史上最详细！逻辑漏洞全方位总结

wangkun05 白帽子左一 2022-12-09 12:33 发表于江西

作者：freebuf-wangkun05

一、水平越权

1.1、原理

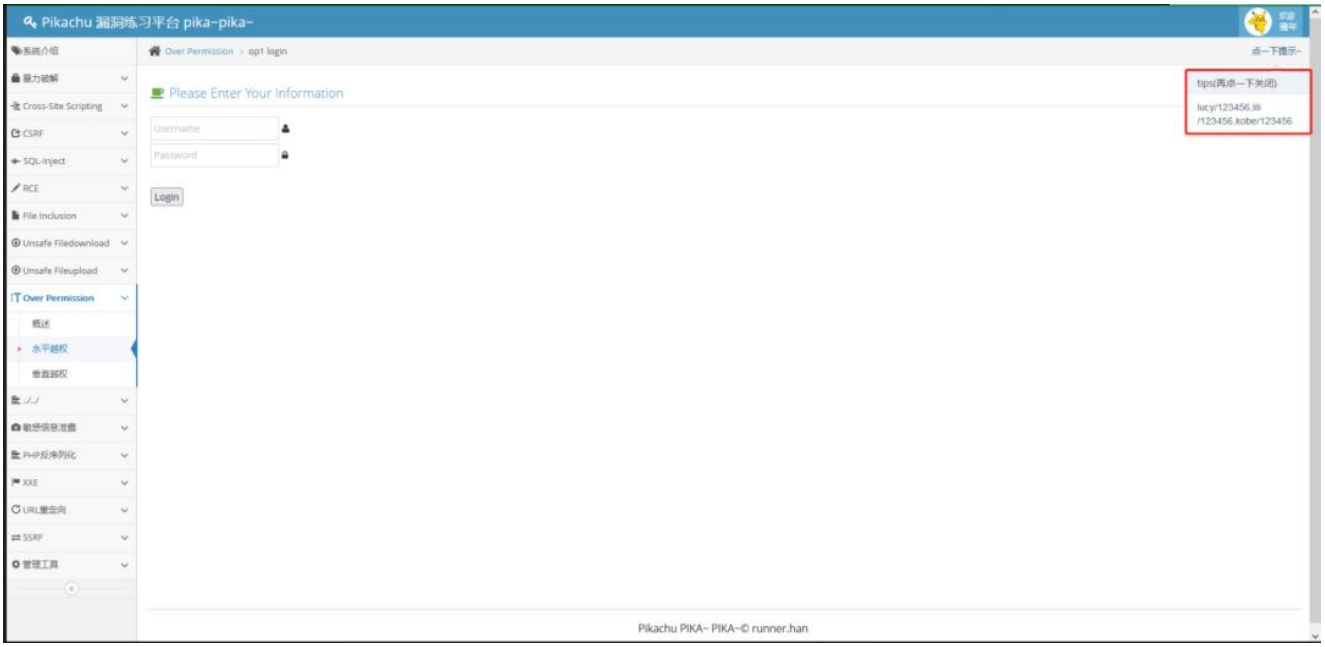
通过更换某个ID之类的身份标识，从而使得A账号获取(修改，删除等)B账号的数据；

1.2、容易出现的地方：

一般越权漏洞容易出现在权限页面(需要登陆的页面)增，删，改，查的地方，当用户对权限页面内的信息进行这些操作时，后台需要对当前用户的权限进行校验，看其是否具备操作权限，从而给出响应，而如果校验的规则过于简单则容易出现越权漏洞；

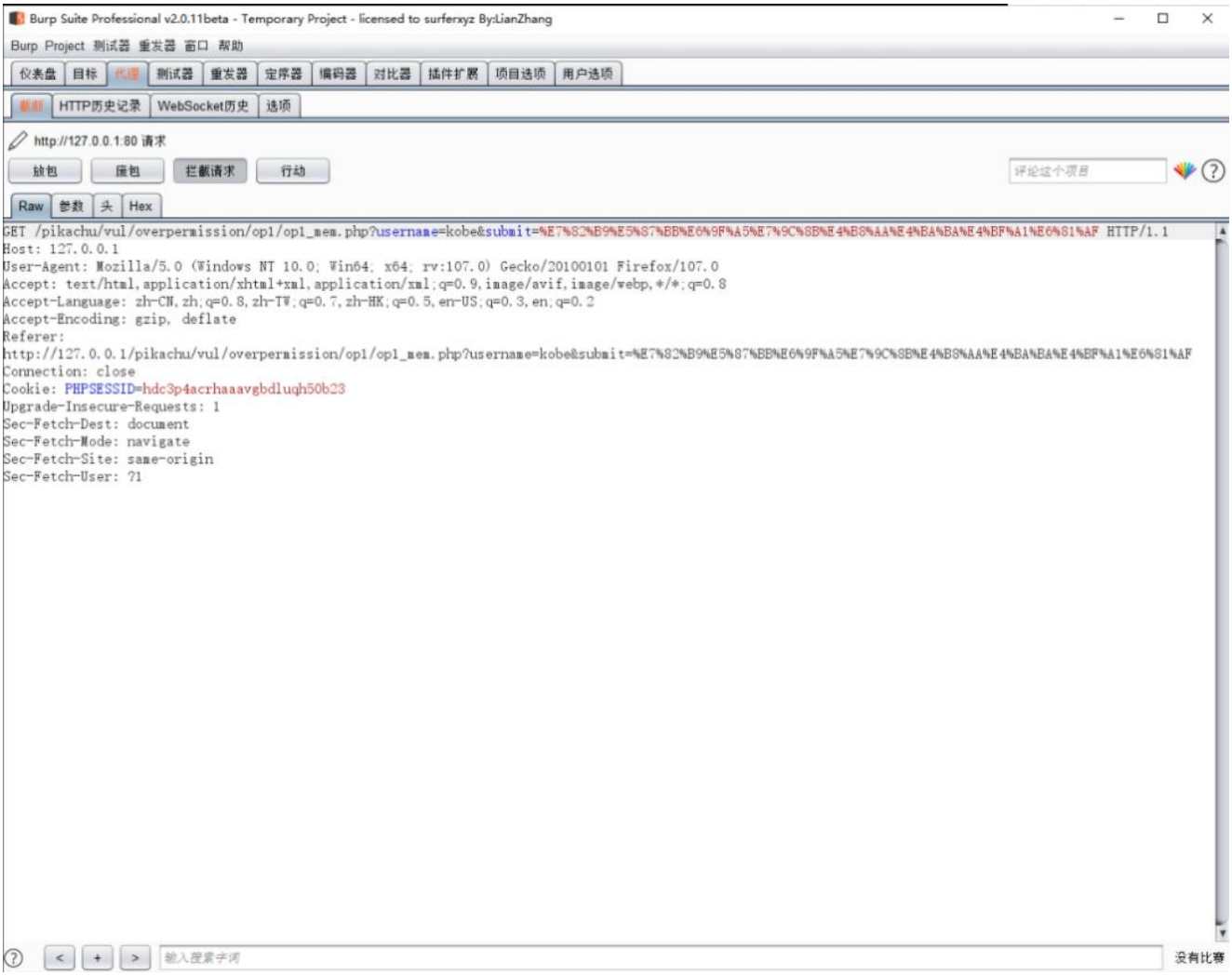
1.3、案例演示

1）我们登陆到pikachu靶场，首先看一下提示：



2）一共两个用户，我们登陆到kobe用户，来测试越权到luck用户，我们首先登录到kobe账户里面，点击查看个人信息，抓到包；

3）我们将这里的kobe改为lucy，放包；



4）成功越权到lucy用户

1.4、危害

在游戏中，假如我们是平民玩家，我们仅仅通过修改ID，就变成了其他玩家，甚至有可能变成氪金大佬；

二、垂直越权

2.1、原理

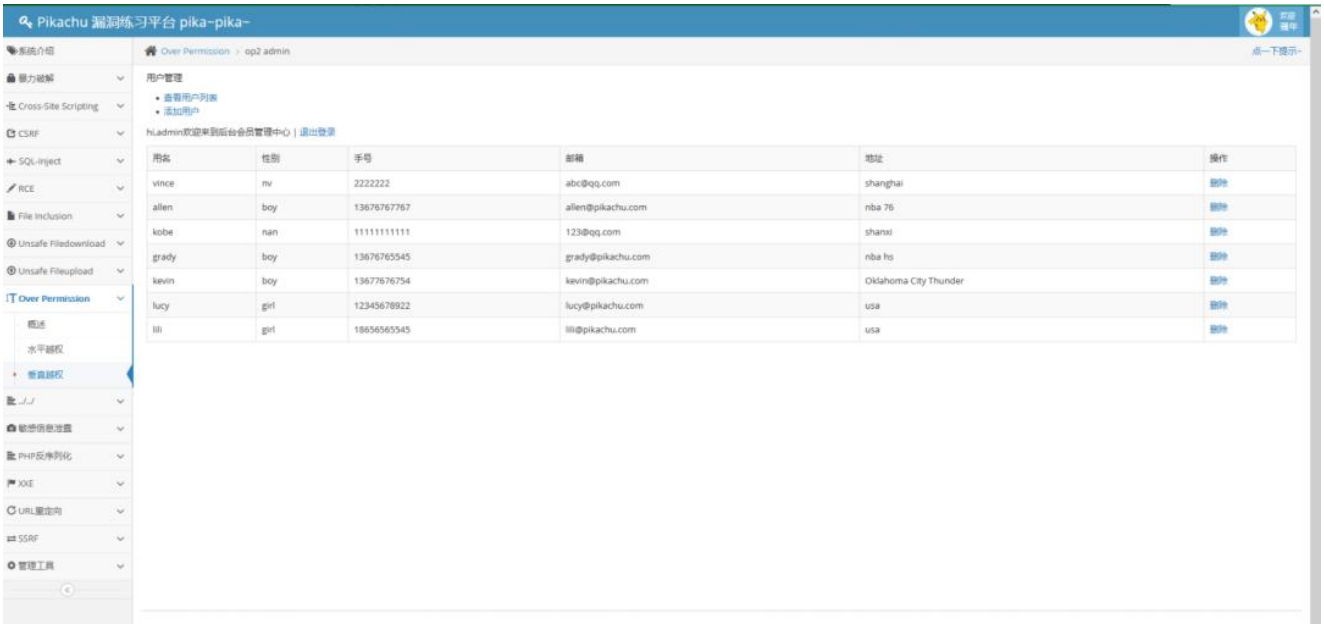
使用低权限身份的账号，发送高权限账号才能有的请求，获得其高权限的操作；

2.2、容易出现的地方

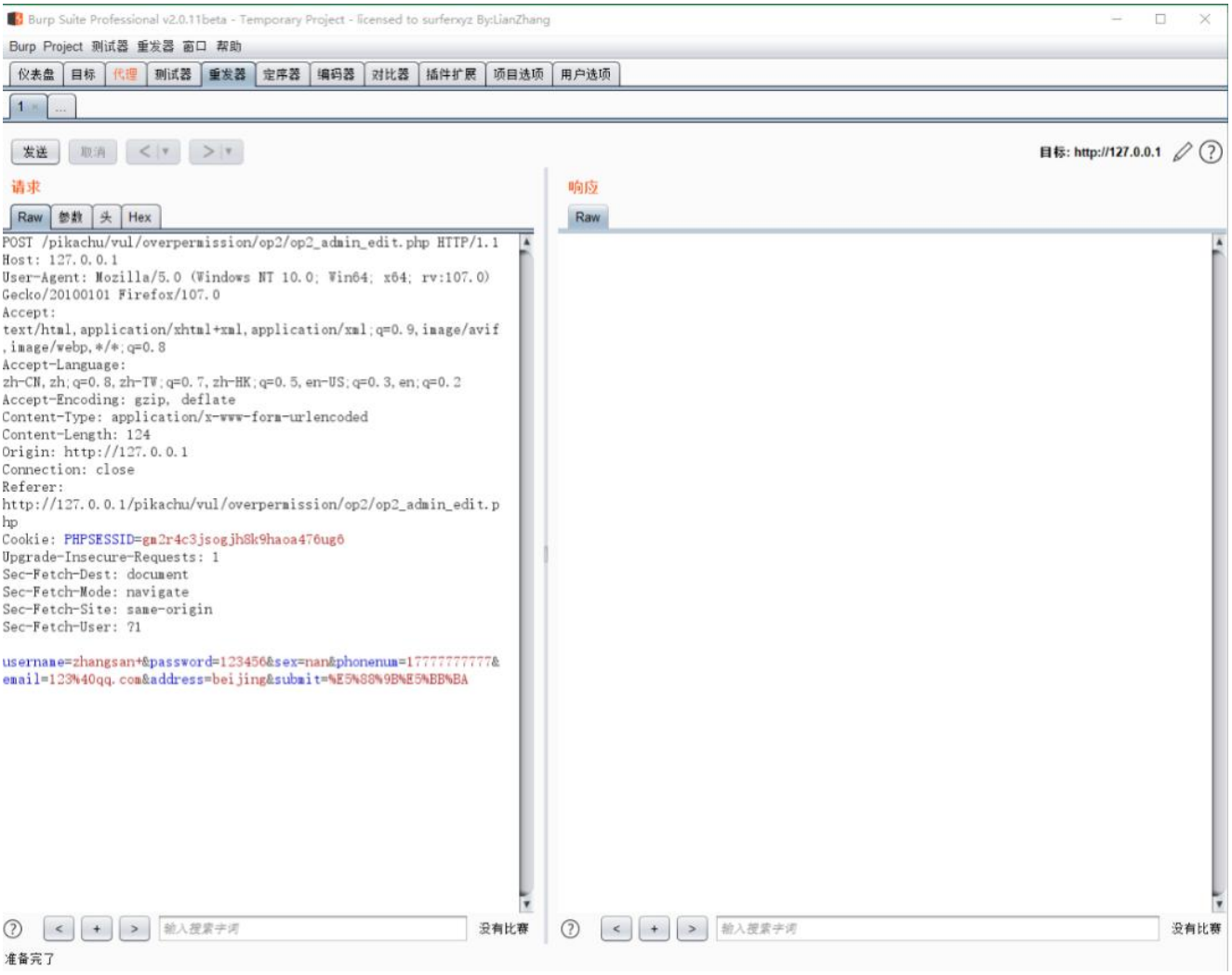
看看低权限用户是否能越权使用高权限用户的功能，比如普通用户可以使用管理员的功能；

