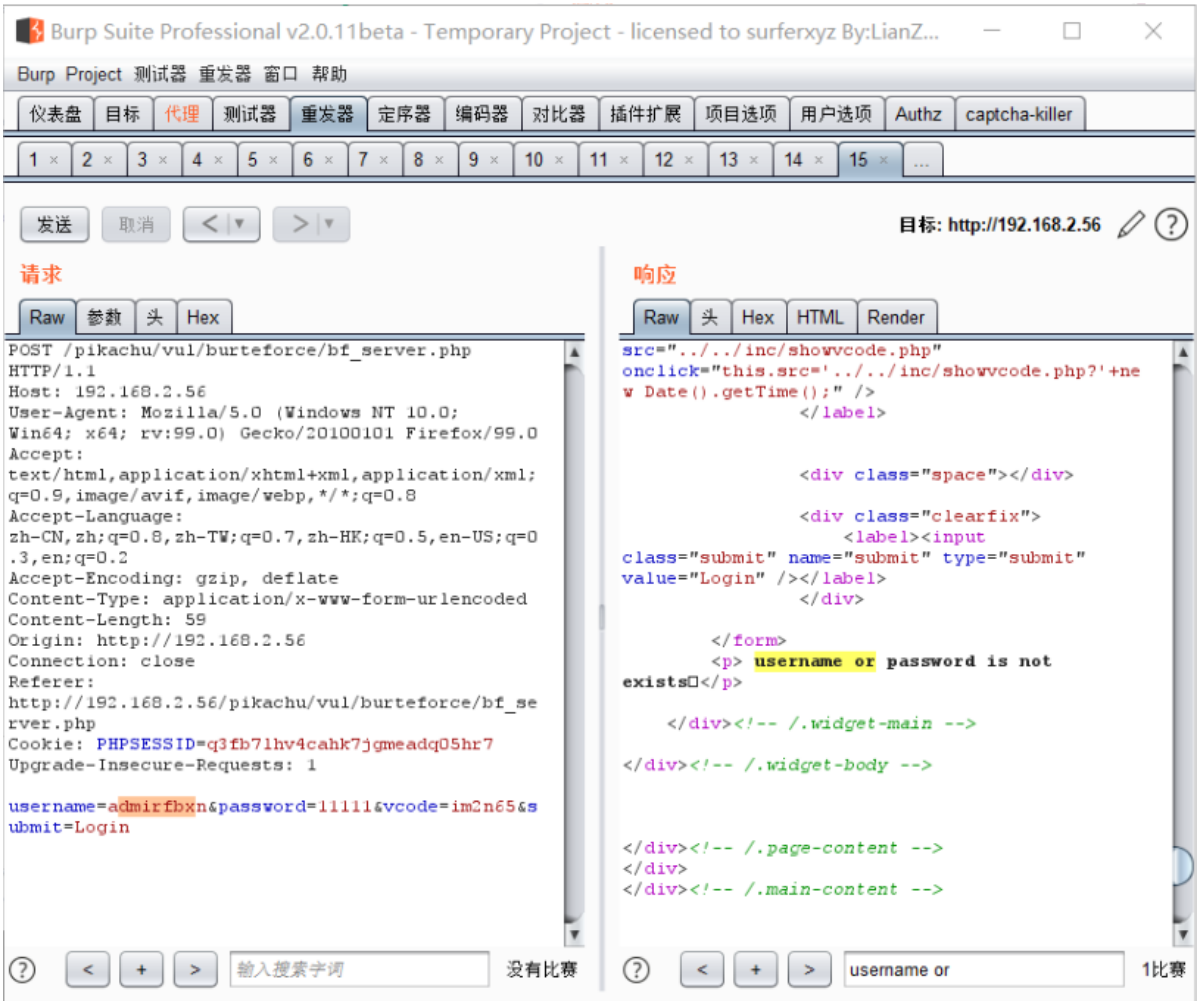
36.1.8 验证码绕过本地及远程验证-本地及实例

1.on server

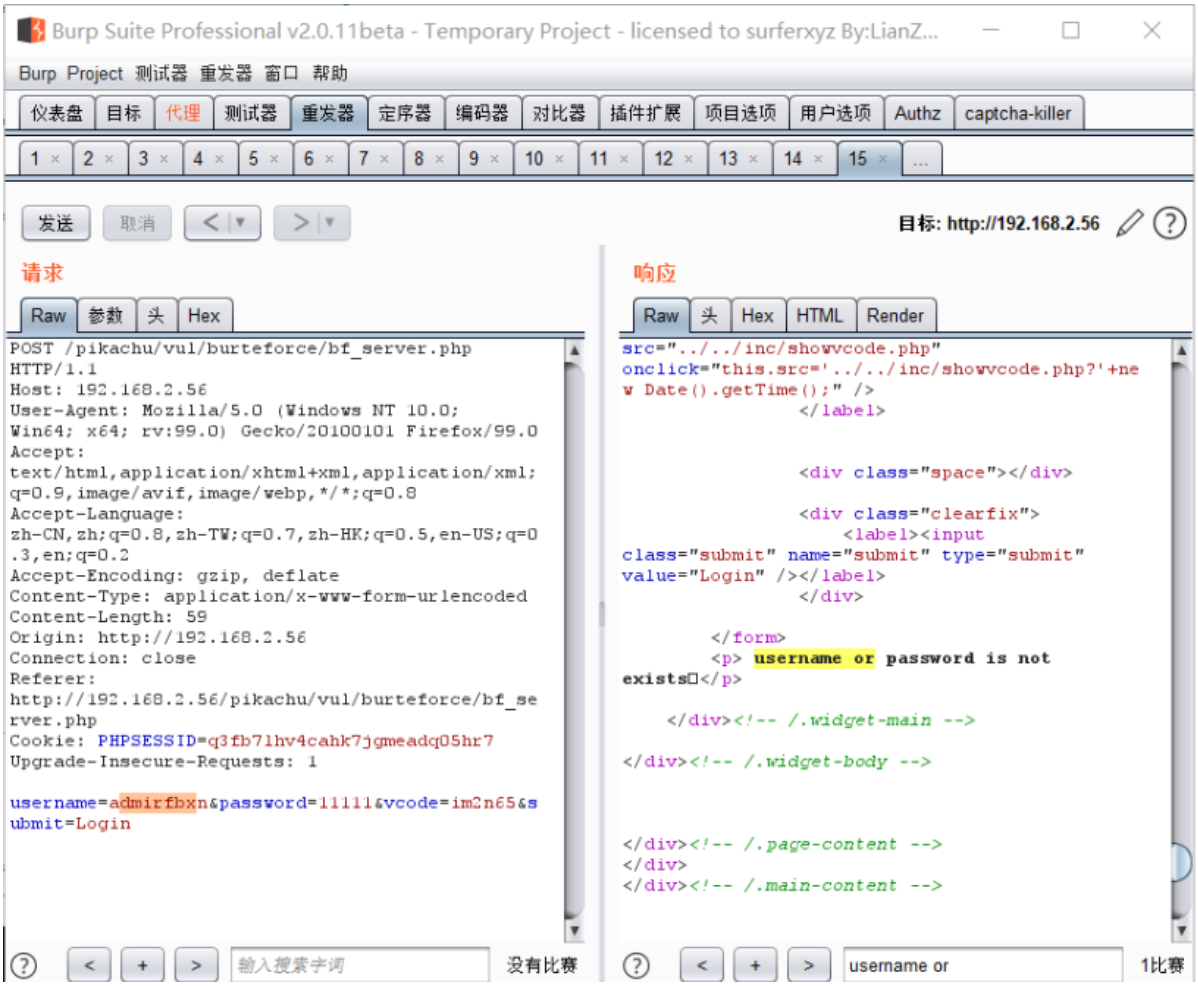
随便输入账号密码，输入正确的验证码，提交， 提示用户名或密码不存在：



抓包正确的验证码，查看返回，验证码正确：



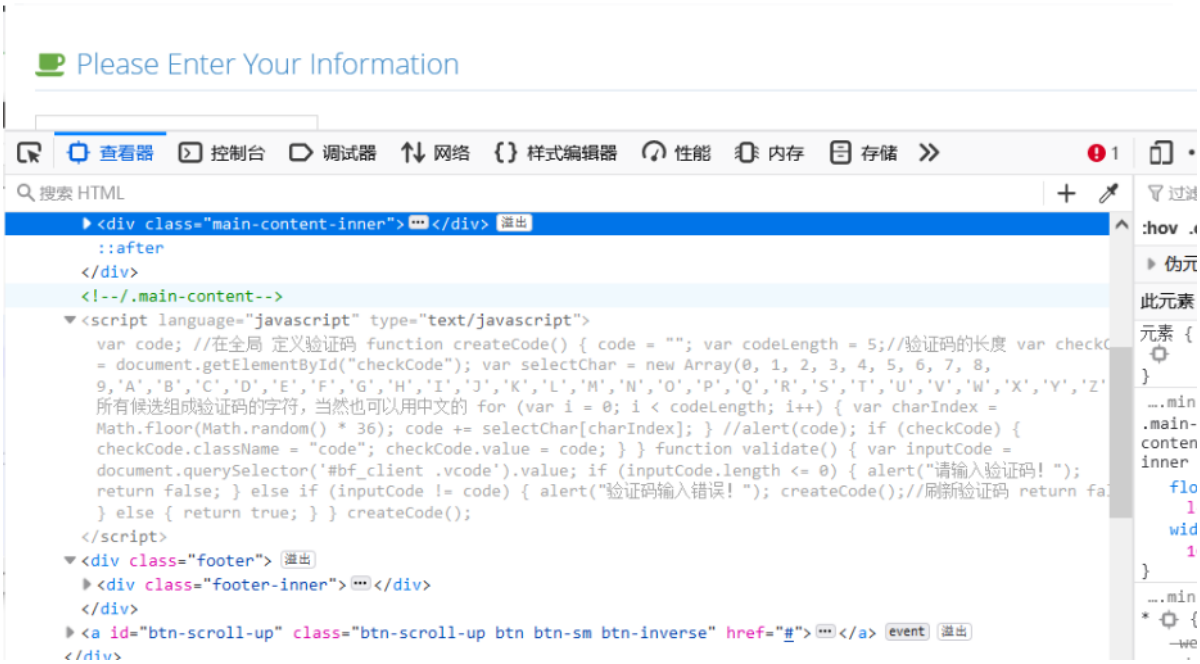
修改用户名发送，还是提示用户和密码不存在，没有提示验证码错误，可以复用：



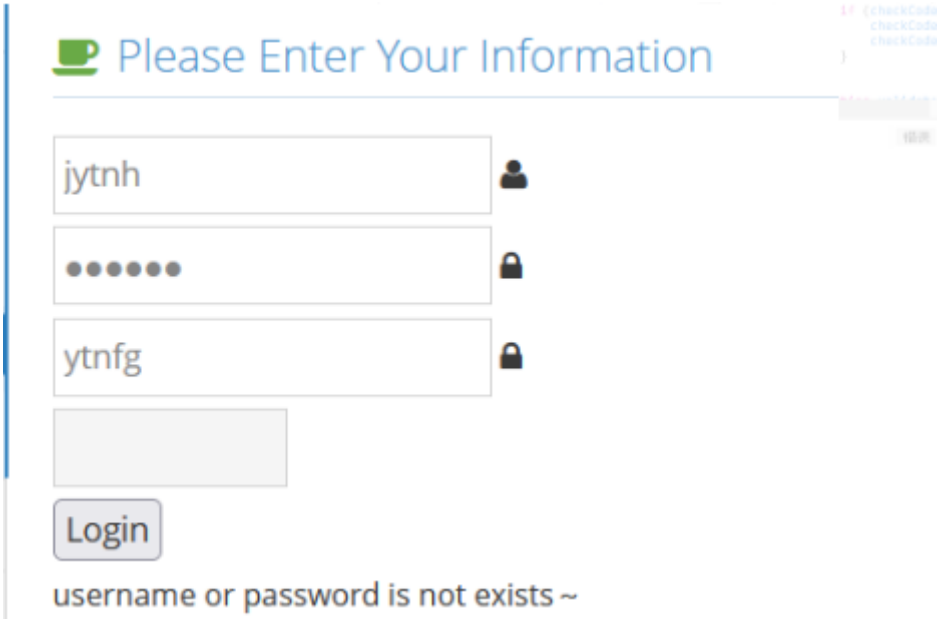
1 思路：将数据包的用户名修改重复发送看回显的值是否发生变化。可以看出验证码存在复用的问题。
2 利用：输入正确的账号和密码，用之前的额验证码，发下登录成功
3 源代码分析
4 ---传递的参数都不能为空
5 ---post提交的验证码也要和session中的vcode相等
6 ---漏洞存在原因：session中的vcode没有销毁，导致下一次还能用