2.3、案例演示

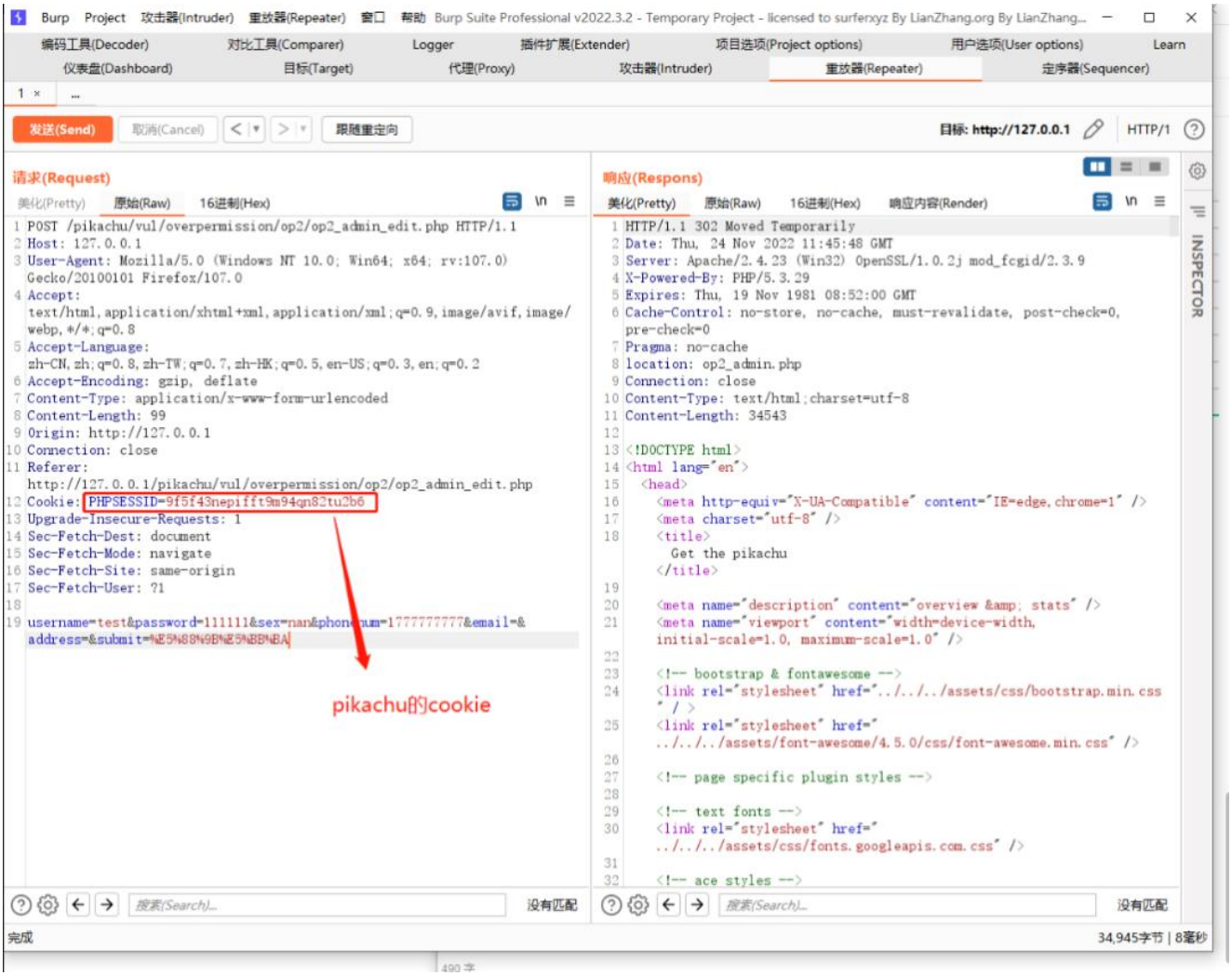
1）我们首先用admin/123456，进行登陆；



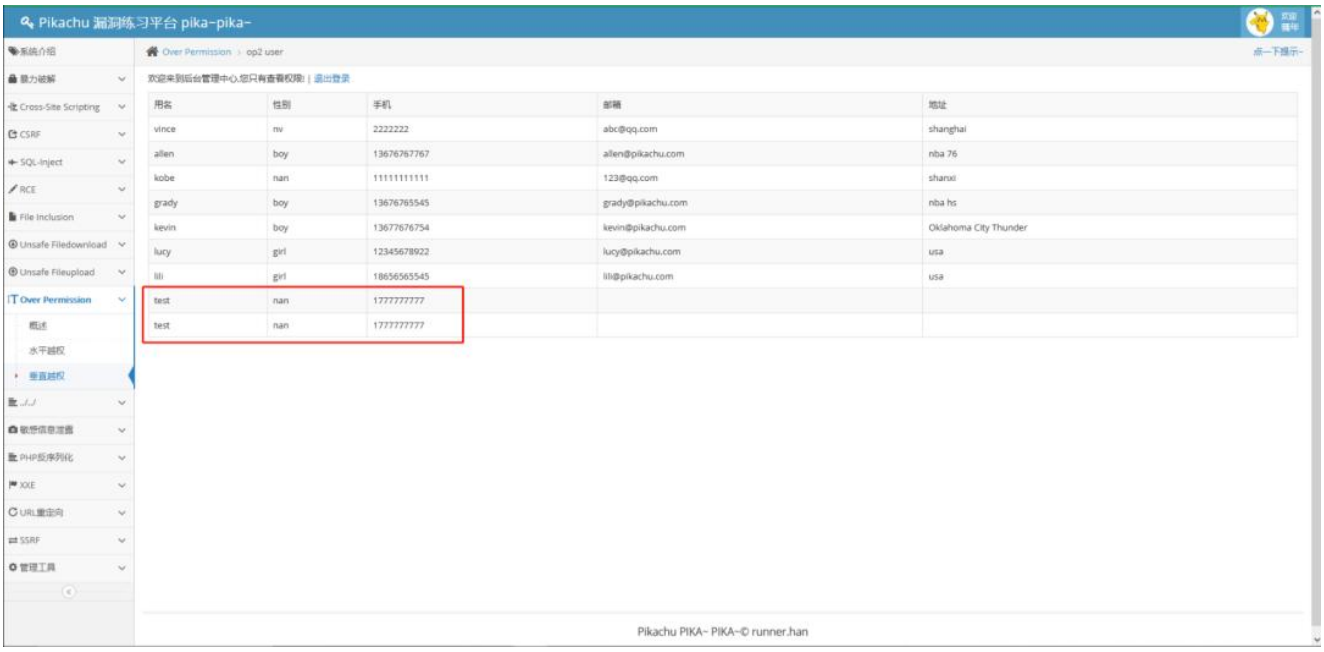
2）我们添加用户进行抓包，然后放在重发器；



3）我们紧接着用pikachu/000000，进行登陆，抓包，将pikachu的cookie替换admin用户的cookie；



4）我们发送包；



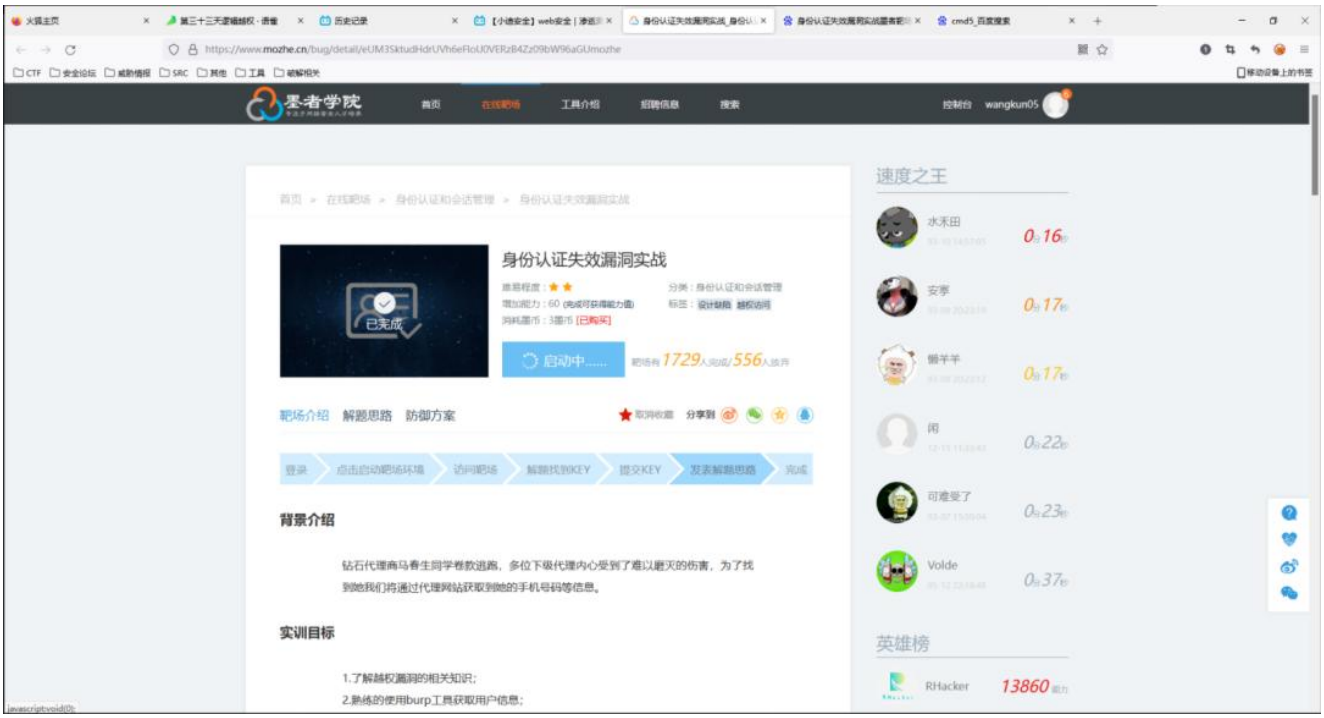
利用pikachu用户利用admin的cookie成功创建了test账户；

2.4、危害

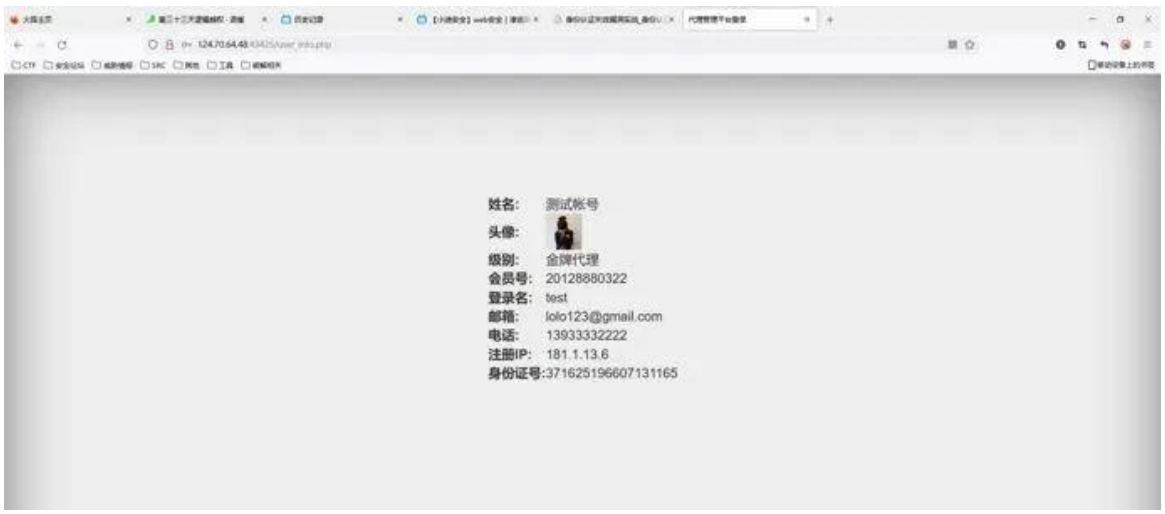
信息泄露，篡改用户信息，严重者可修改密码等等；

三、墨者靶场演示

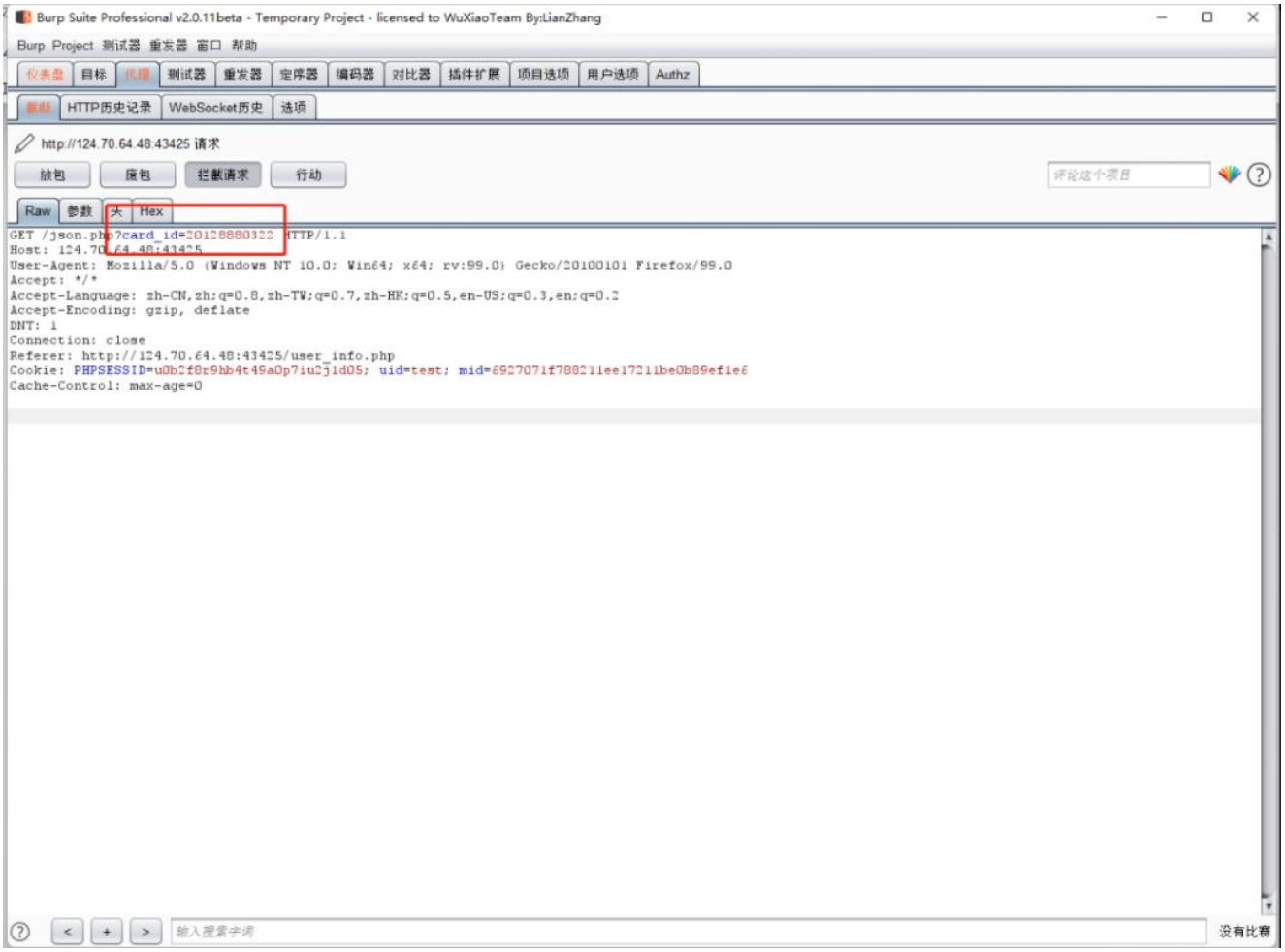
1）打开靶场



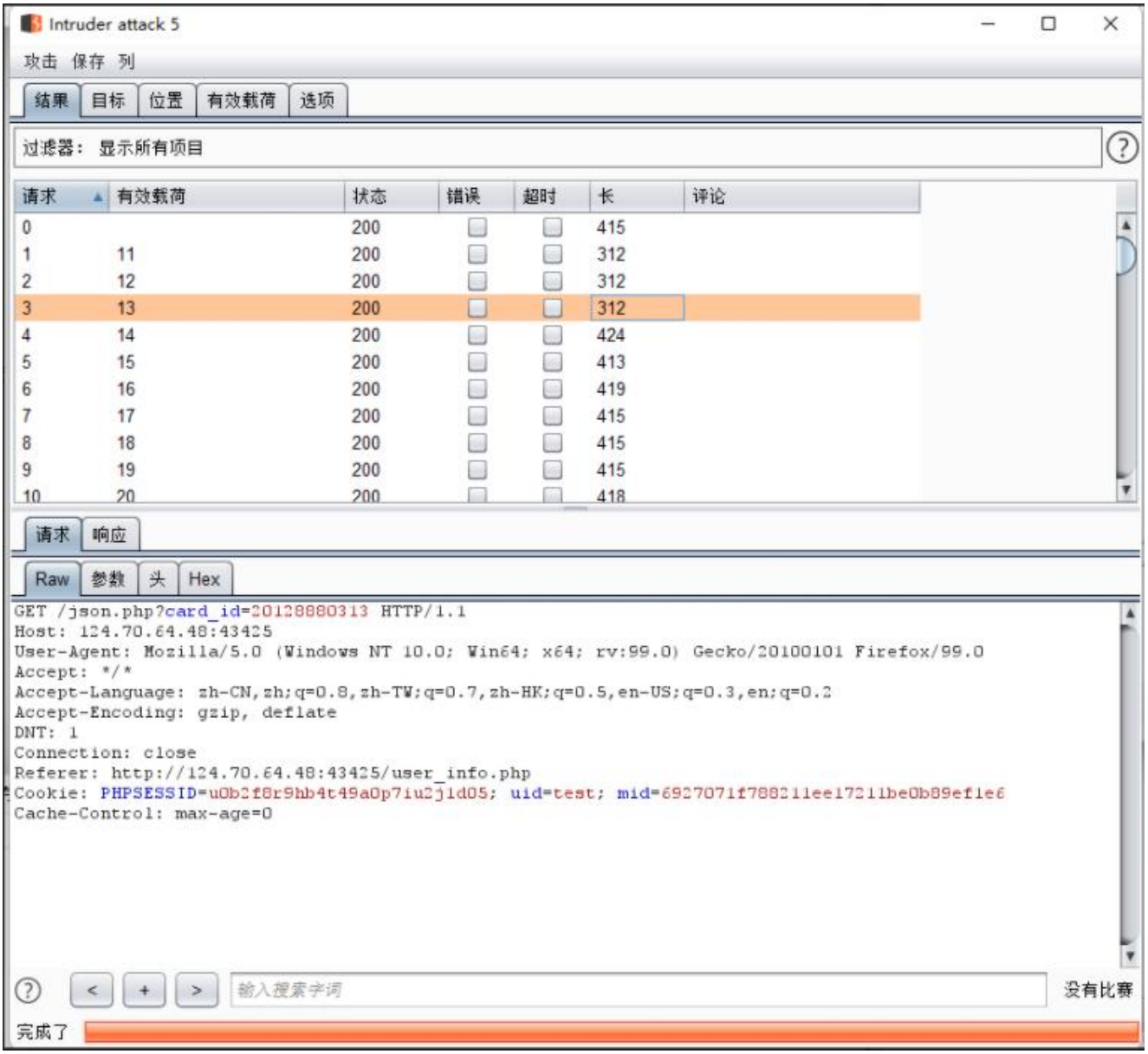
2）登录给出的test用户



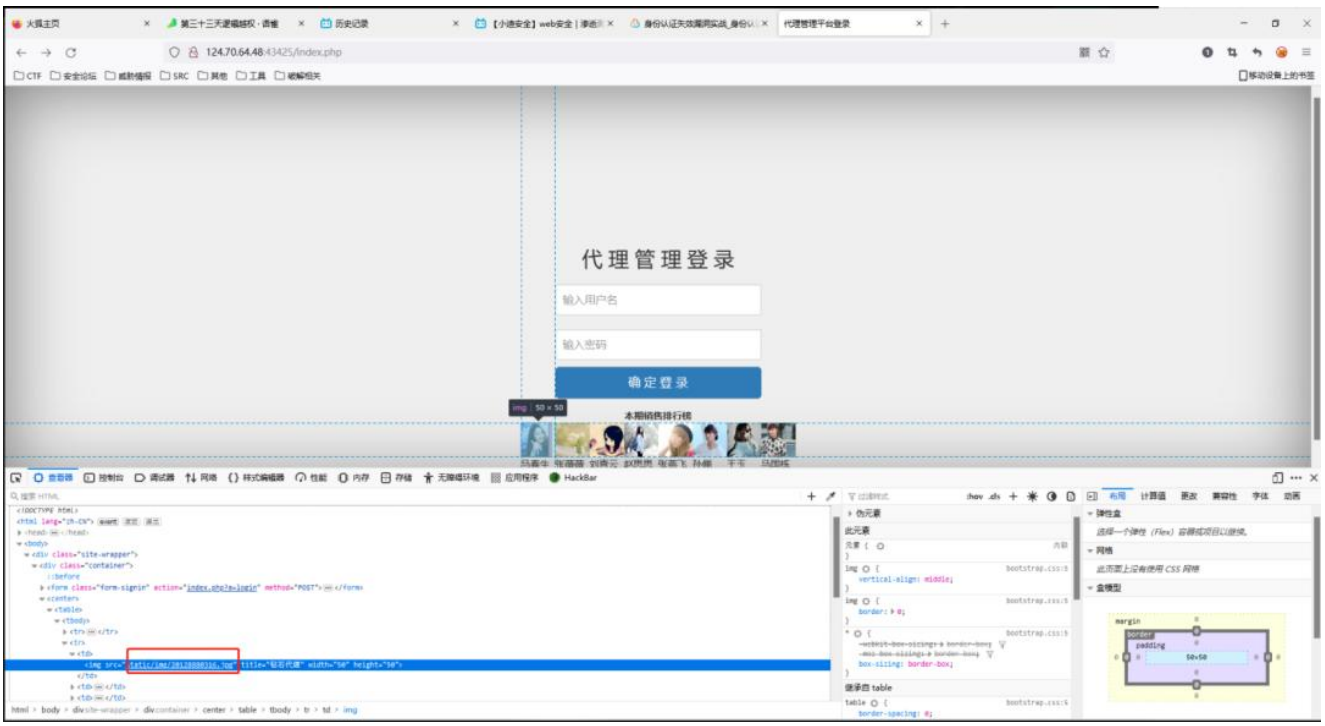
3）我们这里通过抓包进行分析，对card_id进行分析


Burp Suite Professional v2.0.11beta - Temporary Project - licensed to WuXiaoTeam By:LianZhang
Burp Project 测试器 重发器 窗口 帮助
仪表盘 目标 代理 测试器 重发器 定序器 编码器 对比器 插件扩展 项目选项 用户选项 Authz
截断 HTTP历史记录 WebSocket历史 选项
http://124.70.64.48:43425 请求
放包 丢包 拦截请求 行动
Raw 参数 头 Hex
GET /json.php?card_id=20128880322 HTTP/1.1
Host: 124.70.64.48:43425
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: http://124.70.64.48:43425/user_info.php
Cookie: PHPSESSID=u0b2f8r9hb4t49aOp7iu2j1d05; uid=test; mid=6927071f78821lee1721lbeOb89efie6
Cache-Control: max-age=0
输入搜索字词
没有比赛