2.on client

查看前端源代码（验证码写入了前端）：



这里随意输入验证码绕过（前端代码验证码不执行）：



36.2 token 安全

36.2.1 Token定义

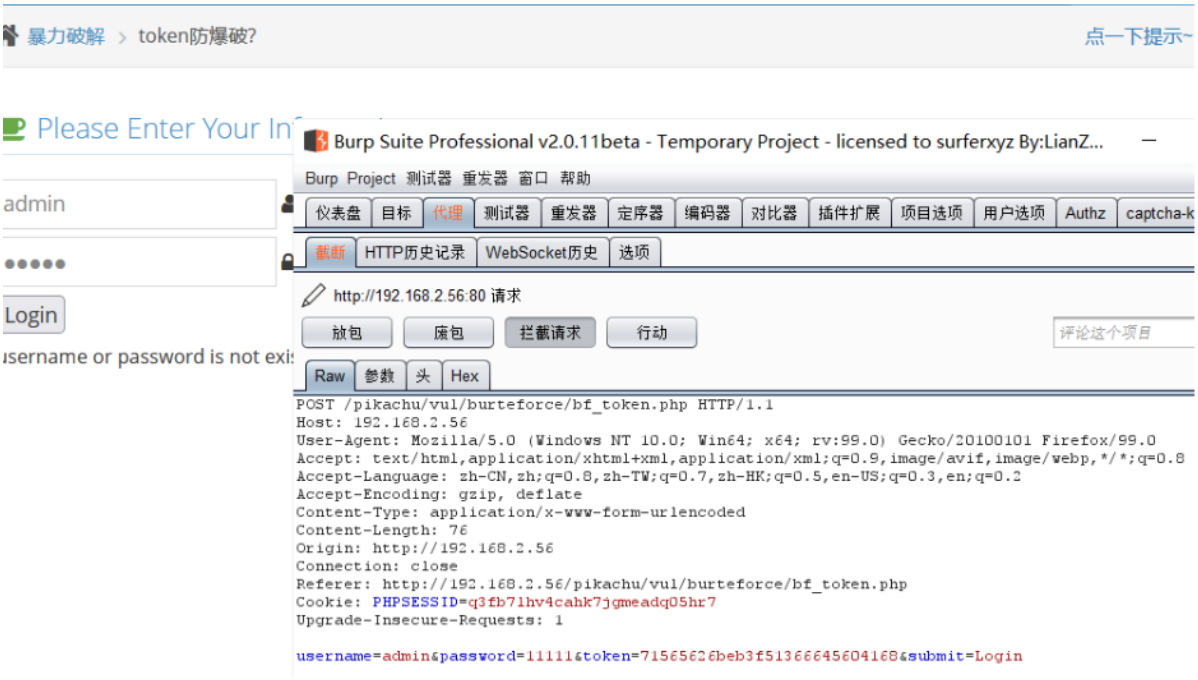
Token是服务端生成的一串字符串。当客户端第一次登录后，服务器生成一个Token并将此Token返回给客户端，Token可以代替用户名和密码作为身份的验证。每次客户端与服务器端通信，会得到新的Token，基本上述同理，主要是验证中可存在绕过可继续后续测试。

36.2.2 token 爆破，token 客户端回显等

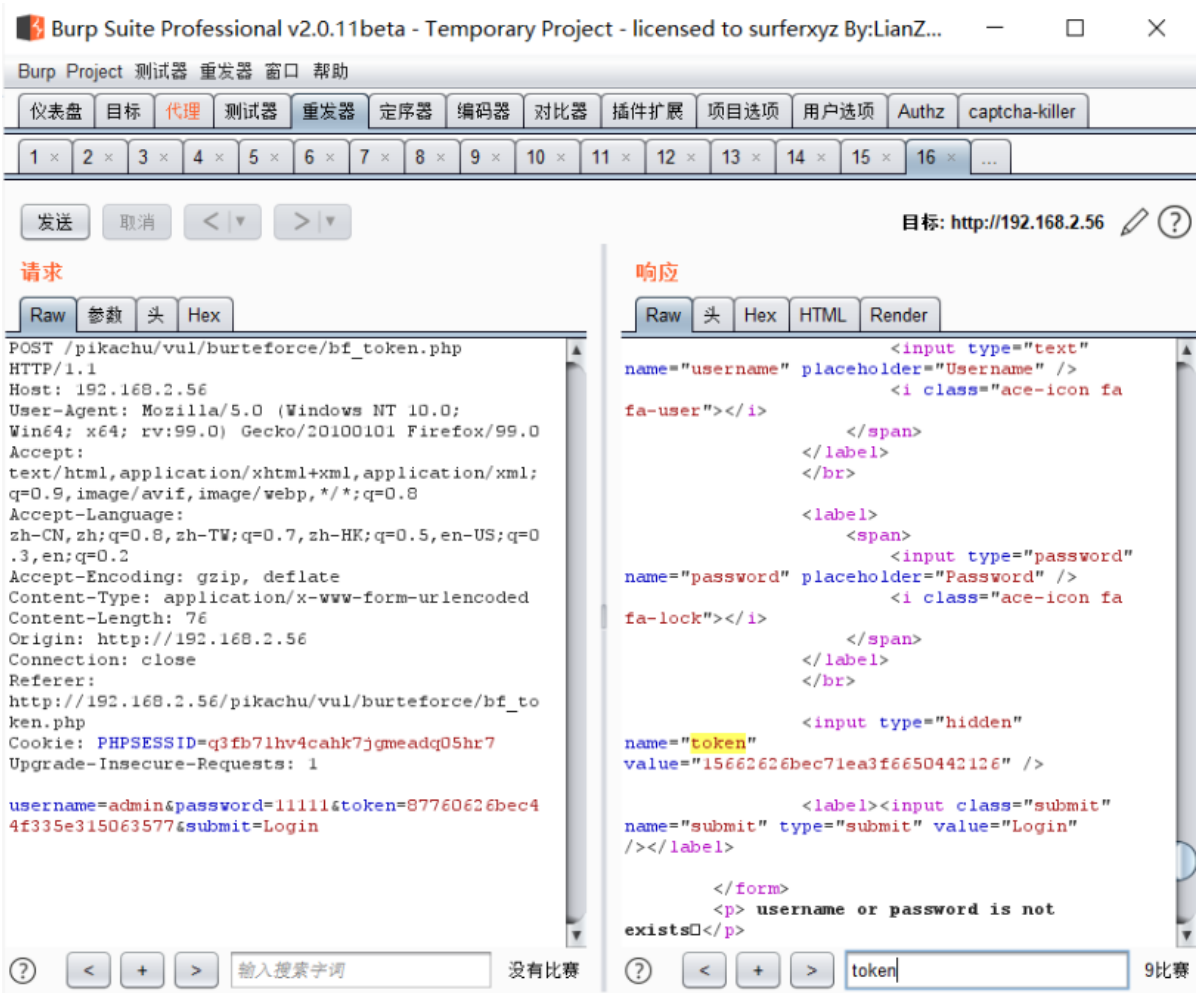
1. token 爆破---token后面会跟上一个字符串，如果知道规律可以进行爆破
2. token 客户端回显---token的数据会在前端数据包（request）里面显示
3. token固定---虽然有token，但是可以通过上一次的token操作下一次的数据包（表面上有，实际没有）

36.2.3 Token靶场练习

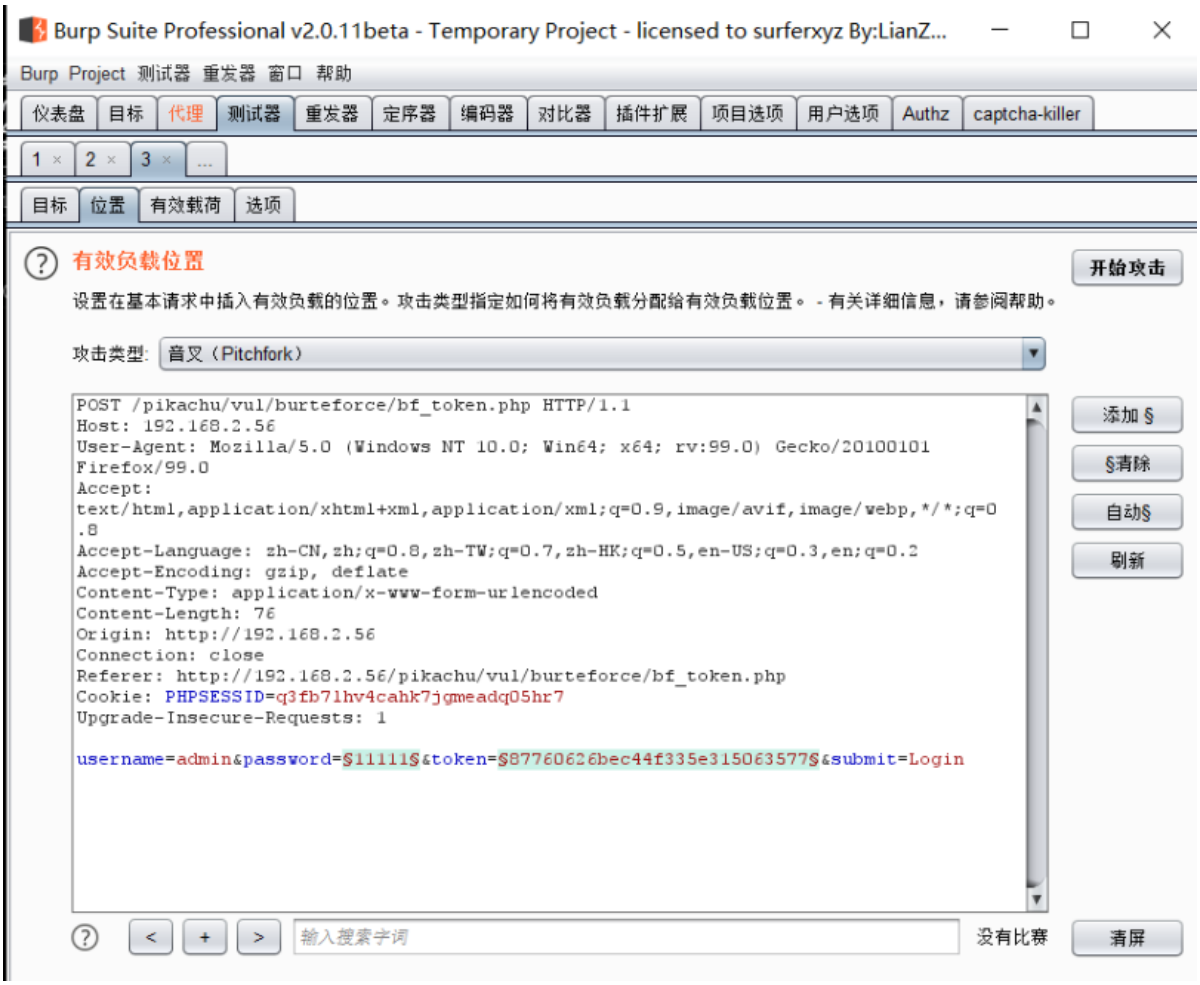
登录查看数据包：



查看前端里也有token，该token是下一次客户端token正确的匹配值：



token是用来防爆破的，但是其token值输出在了前端源码中，容易被获取，因此也就失去了防暴力破解的意义。将密码和token设置为攻击参数，攻击类型选择pitchfork：