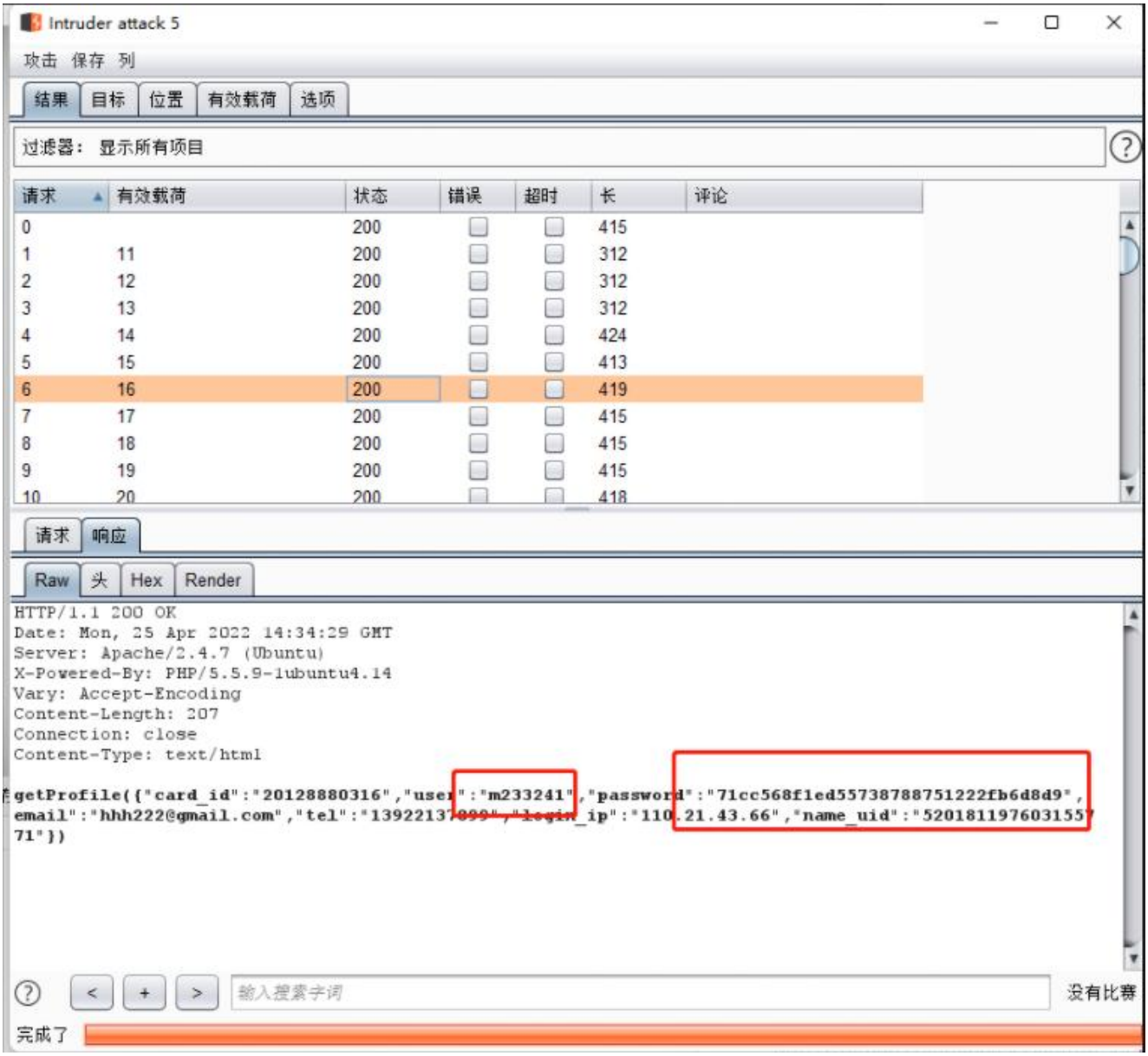
4）发到测试器进行爆破



通过前端分析，他可能是316

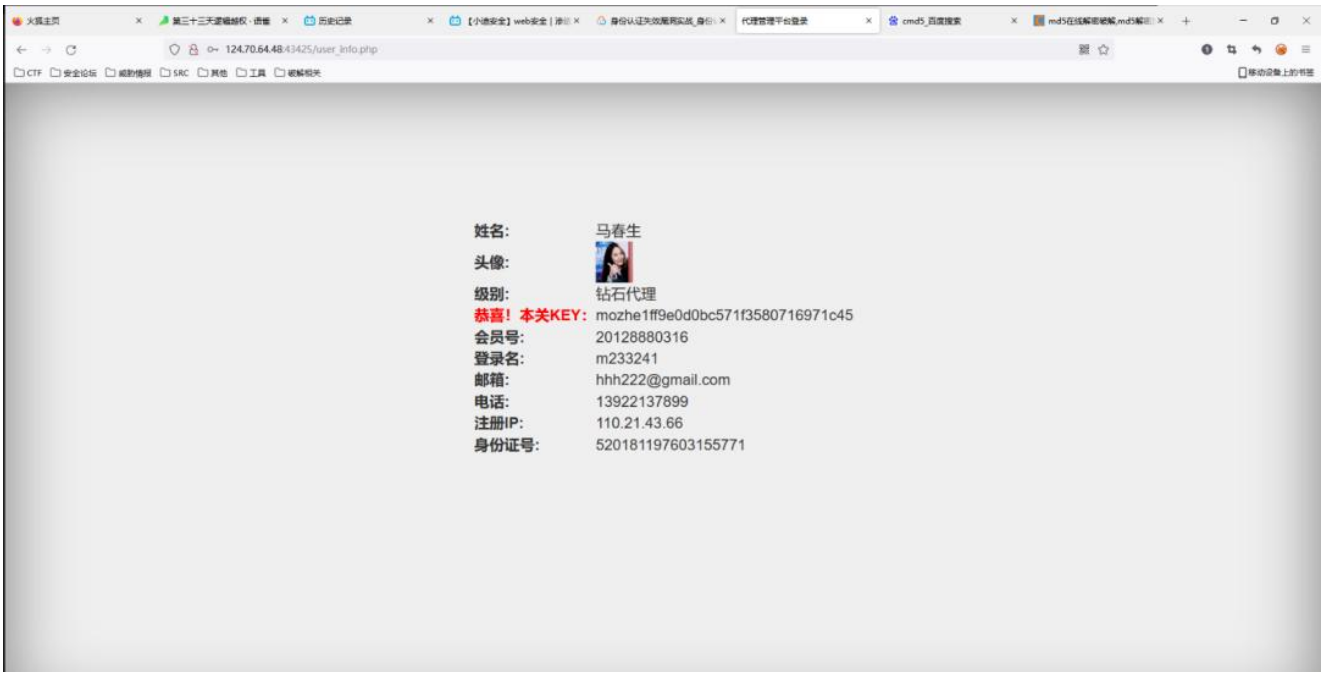


5）看响应



输入账号，对密码进行MD5解密

6）登录成功



四、越权漏洞检测-burp插件-autorize使用

Autorize是一个旨在帮助渗透测试人员检测授权漏洞的扩展，这是Web应用程序渗透测试中比较耗时的任务之一；

将低权限用户的cookie提供给扩展程序并使用高权限用户浏览网站就足够了，该扩展会自动重复每一个请求与低权限用户的会话并检测授权漏洞；

除了授权漏洞之外，还可以在没有任何cookie的情况下重复每一个请求，以检测身份验证漏洞；

该插件无需任何配置即可工作，但也是高度可定制的，允许配置授权执行条件的粒度以及插件必须测试那些请求，那些不需要，可以保存插件的状态并以HTML或CSV格式导出授权测试报告；

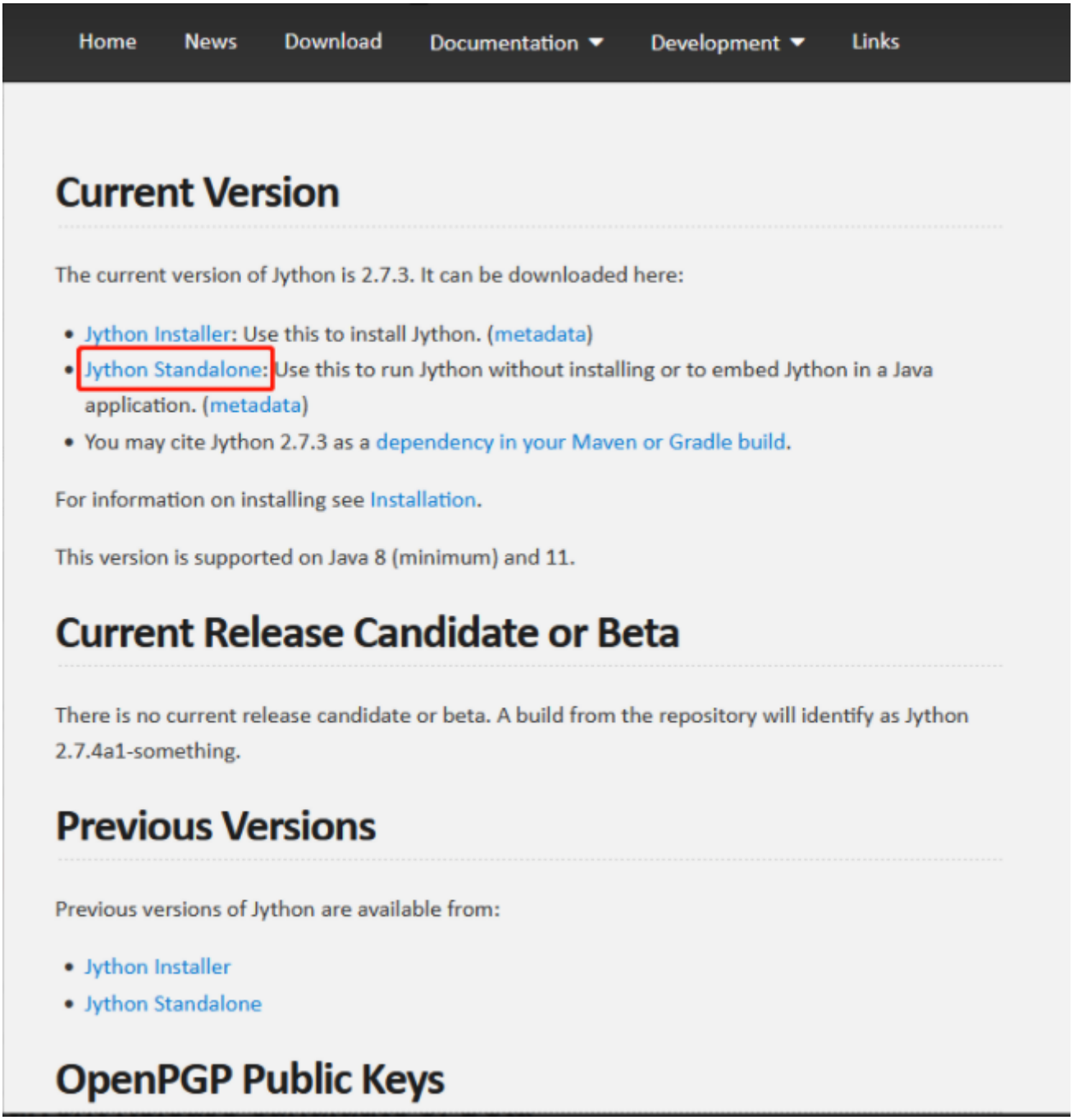
报告的执行状态如下：

- 绕过！-红色
- 强制执行！-绿色
- 强制执行？？？(请配置强制检测器) -黄色

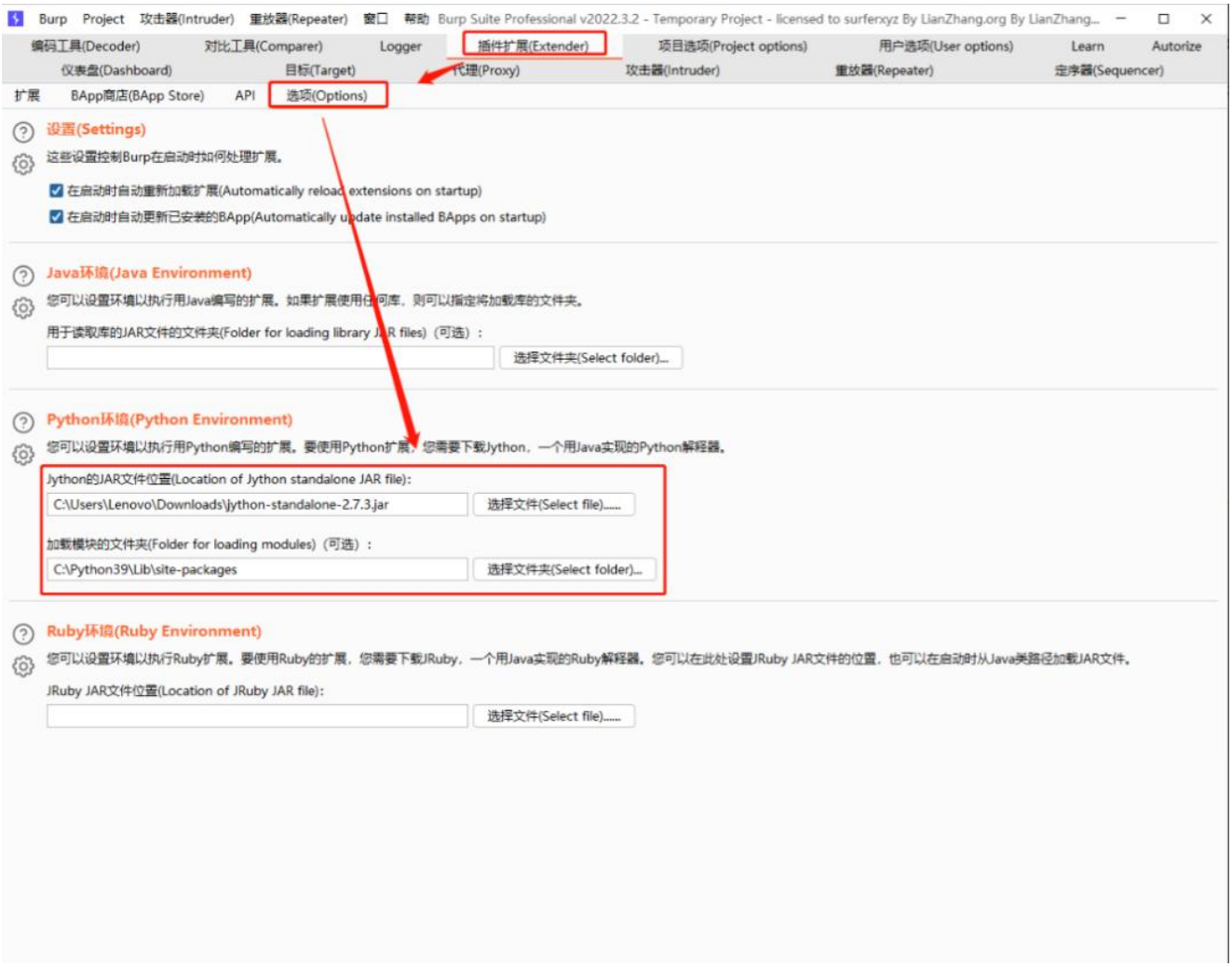
4.1、安装

1）首先我们需要下载Jython Standalone：https://www.jython.org/download

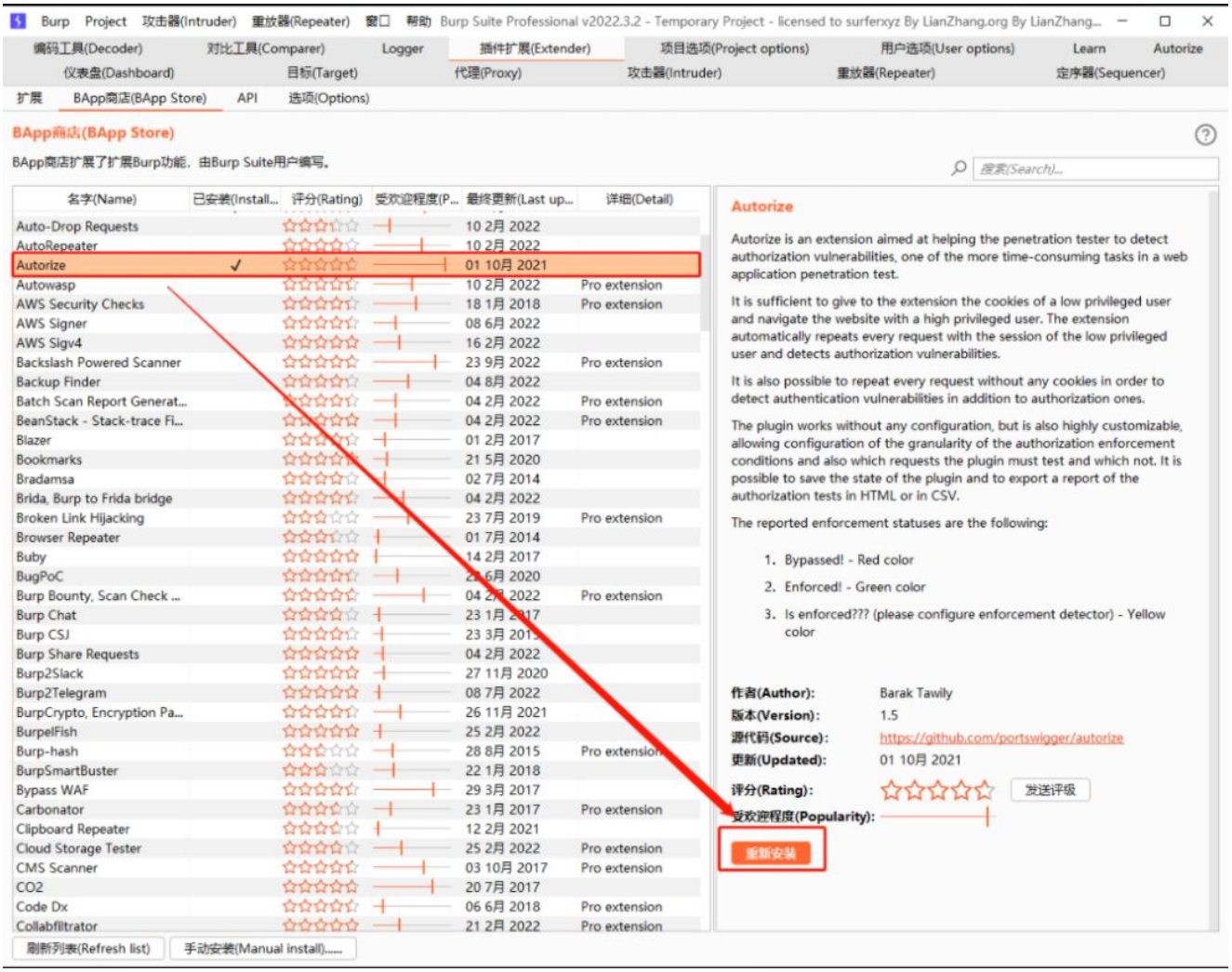
我们需要选择红框里面的进行下载：


Home
News
Download
Documentation
Development
Links
Current Version
The current version of Jython is 2.7.3. It can be downloaded here:
Jython Installer: Use this to install Jython. (metadata)
Jython Standalone: Use this to run Jython without installing or to embed Jython in a Java application. (metadata)
You may cite Jython 2.7.3 as a dependency in your Maven or Gradle build.
For information on installing see Installation.
This version is supported on Java 8 (minimum) and 11.
Current Release Candidate or Beta
There is no current release candidate or beta. A build from the repository will identify as Jython 2.7.4a1-something.
Previous Versions
Previous versions of Jython are available from:
Jython Installer
Jython Standalone
OpenPGP Public Keys

2）配置如下：

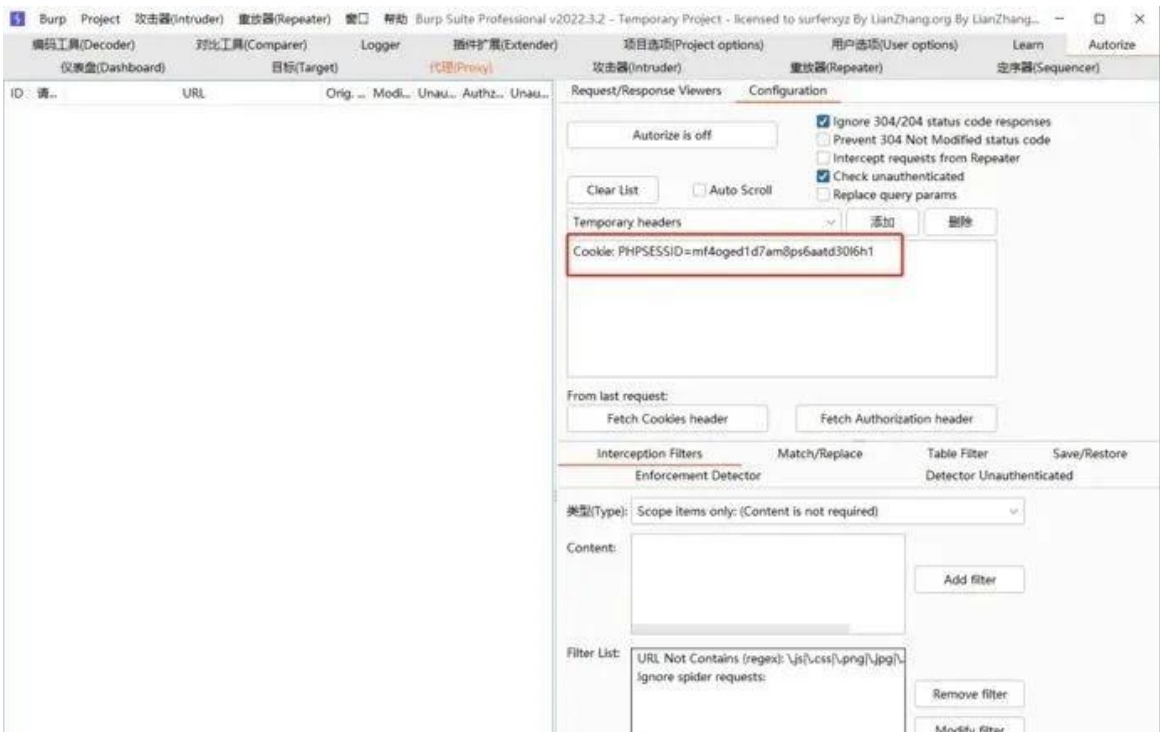


3）我们配置好之后，接着安装!

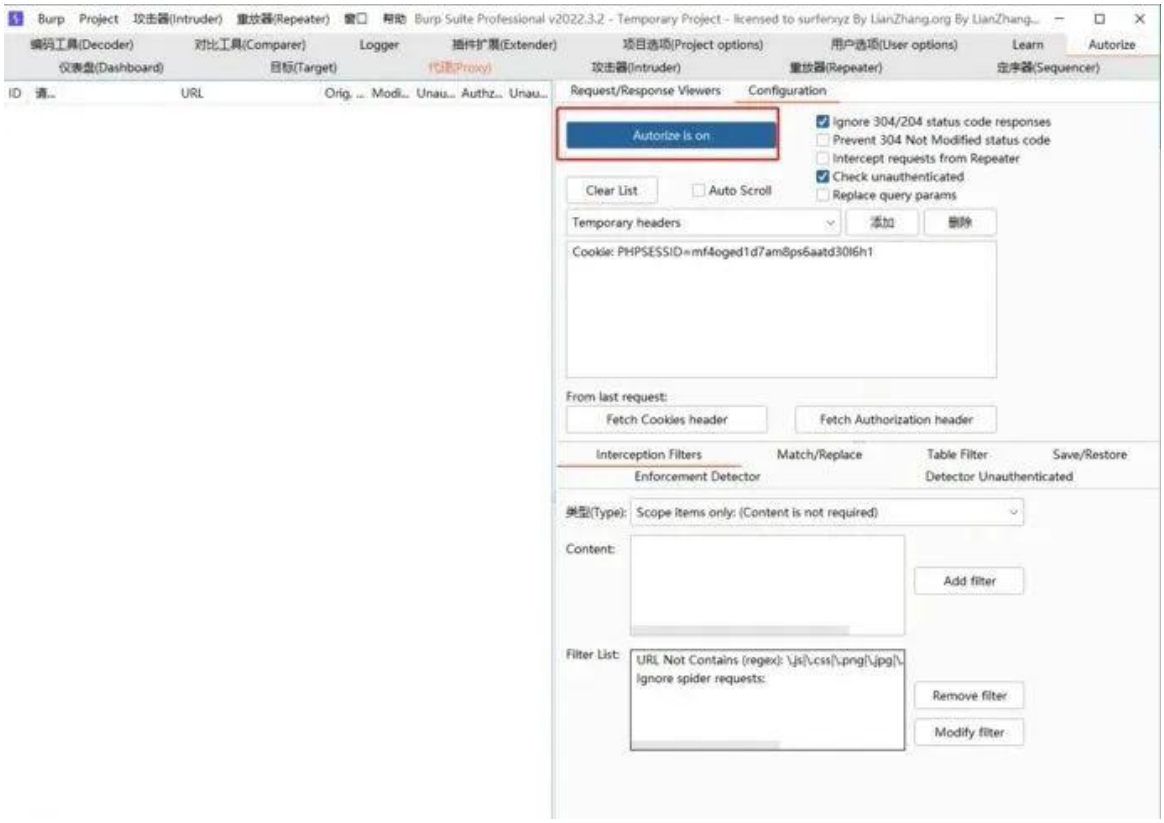


4.2、使用(pikachu为例)

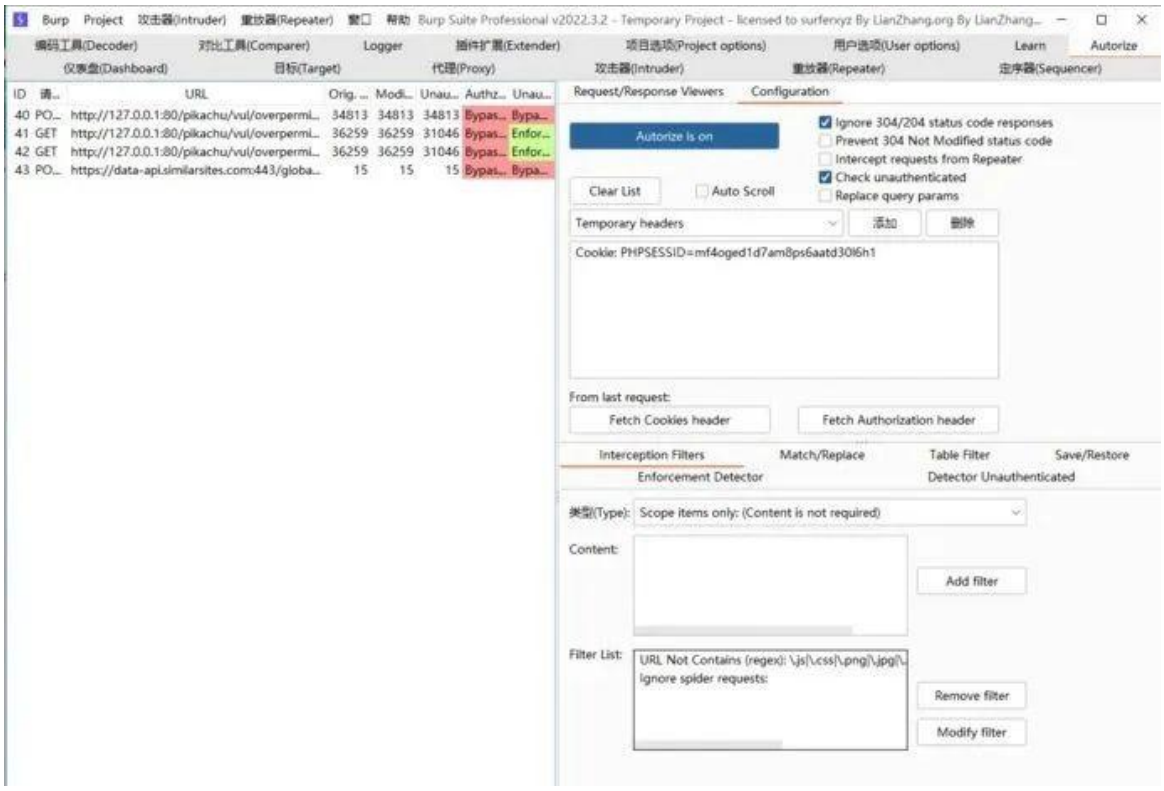
1）获取低权限的cookie，我们在垂直越权的目录，首先获取pikachu/000000的cookie；



2）接着我们去访问高权限的账号，这里需要注意，先打开插件的开关，然后再去访问高权限的账号；



3）接着就会出现结果；



左边一列 红色代表存在越权可能；

右边一列 红色代表存在未授权访问可能；

接着点击 三代表响应长度的数字，在右侧查看具体响应；

如果是 响应中 包含敏感数据，或者一些增删改的post请求，就可以报bug了；

五、支付漏洞

5.1、直接修改商品价格

在支付过程中，购买商品一般分为三个步骤：订购，确认信息，付款，那么这个修改价格具体时修改那一步时的价格？在我看来你可以在这三个步骤中随便一个步骤进行价格的修改，进行测试，如果前面两步有验证机制，那么你可在最后一步付款进行抓包尝试修改金额，如果没有在最后一步做好检验，那么问题就会存在，其修改的金额值，你可以尝试小数目或者负数，去测试；

参考案例：http://wooyun.2xss.cc/bug_detail.php?wybug_id=wooyun-2016-0226613

5.2、修改支付状态

1）这个很好理解，就是假设A商品，我们进行了购买，用bp进行抓包，我们看到某一个字段；

0：表示支付成功

1：表示支付失败

假设我们暂时还未支付，bp显示的是1，我们将1修改为0；

2）还有我们去购买A商品，是10元，B商品是1000元，我们先购买A商品，将A商品的数据包，给B商品，那么我们就能通过10元购买10000元的商品；

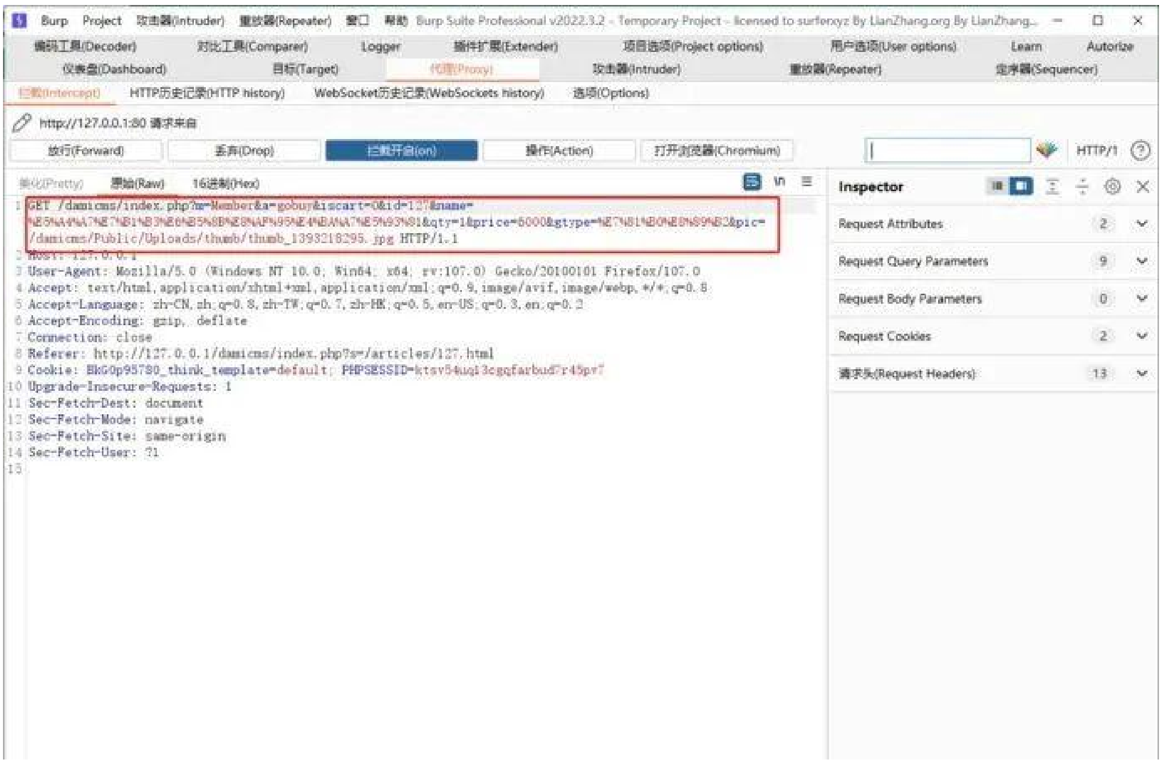
我们抓取购买包，放在重放器，这个商品为5400元；

我们将数据包放在重放器，重要的字段，我们做一下记录；

index.php?m=Member&a=gobuy&iscart=0&id=69&name=%E5%A4%A7%E7%B1%B3CMS%E6%89%8B%E6%9C%BA%E5%BC%80%E5%8F%91%E4%B8%93%E7%89%88&qty=1&price=5400&gtype=%E7%81%B0%E8%89%B2&pic=/damicms/Public/Uploads/thumb/thumb_1393206337.jpg