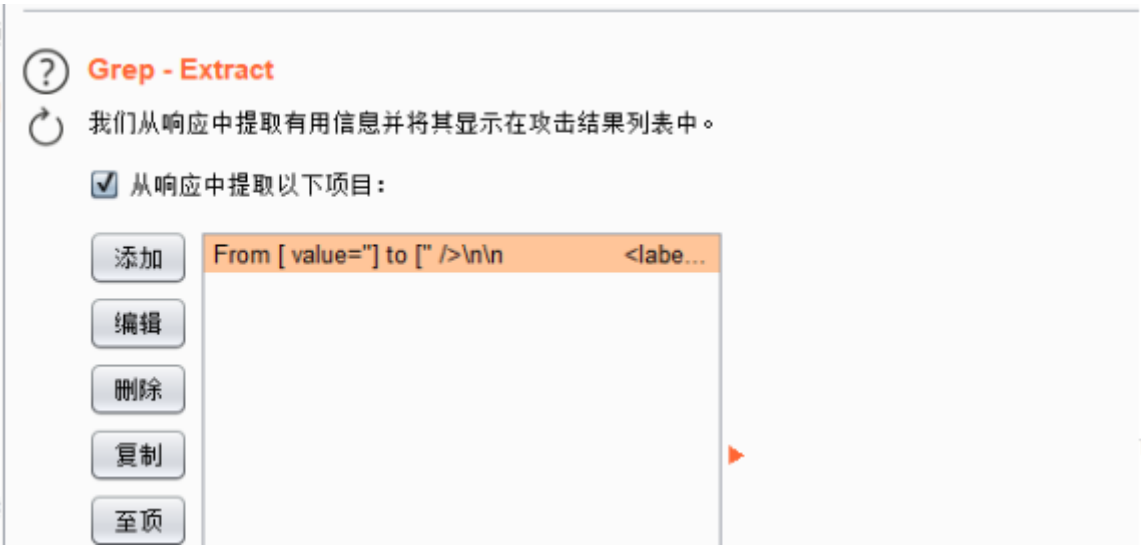


设置第一个参数，添加字典：

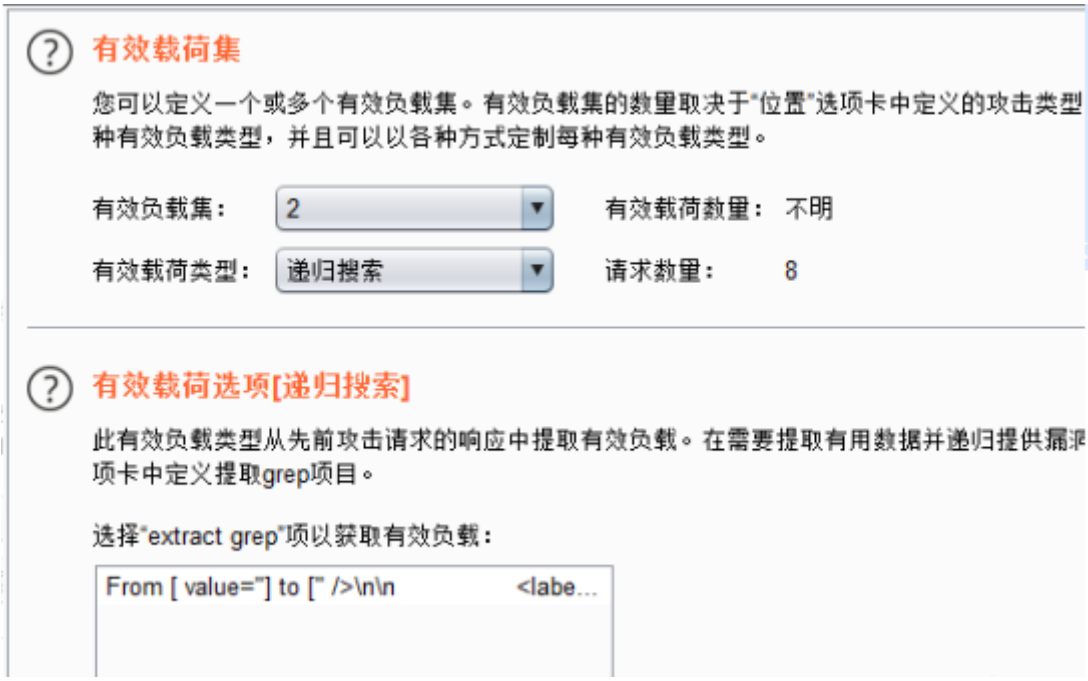
设置线程为1（1次只验证一个账号密码）：

设置重定向，仅限范围：

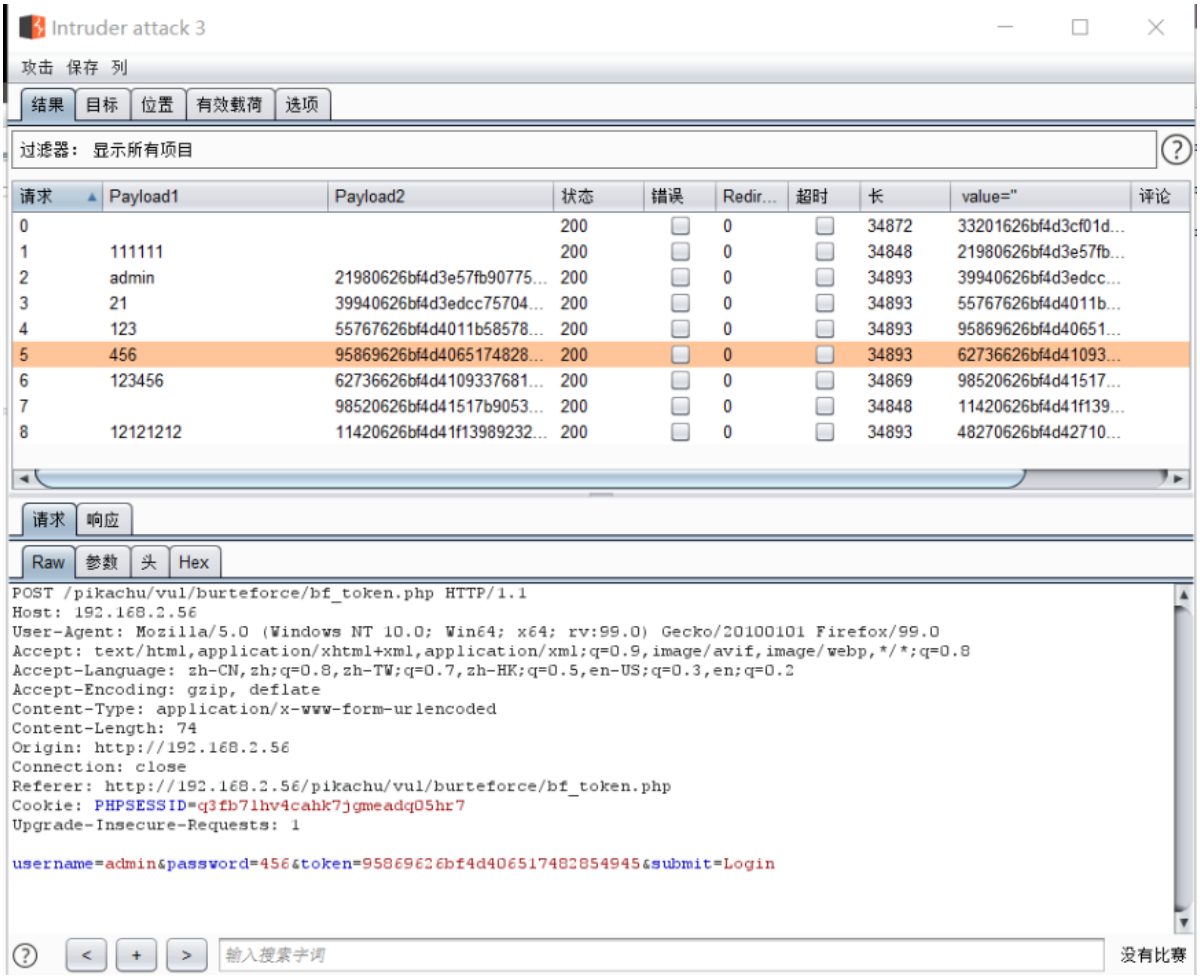
设置grep extract匹配用法，点击添加，查找到token值选择，点击ok：