我们继续在购买6000元的手机；

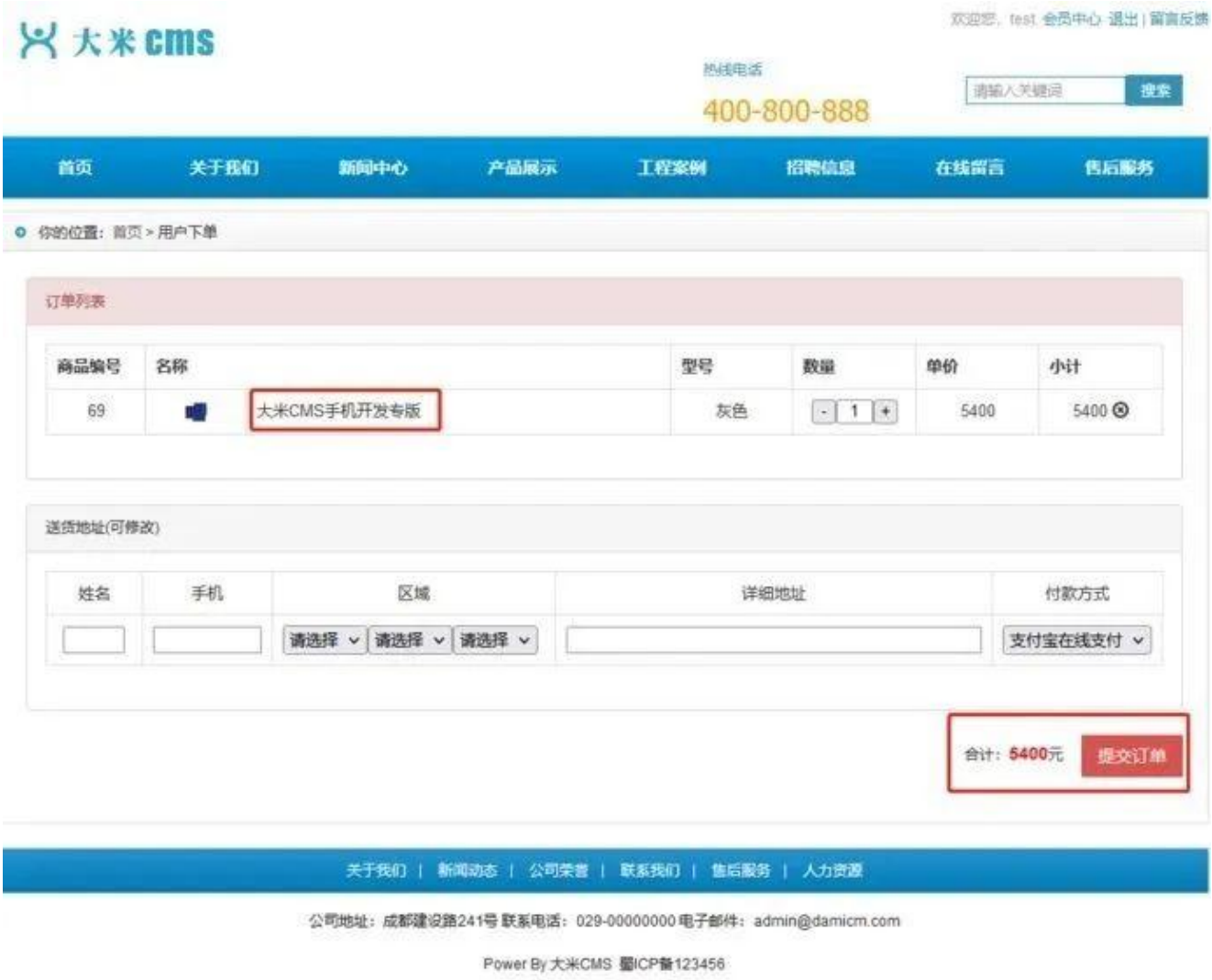
抓包，根据经验，我们将5400手机的数据包，放在6000元手机这里；



两个数据包进行对比：

```
6000元手机

index.php?m=Member&a=gobuy&iscart=0&id=127&name=%E5%A4%A7%E7%B1%B3%E6%B5%8B%E8%AF%95%E4%BA%A7%E5%93%81&qty=1&price=6000&gtyp
e=%E7%81%B0%E8%89%B2&pic=/damicms/Public/Uploads/thumb/thumb_1393218295.jpg


5400元手机

index.php?m=Member&a=gobuy&iscart=0&id=69&name=%E5%A4%A7%E7%B1%B3CMS%E6%89%8B%E6%9C%BA%E5%BC%80%E5%8F%91%E4%B8%93%E7%89%88&q
ty=1&price=5400&gtype=%E7%81%B0%E8%89%B2&pic=/damicms/Public/Uploads/thumb/thumb_1393206337.jpg
```

我们将5400元手机的数据包替换掉6000元手机的数据包；



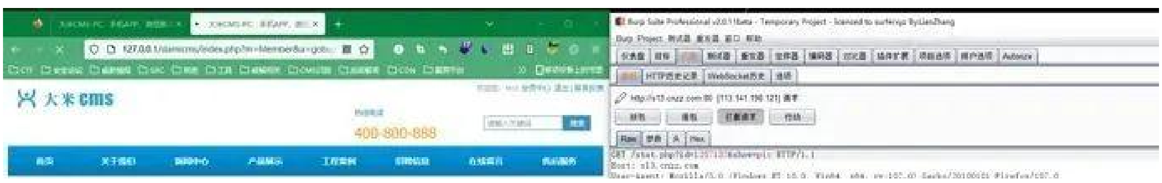
这里商品的价格就变成了5400；

5.3、修改商品的数量

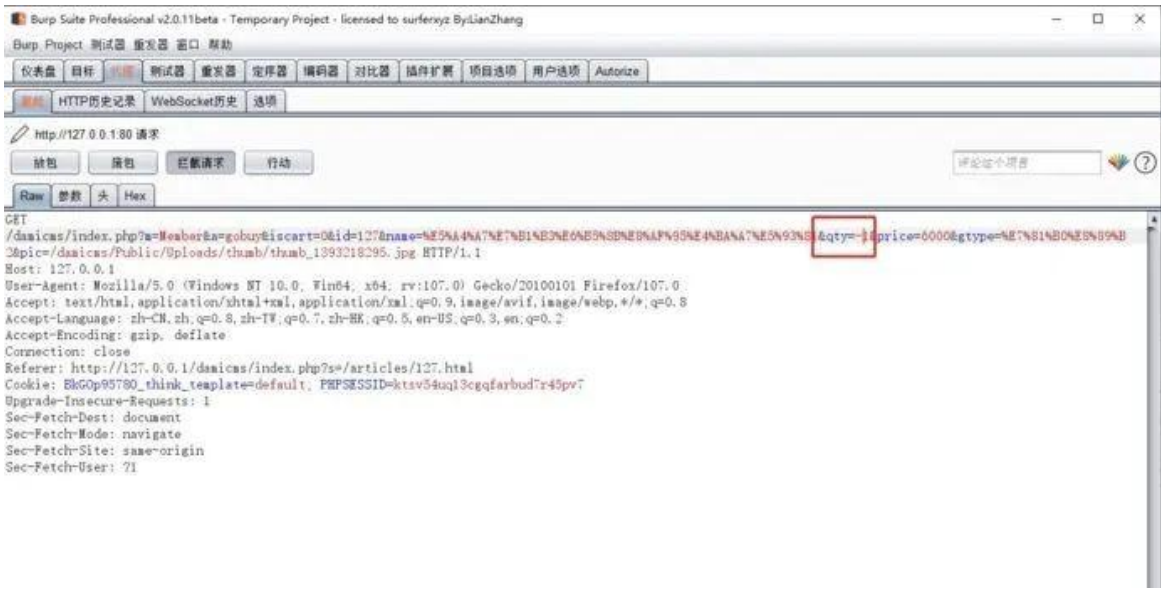
支付中。假如1个馒头10元钱，那么10个馒头就是100元，那么-1个馒头那？这种会不会产生零元购问题那？

我们可以看到一个手机为6000元，我们进行抓包；

这里qty这个字段非常可疑，我们将它修改为10，放包；



订单金额变成了60000万了，那么我们的猜想完全正确，我们将它的数量改为-1，看看什么情况；



这里的价格变成了-6000，说明漏洞确实存在；

5.4、另类支付

我们在支付的时候，常常会给你一些优惠卷，积分，满减等等，而这些值同样都是由操作的空间

1）修改优惠卷

一般优惠卷进行消费往往出现在第二个步骤当中，确认购买信息，这个页面当中，你可以选择相关的优惠卷，我们直接修改优惠卷的金额，等于商品的价格，或者直接将其改为负数，最后进行支付，如果说没有对这个点加以验证，那么直接可以支付成功；

2）修改优惠卷金额及业务逻辑问题

当你修改其优惠卷值为任意值或负数想要支付的时候，会显示支付失败，或者金额有误等一些提示，可能这个时候，你会选择放弃，但是当你点开个人中心的时候，点击订单

详情，如果存在这个逻辑问题，那么此时在你刚刚修改优惠卷金额后点击下一步支付的时候，其实已经产生了订单，在订单的金额内，你可以看到支付金额为0，然后点击支付就可以支付成功；

这里好友一个小技巧，有可能会支付失败，但是如果你找到的这个问题是一个业务分站点，如果有自带的钱包功能，那么你就可以利用这个自带的钱包功能去支付这个订单，而不要利用其他支付类型，就可以支付成功了；

3）修改积分金额

有些网站的积分，比如你消费了多少，就可以拥有一定量的积分，这个积分可以在你付款的时候进行折扣，如果这个积分的金额没有做好校验，那么你在支付当中将积分减去的金额变成商品本身的价格，或者负数，是不是就可以产生0原购物了；

4）满减修改

我们在双十一的时候，很多会有满300减100，这种功能，我们能不能将金额修改为满101减100那？

参考案例：http://wooyun.2xss.cc/bug_detail.php?wybug_id=wooyun-2016-0214319

5.5、修改支付接口

一些网站支持多种支付的接口，微信，支付宝，还有第三方的支付工具，然后每一个接口的值不一样，如果逻辑设计不当，那我随便选择一个点击时进行抓包，然后修改为一个不存在的支付接口，如果接口的相关处理没有做到位，是不是会支付成功；

5.6、重复支付

淘宝，京东会有很多试用卡。一张卡可以试用一个商品，我们可以将这个试用的商品数据包多次重复提交，如果服务端没有进行严格的校验的话，就会产生很多这样的订单，这时候，我们将这个商品退掉，会怎么样？我们会不会退回很多试用卡？

5.7、最小支付和最大支付

1）最小支付

很多测试人员在测试漏洞的时候，往往会将金额修改为0.01或者负数，但这这样会很容易错失掉一些潜在的漏洞，比如一些网站有金币或者积分什么的支付的时候可以用这些

来支付，10元等于100积分，50元相当于500积分，这个问题如果你在充值的时候将金额修改为0.01和负数会显示支付失败，但是如果你修改金额为1元，那么支付就会成功，也就是说1可以购买任意积分了？

其实你在测试的时候，就会发现，1元对应10积分，如果你修改0.01，那么对应的积分就是null，所以会显示失败，而当你修改为1元支付接口时存在的，其后面积分数为其他金额的积分，然后跳转过去支付就会以1元购买到比它多得多的积分，也可以时任意积分；

2）最大支付

一般在开发当中，商品的金额都会用int型来定义，最大值2147483647，我们尝试修改为2147483648，看看是否能造成整数的溢出，有可能支付状态异常，从而导致支付成功；

5.8、四舍五入导致支付漏洞

我们以充值为例，余额值一般保存到分为止，那么如果我充值了0.001元也就是1厘，一般开发会在前端判断我们的数字，或者将最后一位四舍五入，使用支付宝值直接报错，因为一般第三方只支持到分；

那我们如果充值0.019？由于支付宝只判断到分，所以导致只能支付0.01，而由于我们支付成功，前端会将9四舍五入，直接变成0.02，所以等于直接半价充值；

5.9、首单半价，无线重构

很多会员啊什么的，为了留住用户，都会有首月半价，或者免费等等活动，我们可以抓取这个数据包，进行多次支付，就可以一直优惠购买(百度云去年就有这个漏洞，可以6元一月超级会员)

5.10、越权支付

这个问题很早之前就有了，现在很少存在这类问题，在支付当中会出现当前用户的ID，比如ID=1，我们将ID改为2，如果没有加以验证，我们是不是用用户2的钱支付了用户1买商品的钱；

5.11、无线次试用

一些网站的一些商品，比如云系列产品支持试用，试用时期一般为7天或者30天，一个账户只能试用一次，试用期间不能试用，但如果这个试用接口没有做好分配，那么很容

易产生漏洞，比如支付的时候它的URL后面的支付接口为3，那么此时就会调用购买支付接口，但是由于你本身这个产品就是试用的，其相应值绑定了这个试用的产品，那么金额肯定是0，那么最后点击支付，你就可以看到支付成功，试用成功，又重复的试用了一次，然后他们的使用时间会累加到一起，这就导致了可无限制购买任何产品了；

5.12、多线程并发问题

多线程并发问题就是没有实时的处理各种状态所导致的问题，比如很多平台都有自家的钱包，而这个钱包是一个迷你钱包，这个钱包作用也仅是用于当前这个平台网站，再提现的时候，没有做任何的验证码或者校验机制，只要输入金额就可以体现，并且是秒到账，如果是什么负数，修改金额都测试了，不行的话，那你就可以实试一试多线程并发问题，体现的时候进行抓包，比如我现在钱包内有0.1元，那么按照每提0,01元可以提10次的话，也就是发送10次进程，但是利用这个问题可以达到多打几次成功的进程，体现时进行抓包，然后把数据包发送到BurpSuite工具的Intruder当中，进行批量发送12次，然后可以看到成功体现12次，也就是0.12元，从这里就可以看出这个问题的危害了，当然此时账户的金额肯定是为负的了，如果把这个提现金额变大，那么这多提现的金额可不是闹着玩的。

当然，多线程也可以在其它功能处进行测试，比如我之前讲到的试用商品问题，就可以通过多线程进行多几次的使用，比如利用积分总换礼品，一个账户只能进行总换一次，利用这个问题，可以多几次总换，一些转账功能，提现功能，购买功能等等很多。

六、Cookie脆弱性

1）打开环境

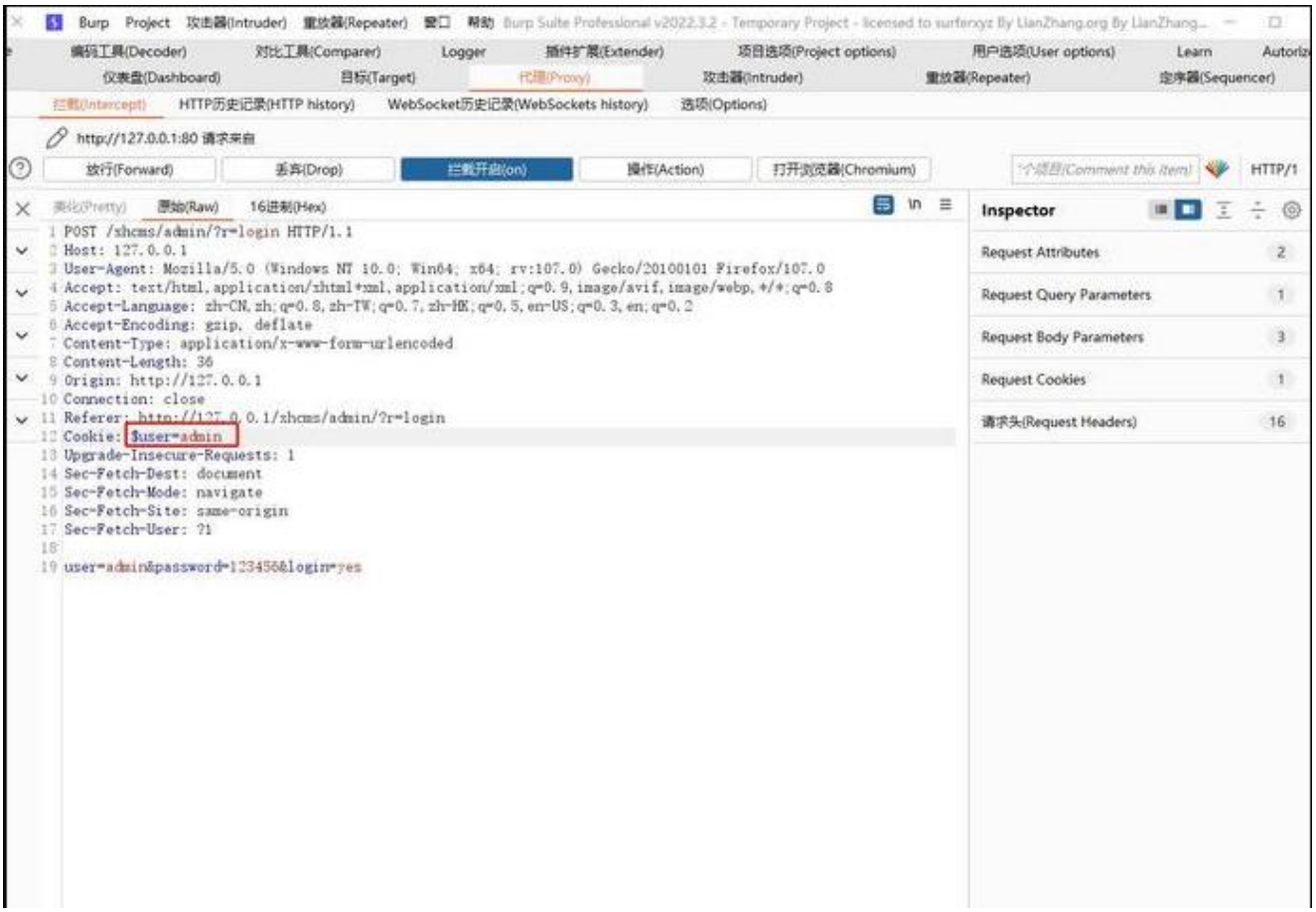
2）代码审计，我们打开login.php;

checklogin.php - 记事本

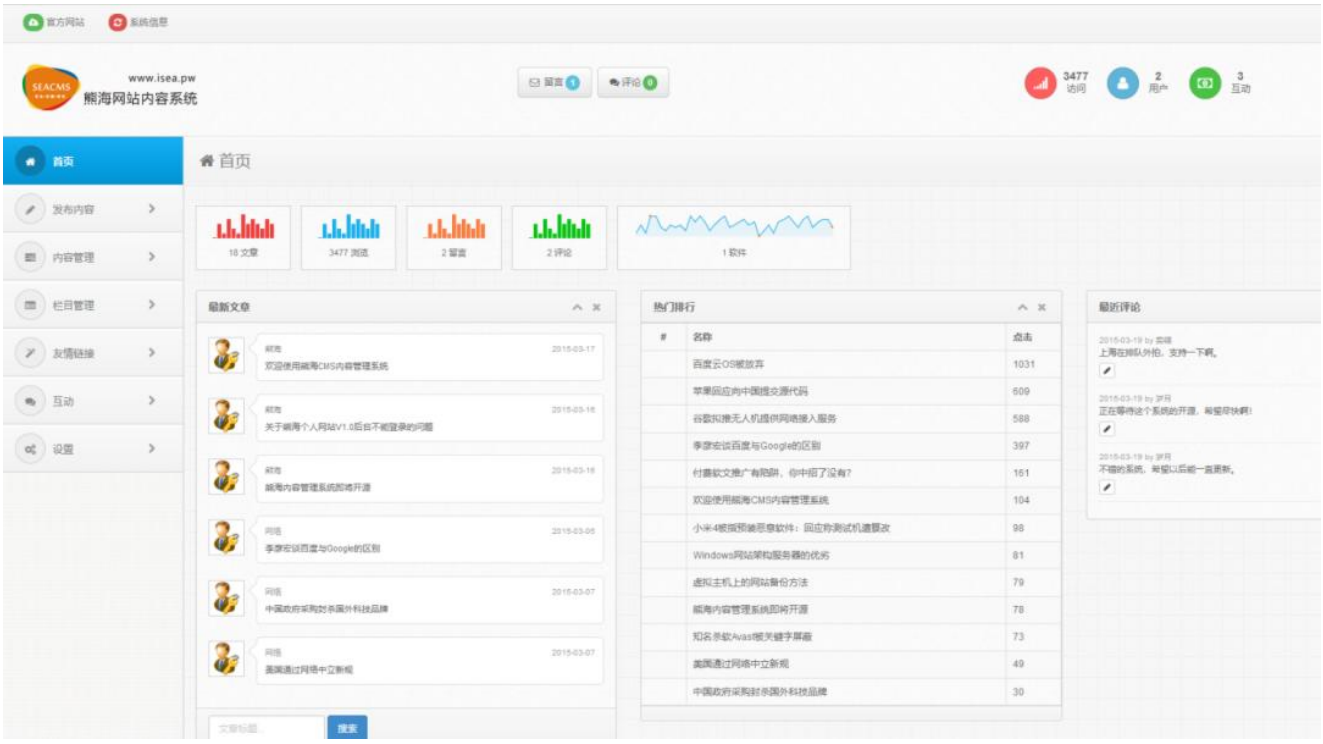
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<?php
$user=$_COOKIE['user'];
if ($user==""){
header("Location: ?r=login");
exit;
}
?>
```

明显可以看出来，Cookie传入了user，如果user为空，退出，就是登陆不成功，我们可以抓包，让他不为空；



我们让$user=admin，放包；



这个漏洞可以说非常鸡肋，实战没有代码，怎么知道cookie的脆弱性那？那我们就要分析，比如抓包看到了，cookie=admin，那我们可以改一改，让cookie=test，是不是可以跳转到test用户了；

七、客户端回显

1）介绍

这种利用回显抓取数据包，在修改验证，我们尝试的时候，有的网站会在你填写手机号后，点击发送验证码，会对你的手机号和IP进行验证，看看是不是本地号码，有的会对传输的信息有加密；

2）原理

在调用短信平台发送信息时，没有判断验证码和手机号是否绑定，把验证码校验功能直接放到客户端进行(也就是返回数据包中)，从而导致验证码在客户端回显；

3）危害

这个危害有一点明显，登陆，注册，绑定，重置任意用户的密码；

4）利用

客户端回显，就是当注册或者绑定的时候，用户向网站系统发送一条短信验证码的请求，cookie中会包含短信验证码并且回显在数据包中，如cookie=xxxxxx

es_id=534324,我们这时候通过抓包工具可截取真实的验证码，如cookie=xxxxxx es_id=567475，通过分析，真实得验证码为567475，我们这时候，将验证码修改为正确的验证码，提交上去，就会成功注册或者绑定；

八、Response状态值

1）原理

Response状态值，就是在服务器发送某个密码重置的凭据之后，出现特定的响应值(ture，1，ok，success等等，例如响应头中的HTTP/1.1 200 ok)

2）利用过程

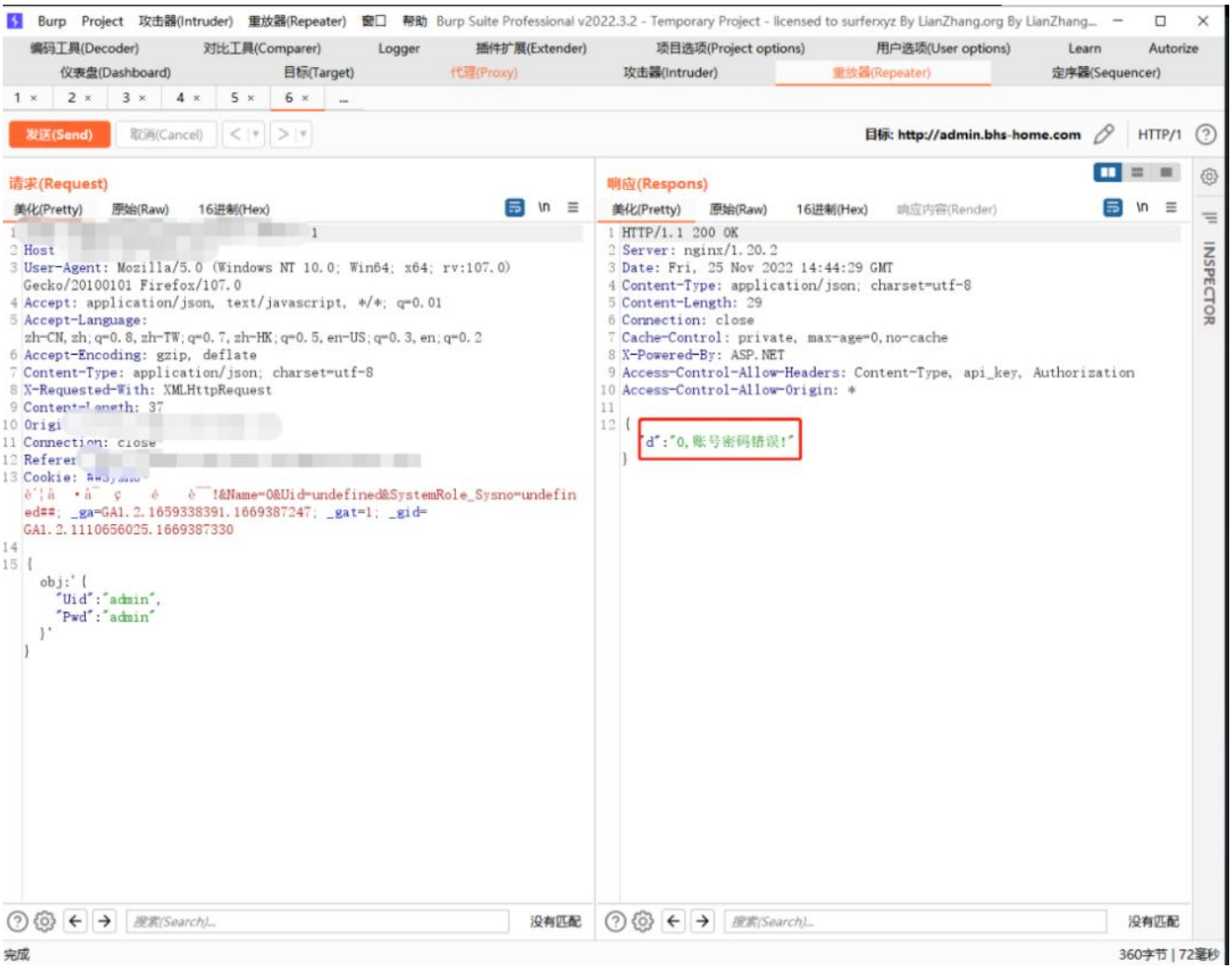
对Response状态值修改后，如果存在校验不严(存在逻辑漏洞)，并且回显值得校验是在客户端进行，就能使相关操作成功被执行；

就像密码重置中的验证码问题，如果回显值得校验是发送到客户端进行，通过对校验值得使用规则进行分析后，抓包将Response状态值改为正确得，然后放包，从而达到重置密码得效果；

3）案例演示

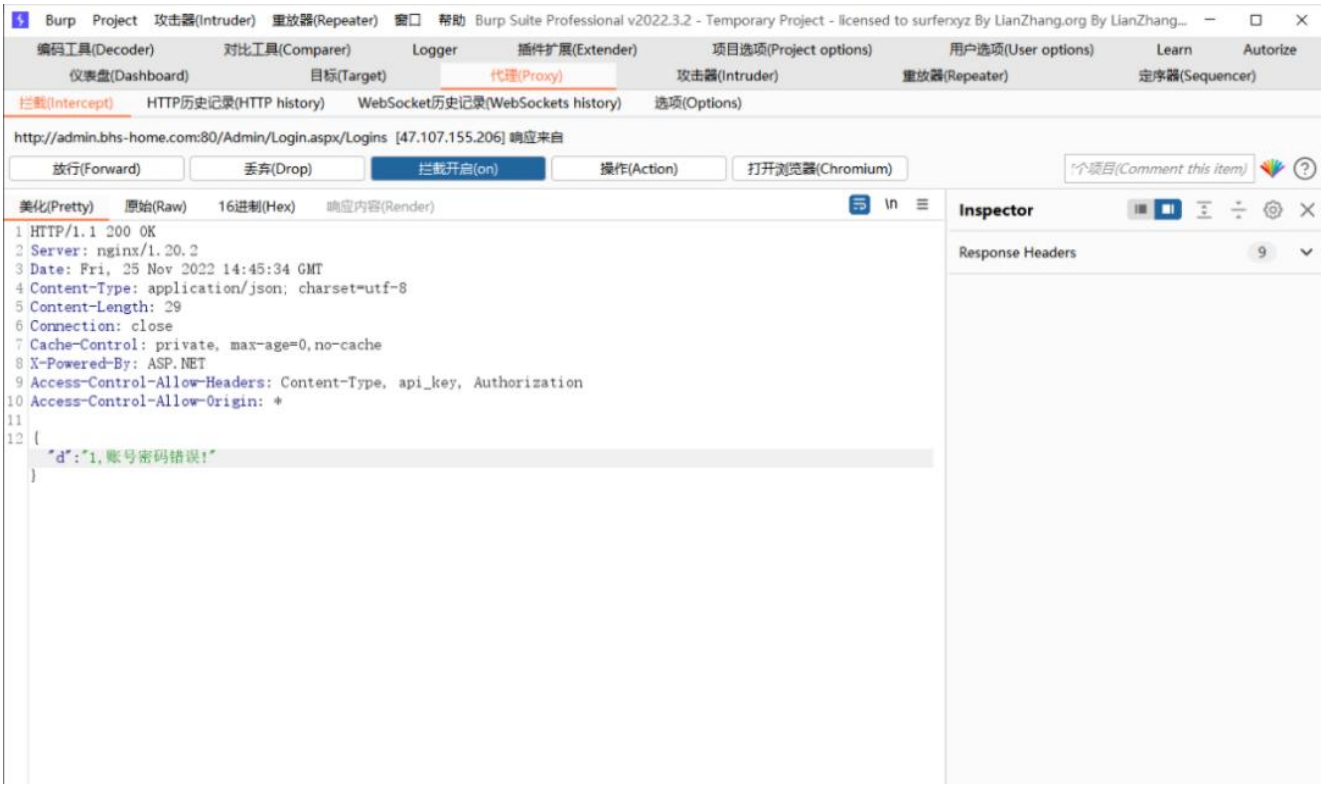
演示地址：[http://xxx.com/Admin/Login.aspx]

我们账号密码输入admin/admin，进行抓包，我们将包首先放到Repeater中看看结构；

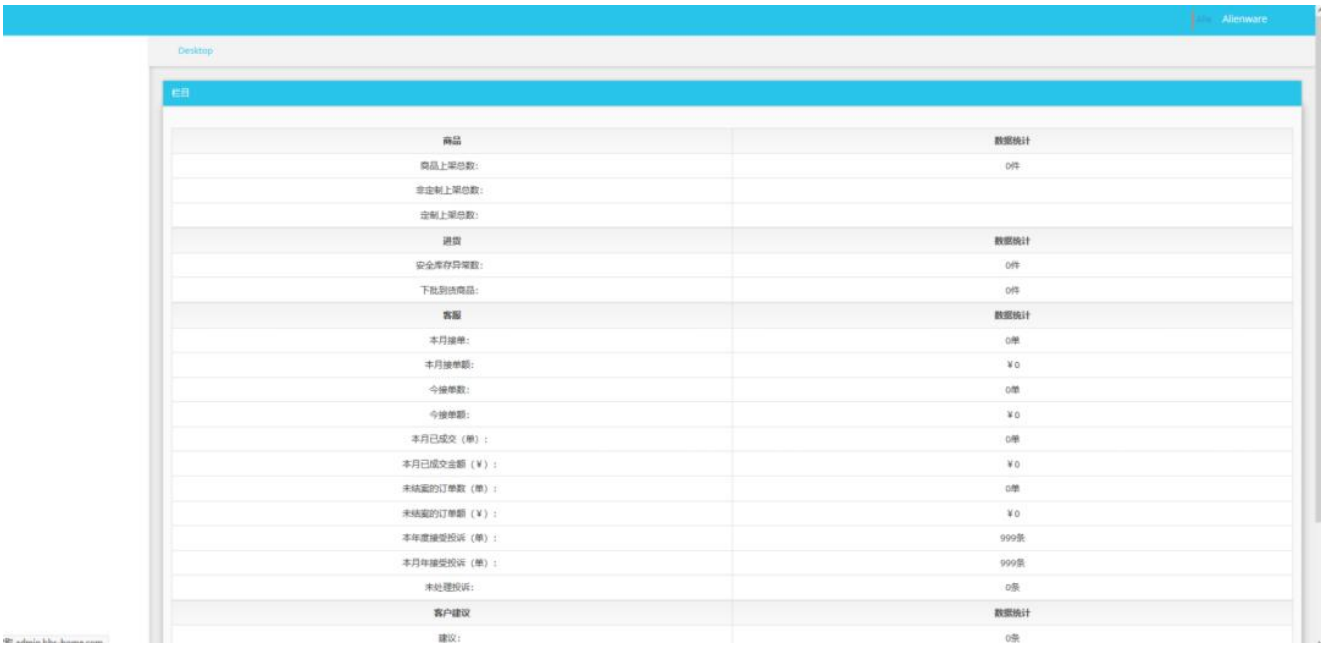


我们能不能猜测"d":"0,账号密码错误!"，我们将0改为1，直接登陆那？

我们拦截响应包，将0改为1；



成功进来了；



九、验证码

9.1、无效验证

在验证码模块，但验证模块与业务功能没有关联性，此为无效验证，一般在新上线得系统中比较常见；

我们在获取短信验证码后，随意输入验证码，直接输入两次密码，可成功更改用户得密

码，没有对验证码进行验证；

9.2、任意用户注册

我们首先利用自己得手机号接收验证码进行验证，下一步跳转一个设定密码得页面，接下来抓包，篡改手机号码，使用任意手机号进行注册，问题解读：这里业务一致性存在安全隐患，身份验证与密码修改得过程是分开的，验证无效；

9.3、客户端验证绕过

客户端验证是不安全的，和我上面的逻辑漏洞一样，可能会导致任意账号的注册，登陆以及重置任意用户等一系列的问题，有的时候会直接在前端显示验证码的明文信息；

我们拿到前端的验证码，即可成功重置密码；

9.4、返回密文验证码

我们用抓包工具，抓取到包之后，会返回一大串密文，这个时候，我们把密文拿去解密，即可获得验证码；

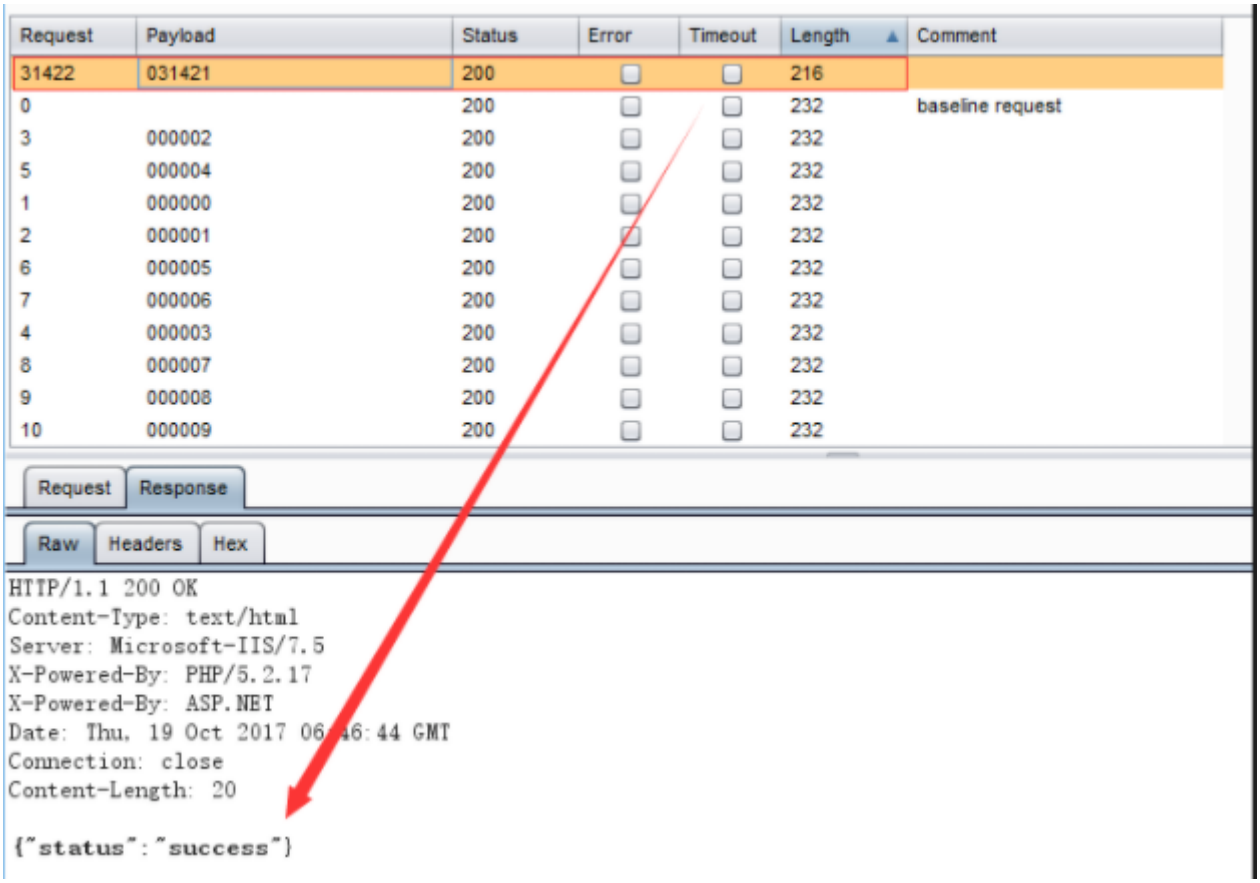
9.5、拦截替换返回包

我们第一步使用正常的账号修改密码，获取验证码通过时服务器的返回包，保存该信息，我们第二次，用另一个账号进行登陆，这时候，我们不知道验证码，我们随便输入验证码，在抓到验证码错误的返回包，我们将刚才正常的返回包替换掉错误的返回包，这时候会提示密码修改成功；

9.6、验证码爆破

短信验证码一般由4位后者6位数字组成(还有特殊情况，字母数字组合等等)，若服务器未对验证时间，次数进行限制，则存在爆破的可能；

输入手机号获取验证码，输入任意验证码，抓包放到Intruder模块，将短信验证码字段设置伪payload，然后取值范围设定好，进行暴力破解，根据返回响应包的长度判断是否爆破成功；



9.7、验证码与手机未绑定

一般来说短信验证码仅能使用一次，验证码和手机号未绑定，验证码一段时间内有效，那么就可能出现如下情况：

- A手机验证码，B可能拿来用
- A手机在一定时间间隔内接收到两个验证码，都可以用；
- 检测接收验证码的手机号和绑定的手机号是否一致

案例：任意用户密码重置

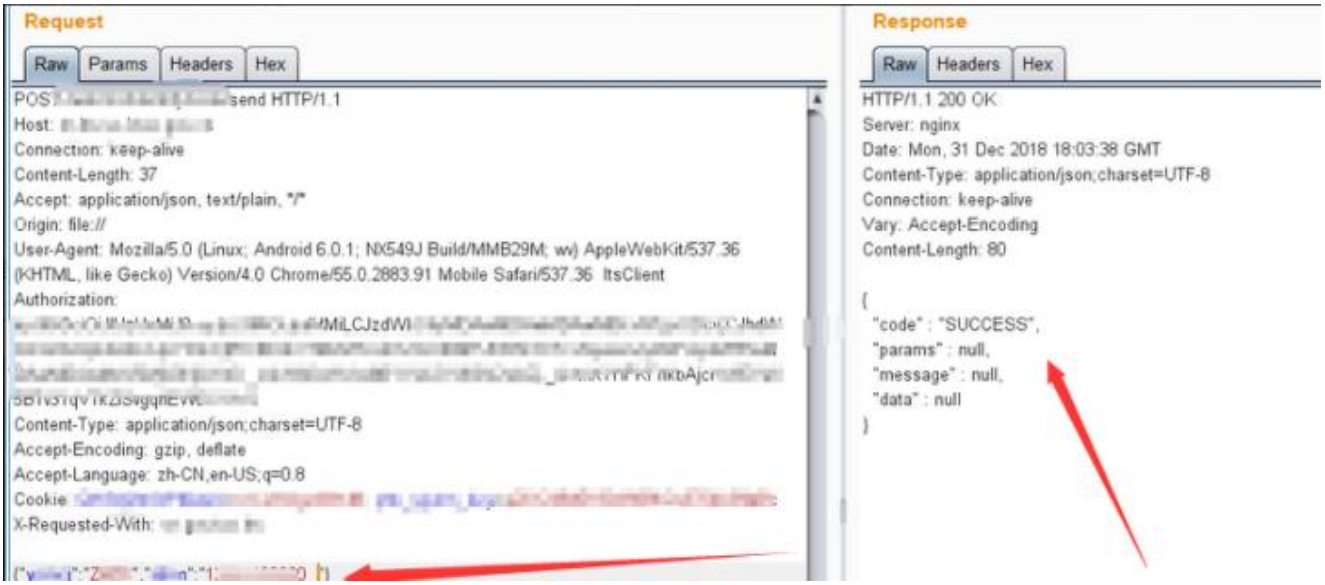
- 使用自己的手机号收到验证码
- 自己的验证码和对方手机号填上，下一步就是设置新的密码

9.8、代码层逻辑缺陷

如果代码层的逻辑是这个样子：

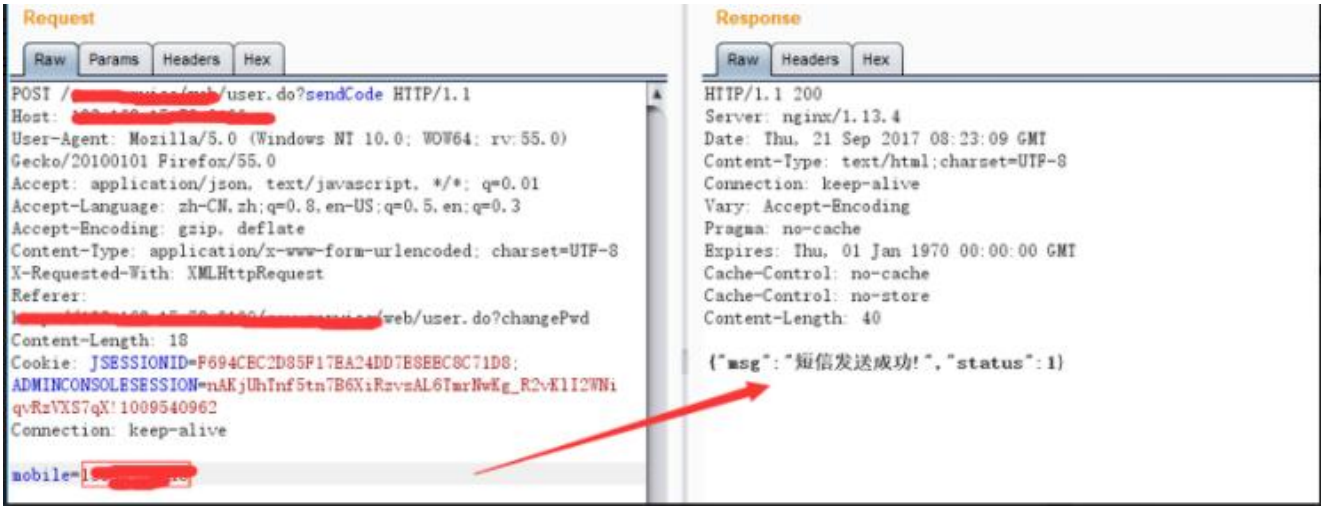
- 验证手机号是否已发送验证码
- 去除用户输入的空格和其他特殊字符
- 重新发送验证码

那么，可利用"\n"和空格进行绕过，持续递增空格就可造成无线短信轰炸，我们在手机号后面加一位进行fuzz，可能会有不一样的结果；



十、短信轰炸

1）短信轰炸时手机验证码中最常见的一种漏洞类型，在测试过程中，对短信验证码接口进行重发数据包，导致大量发送恶意短信；



2）大多数情况下，短信验证码都是间隔60秒，这种情况，我们就每隔60秒发一条短信，无线下发，短信轰炸，在测试过程中，可通过编写Python脚本来实现短信轰炸；