这里设置第二个参数token，类型为recrusive grep：



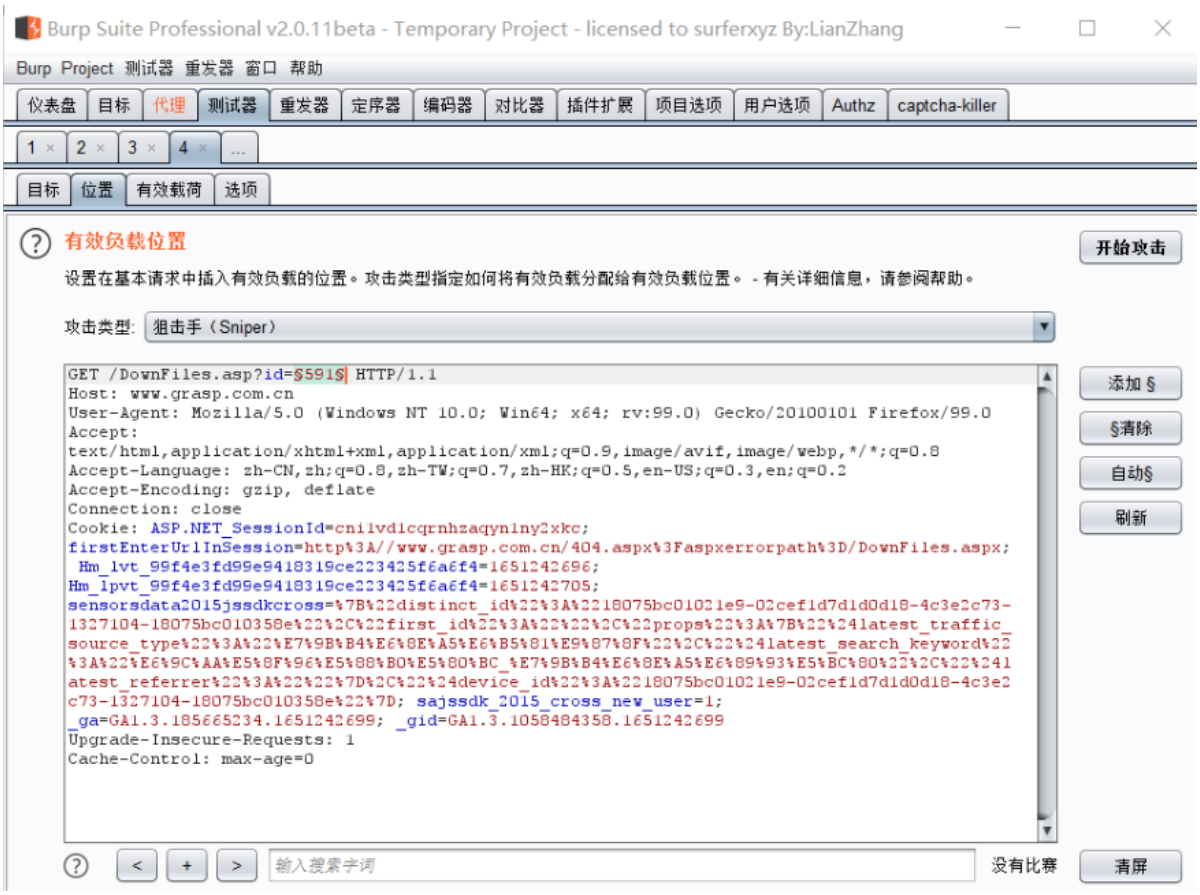
攻击，发现下面的token值就是上一个的值：



36.3 本地某URL下载接口ID值调用遍历测试-实例

```
1  url:www.grasp.com.cn/DownFiles.asp?id=591
```

抓包发送给intruder，设置id为变量:



设置攻击类型和攻击范围:

查看结果，看是否可以发现其他用户的信息，用来水平越权:

Request	Payload	Status	Error	Timeout	Length	Comment
752	752	302			444	
753	753	302			444	
754	754	302			444	
808	808	302			444	
854	854	302			444	
864	864	302			444	
229	229	302			490	
230	230	302			512	
163	163	302			516	
164	164	302			516	
165	165	302			516	
166	166	302			516	

36.4 Callback自定义返回调用安全-漏洞测试-实例

36.4.1 什么是callback

一般而言，函数的形参是指由外往内向函数体传递变量的入口，但此处加了callback后则完全相反，它是指函数体在完成某种使命后调用外部函数的出口！这时候应该明白什么叫"回调"了吧，也就是回头调用外部函数的意思。