```
#coding=utf-8
import json
import requests
import time
start_time = time.time()
count =input("Please input counts:")
phone =raw_input("Please inut your phone:")
i=0
while (i<count):
```

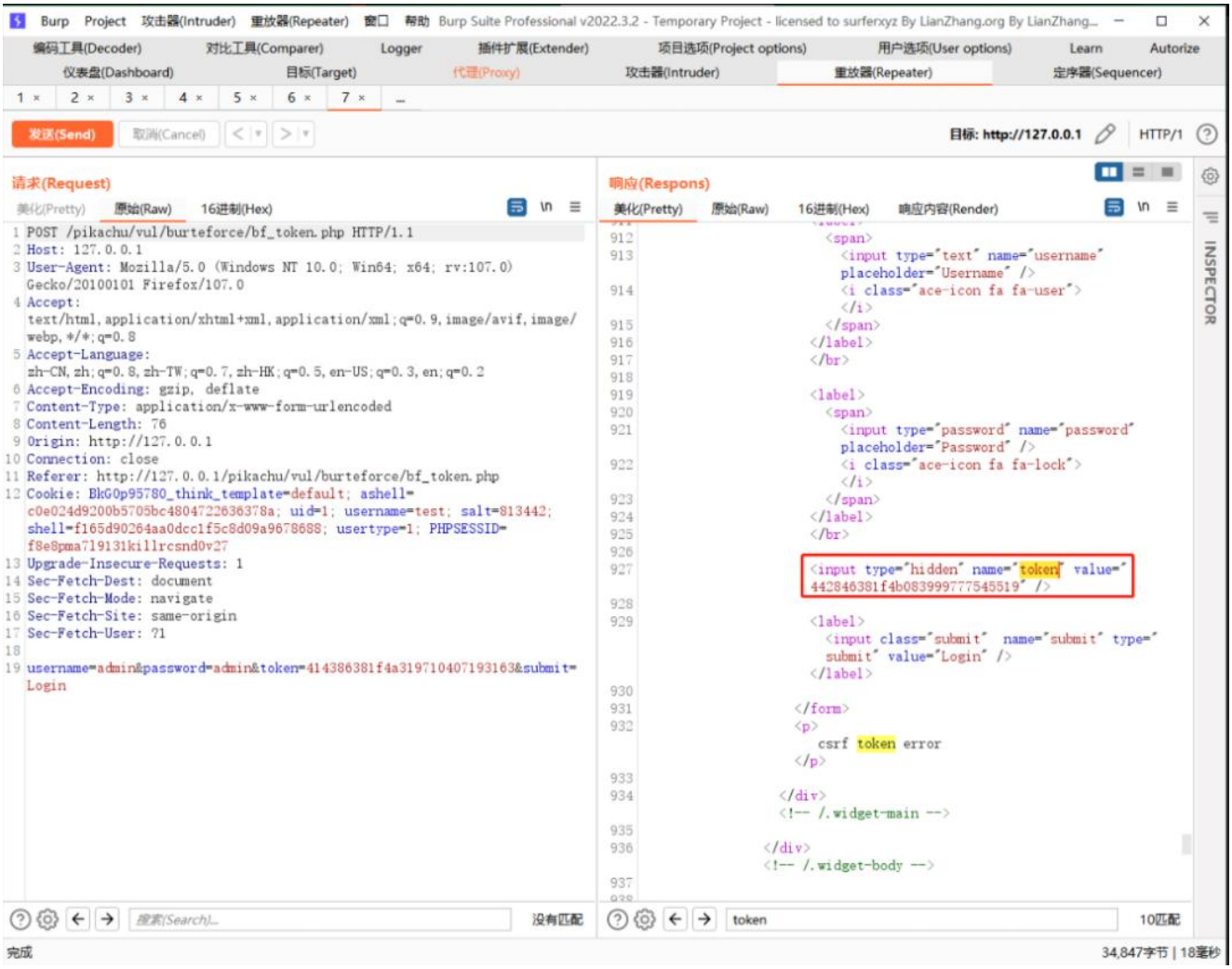
```
    url= "http://xxxx.cn:9092/map/GenerationUpdate"
    data=json.dumps({"headerInfo": { "functionCode":
"randomcode4G"},"requestContent":{"phoneNumber":phone}})