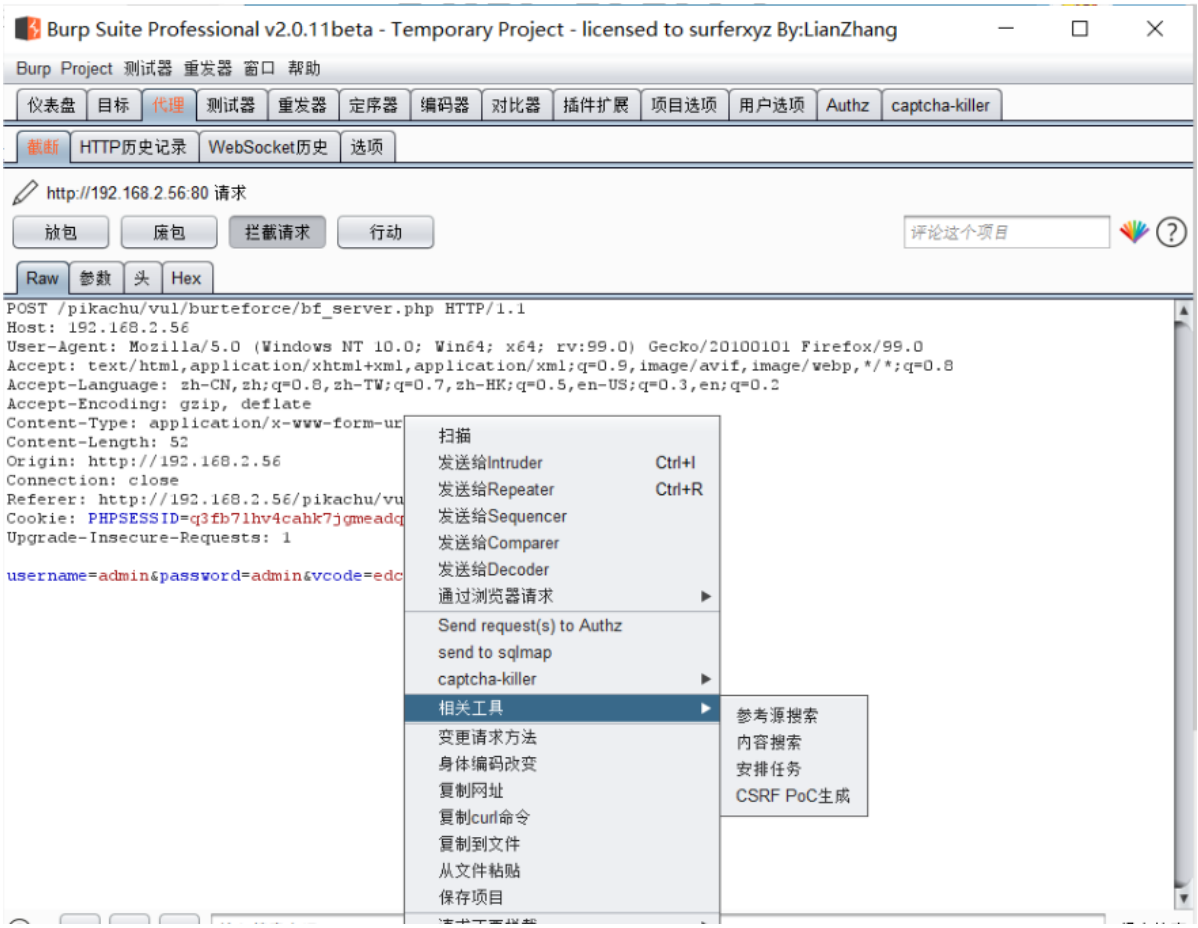
36.4.2 漏洞利用

callback参数可以更改，可以和跨站漏洞结合，在网页源代码搜索传递的参数，如果存在，意味着URL传递的参数会在网页的前端回显，那么，也意味着可以构造XSS漏洞（测试有没有过滤，完不完整）

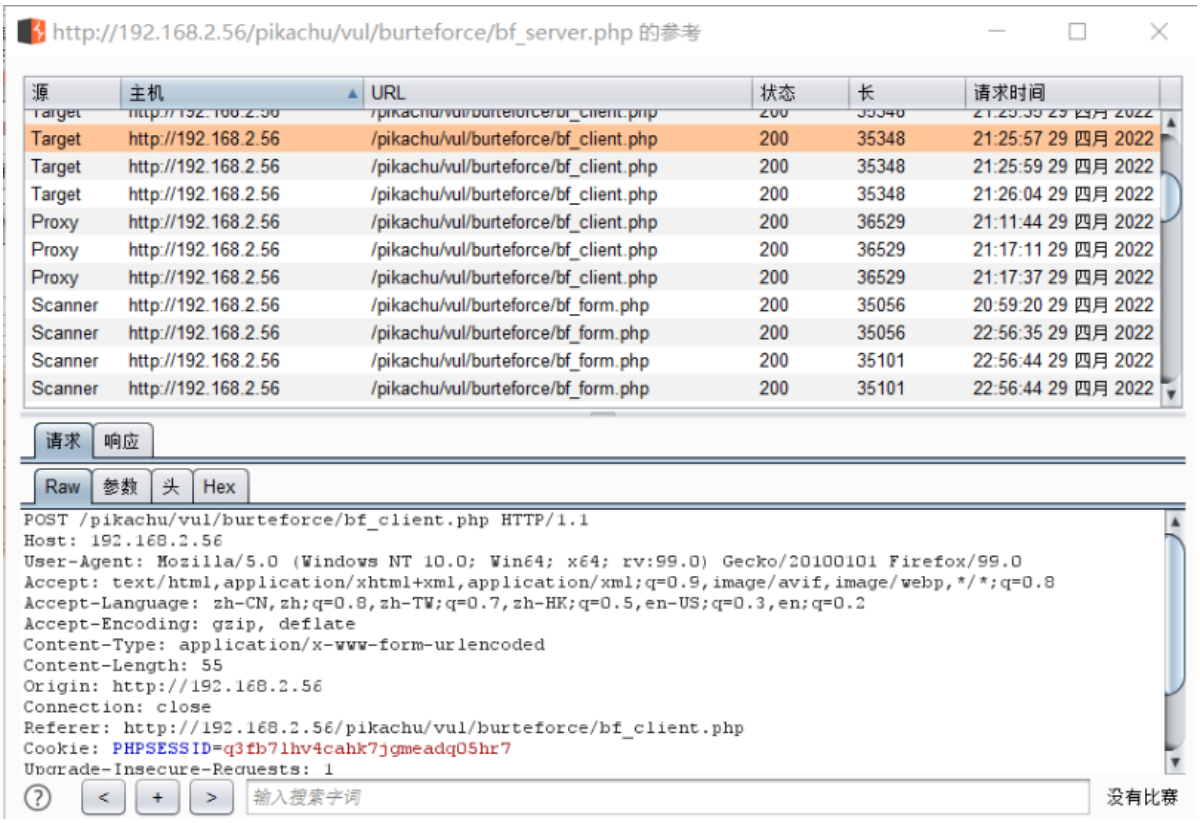
```
1  上述在实战中如何做到漏洞发现-bp功能点
2  原理：逻辑漏洞挖功能点和参数值（关键的参数：id，
   callback，filename，uid等等）
```

36.5 保存所需数据包

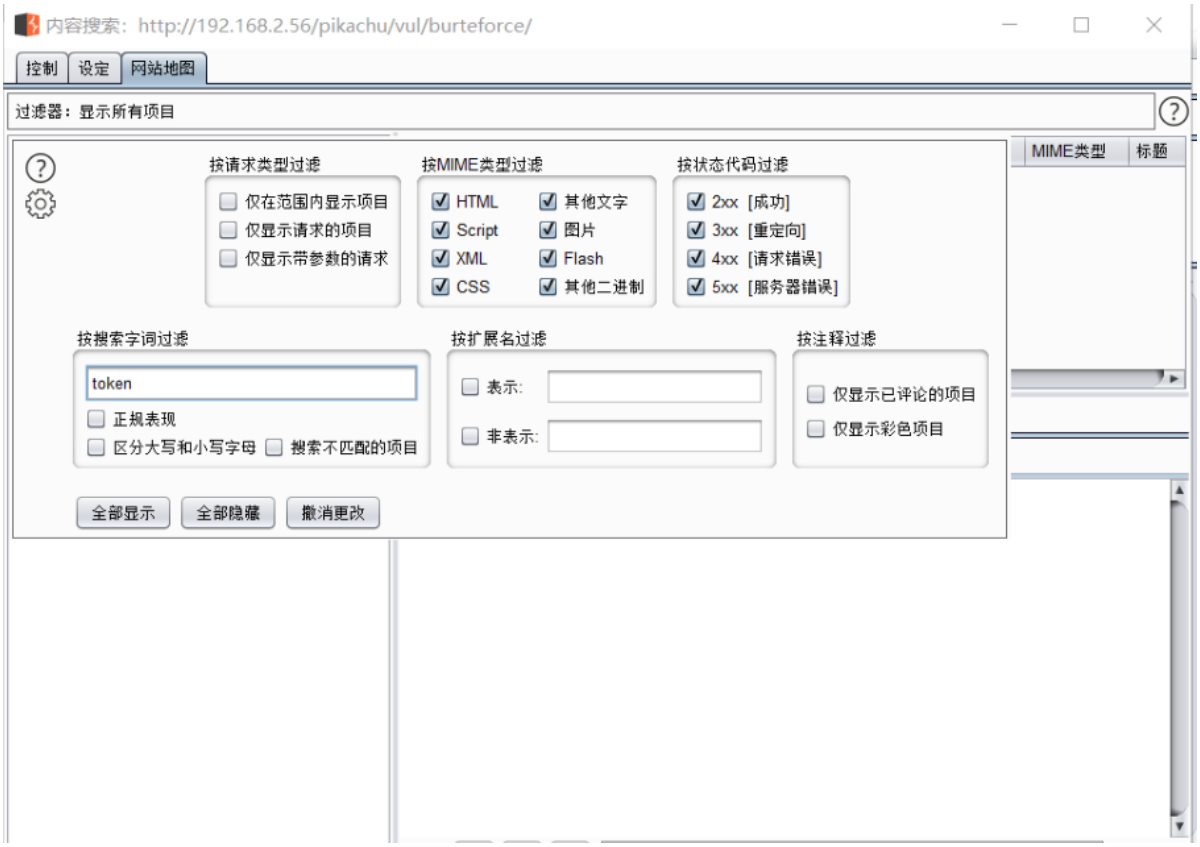
抓包发送到爬虫（这里使用右键点击tools中的find references）：

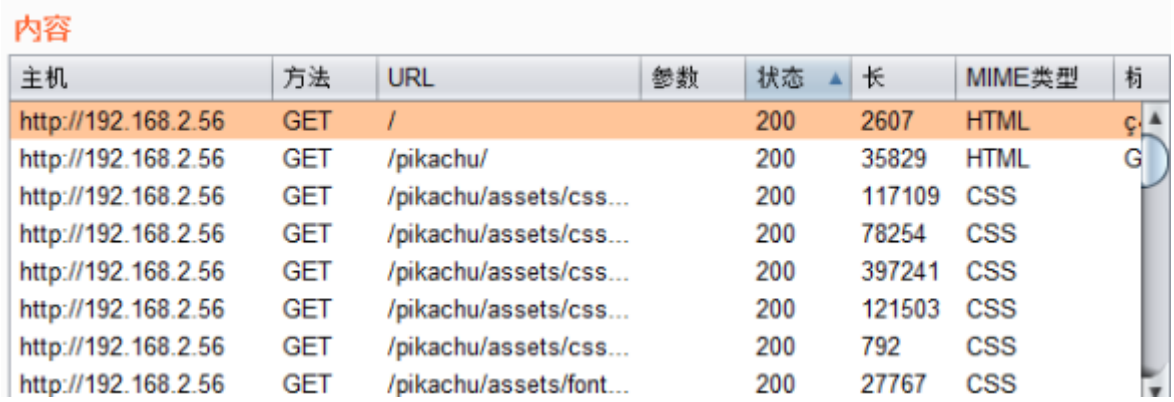


查看抓到的结果：

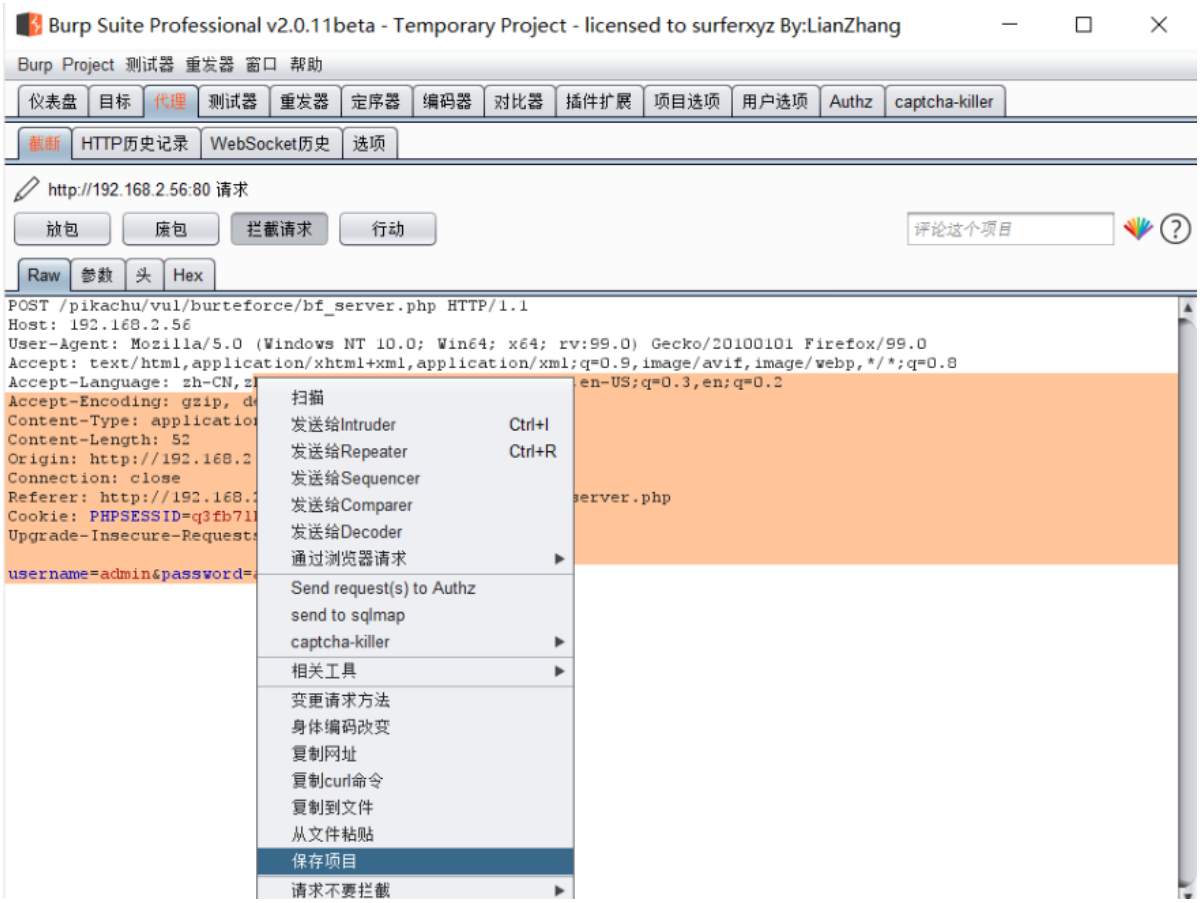


真实目录路径，搜索关键字（这里可以使用 discover content模块过滤搜索内容，相当于帮你筛选）：





导出数据包（save bom）：



资源：

```
https://www.lanzous.com/i1z2s3e
https://www.cnblogs.com/nul1/p/12071115.html
https://github.com/c0ny1/captcha-killer/releases/tag/0.1.2
https://github.com/bit4woo/reCAPTCHA/releases/tag/v1.0
```