    header = { 'User-Agent' : 'Mozilla/5.0 (iPhone; CPU iPhone OS 10_2_1 like Mac OS
X) AppleWebKit/602.4.6 (KHTML, like Gecko) Version/10.0 Mobile/14D27
Safari/602.1''Host': 'xxxx.com:9092',
            }
    r = requests.post(url, data=data,headers=header,timeout=5)
    result=r.content
    if result.count('serviceCode":0'):
        print 'Sending message : %d seconds ' % (time.time()-start_time)
    i=i+1
```

十一、绕过token

11.1、前端回显

我们启动pikachu靶场；

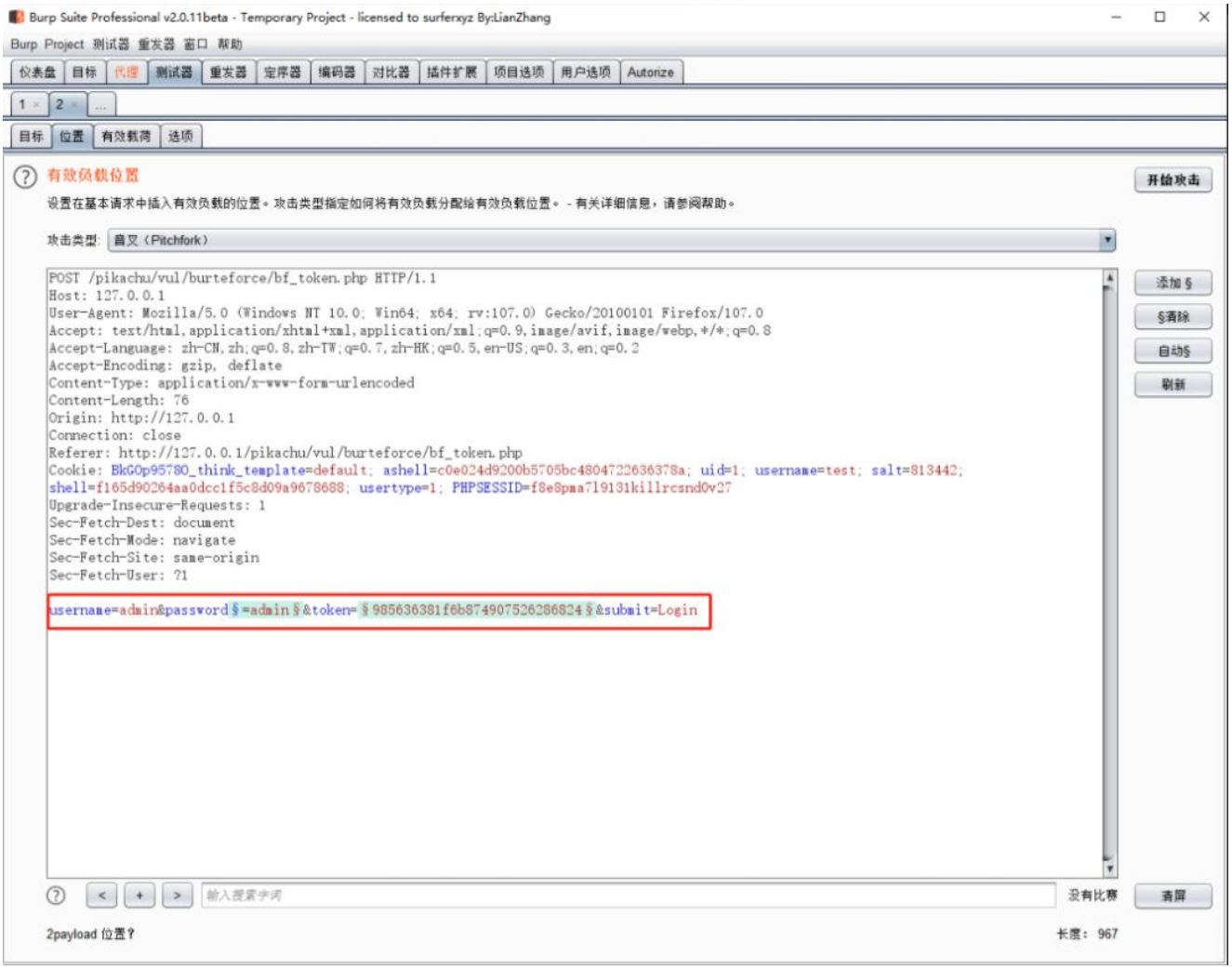
抓包放在重发器，可以看到token直接回显在响应里面；



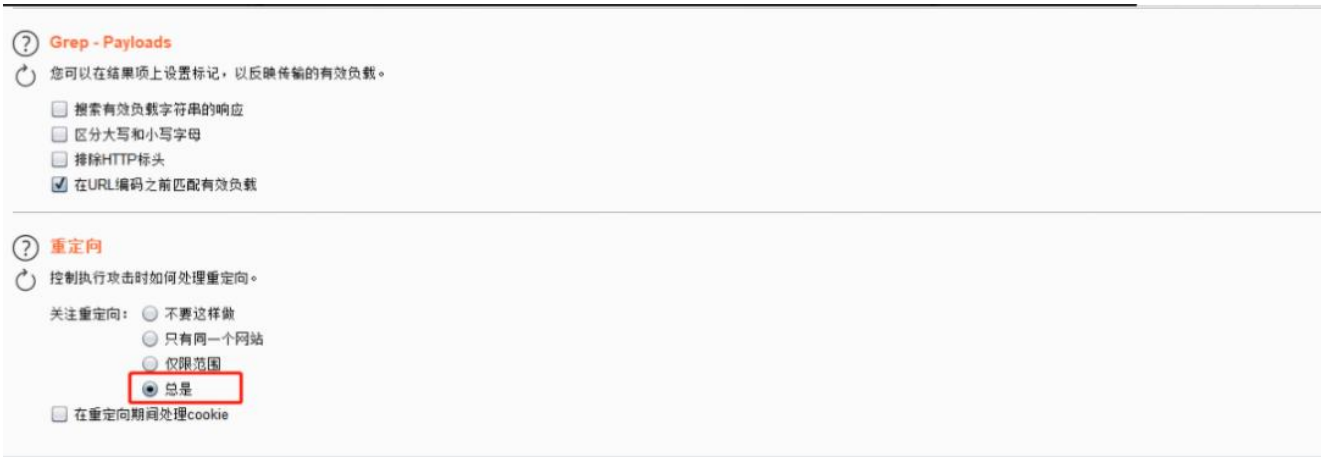
这里就非常不安全，我们可以借助回显，对token进行爆破；

11.2、token爆破

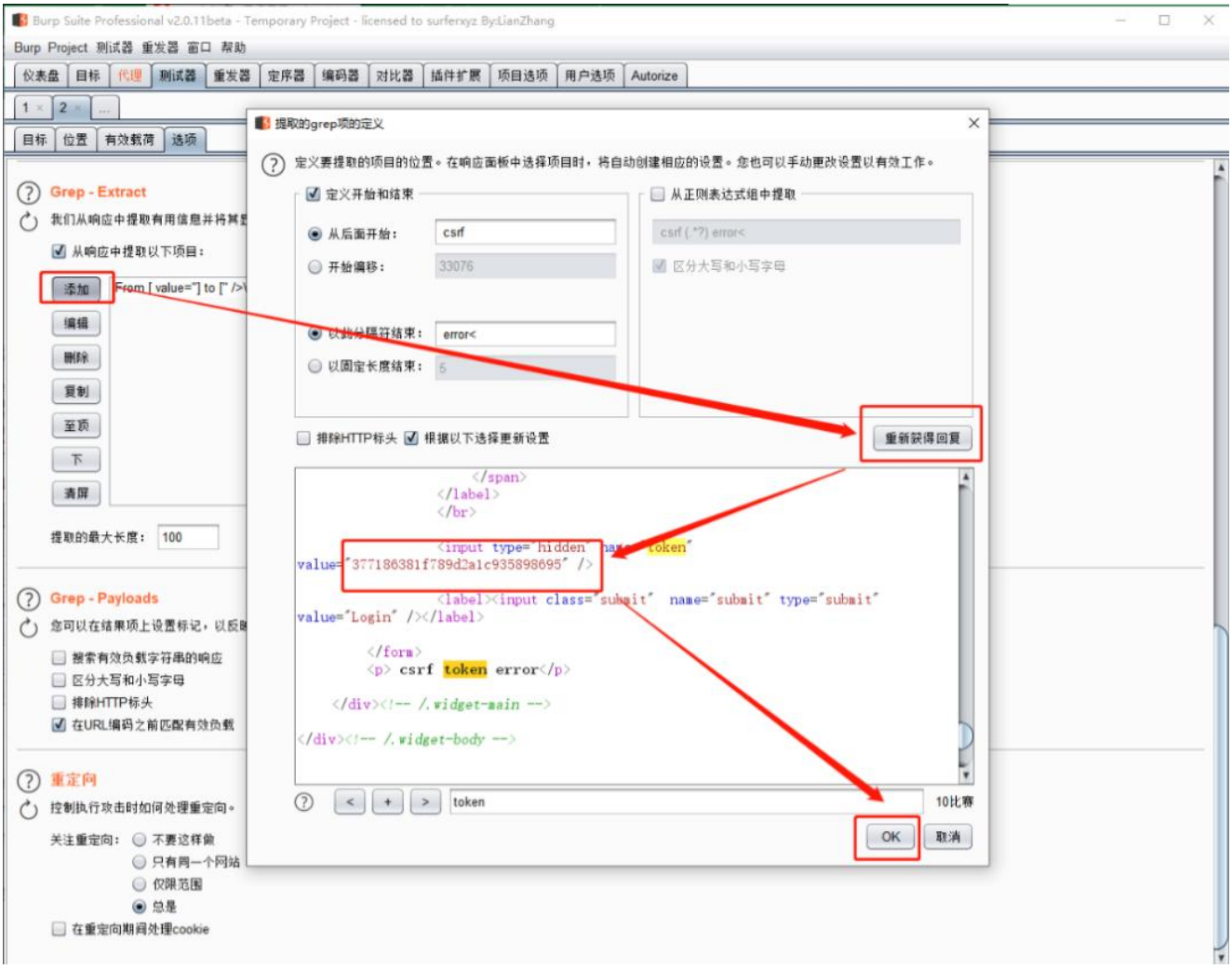
我们将抓包的内容发送到Intruder模块；



在Options中找到Request Engine模块，把线程设置为1，这里如果说线程设置过高，是没有效果的；

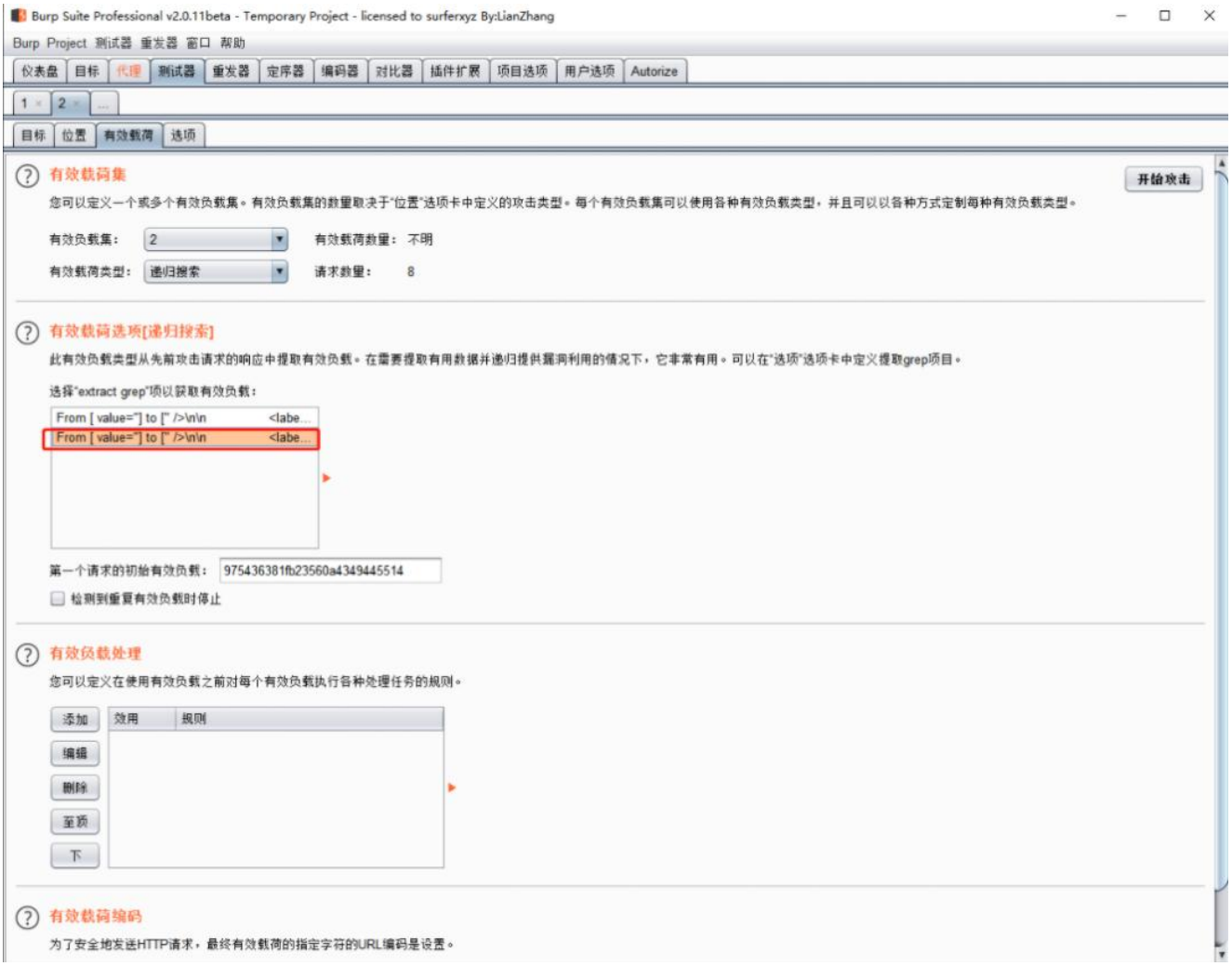


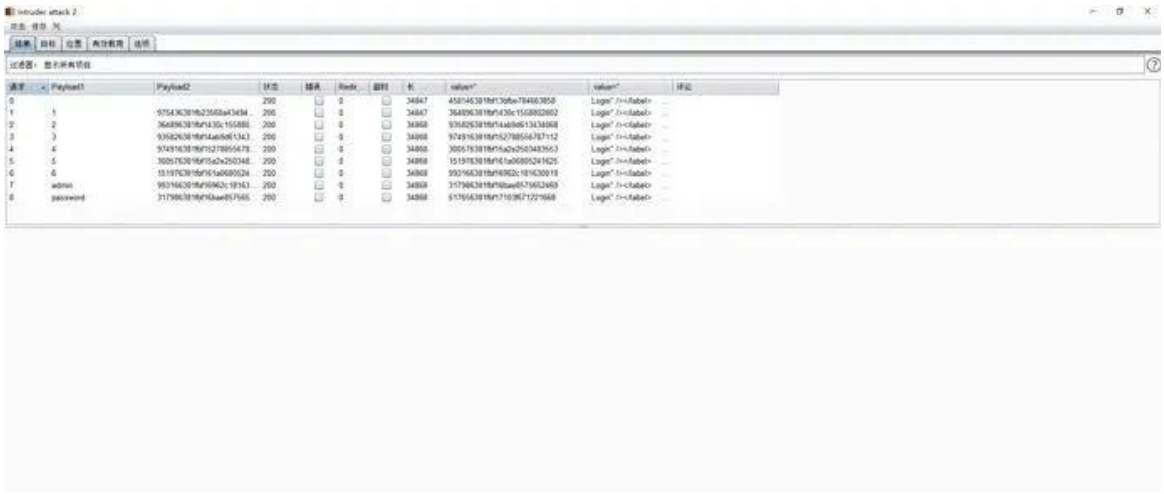
在Options在grep位置添加找到token并添加token；



payload1的位置随便输入

payload2的位置训责递归搜索；





十二、Callback自定义返回调用

12.1、什么是Callback

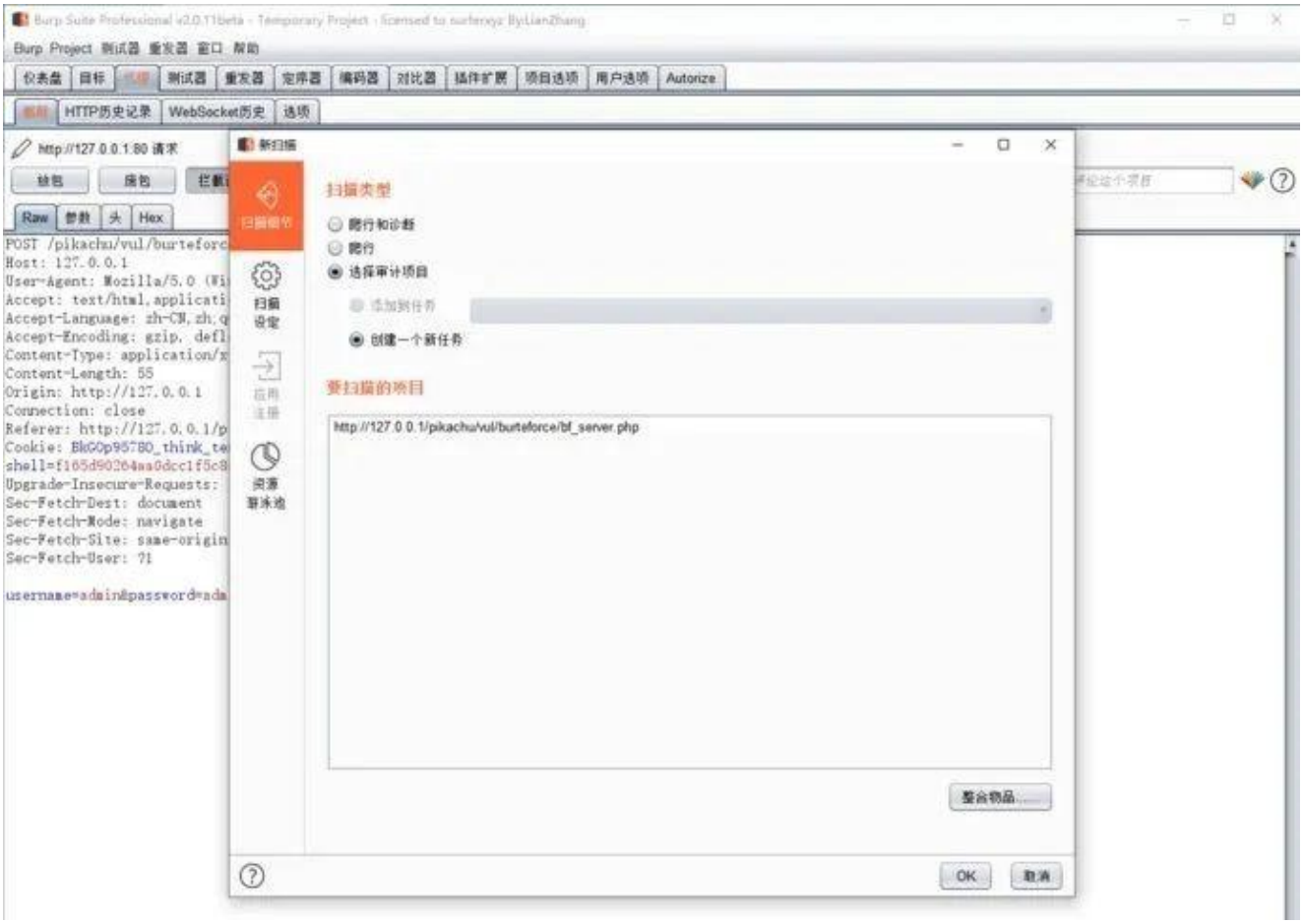
一般来说，函数的形参是指由外往内函数体传递变量的入口，但此处加了callback后则完全相反，它是指函数体在完成某种使命后调用外部函数的出口，也就是回头调用外部函数的意思；

链接如下：https://xxx/login?callback=https%3A%2F

%2F2haohr.com%2Fpersonnel

这里的callback后的数据代表微信登陆，然后将微信登陆的数据返回给callback，callback参数可以更改，可以和跨站漏洞结合，在网页中源码中搜索传递的参数，如果存在，意味着URL传递的参数会在网页的前端显示，是不是意味着我们可能可以构造XSS漏洞；

我们在挖漏洞的时候，着重关注关键的参数：id，callback，filename，uid等等



我们可以在这个页面搜索关键字来进行过滤；选出敏感的信息；

```
1 作者：wangkun05
2 转载自：https://www.freebuf.com/articles/web/350865.html
```

声明：文中所涉及的技术、思路和工具仅供以安全为目的的学习交流使用，任何人不得将其用于非法用途以及盈利等目的，否则后果自行承担。所有渗透都需获取授权！

@ 学习更多渗透技能！体验靶场实战练习

(hack视频资料及工具)

(部分展示)

往期推荐

【精选】SRC快速入门+上分小秘籍+实战指南

爬取免费代理，拥有自己的代理池

漏洞挖掘 | 密码找回中的套路

渗透测试岗位面试题(重点：渗透思路)

漏洞挖掘 | 通用型漏洞挖掘思路技巧

干货 | 列了几种均能过安全狗的方法！

一名大学生的黑客成长史到入狱的自述

攻防演练 | 红队手段之将蓝队逼到关站!

巧用FOFA挖到你的第一个漏洞

看到这里了，点个“赞”、“再看”吧

喜欢此内容的人还喜欢

实战 | 我是如何在5分钟内获得上千美金的漏洞赏金

HACK学习呀

【已复现】pyLoad远程代码执行漏洞(CVE-2023-0297)安全风险通告

奇安信 CERT

批量漏洞挖掘思路小结

鹏组安全