

OWASP Hackademic挑战项目-挑战1

我的第一个解决方案将来自OWASP Hackademic Challenges Project - Challenge 1。

下面是方案：

我们的代理商（黑客）告诉我们，有合理的怀疑，认为该物流公司的所在地对人体走私组织是一个盲点。

该组织通过招聘高薪工作的广告吸引受害者。他们选择那些没有很多亲戚的人，将他们暗杀，然后以很高的价格将其器官卖给非常有钱的客户。

这些员工在公司的秘密文件中被注册为“特殊客户”！

我们的代理商之一已被该公司聘请。不幸的是，自2007年1月1日起，他失踪了。

我们知道我们的经纪人还活着，但我们无法与他联系。他上次与我们联系时提到，如果出现问题，我们可以从公司提供给他的电子邮件地址与他联系。

问题是，当我们上一次与他交谈时，他还从来没有公司的电子邮件地址，但是他告诉我们可以通过公司的网站找到他的电子邮件。

我们唯一记得的是他在13日星期五被录用！

您必须找到他的电子邮件地址，然后使用公司网站的中央通信面板将其发送给我们。

祝好运！！！

解：

进入物流公司向我展示以下内容：

如您所见，我们需要找到代码和密码。查看页面确实并没有给我们任何帮助。

接下来，我看一下页面源代码。也许这里有一些好吃的东西！

```
<p><i><span class="style3"><span lang="el">anonymous Corporation since 1990. Reg. No: K7827-232-210B/1990</span></b><font color="#808000"> white, rabbit</font></b></i></p>
<form action=".main/index.php" method="post">
<p align="center">Enter Code / Password
<input type="text" name="name1" size="20">
<input type="text" name="name2" size="20">
```

好看看我们这里有什么。看第一行的结尾，我们看到字体颜色是#FFFFFF =白色。在此白色字体内，它具有白色和兔子色。让我们看看这是代码和密码。

输入我们的信息后，我们登录到Greek Logistics Company的门户。

从左侧看，我们看到“发送电子邮件”选项。我们稍后将使用它，因为我们需要发送电子邮件以找到我们捕获的朋友。

让我们看一下邮箱特殊客户的邮箱。

输入我们的信息后，我们会看到一个电子邮件地址列表。重新阅读场景，我们知道我们的朋友在13日星期五去了MIA。查看电子邮件列表，其中之一跳出来：

Jasson Killer Friday13@JasonLives.com。让我们使用此电子邮件地址，看看是否可以与被绑架的朋友联系。

回到秘密区域门户，我们转到发送电子邮件链接：

我们有什么在这里？有一个secret_area_目录...让我们看看这个目录中的内容。

打开mails.txt文件，我们会看到一个电子邮件地址列表。重新阅读场景，我们知道我们的朋友在13日星期五去了MIA。查看电子邮件列表，其中之一跳出来：

Jasson Killer Friday13@JasonLives.com。让我们使用此电子邮件地址，看看是否可以与被绑架的朋友联系。

回到秘密区域门户，我们转到发送电子邮件链接：

按发送，我们得到以下信息：

好极了！我们能够完成挑战并能够联系我们被绑架的朋友。

得到教训：

1. 即使将文本设置为白色而使代码和密码模糊不清，但这仍然不能阻止我们进入门户。如果不確定如何进行操作，请查看页面源代码。不幸的是，开发人员留下了许多不该存在的宝石。

2. 一旦我们进入了秘密门户。我们查看了帧源，并发现了一个秘密目录。再次，开发人员留下了宝贵的宝石供我们继续

3. 最终，当我们转到Apache服务器时，我们注意到了两个文件。查看场景，我们能够跳过第一个文件，因为它只是一个gif。第二个文件是我们想要的文件.....它包含电子邮件地址列表

4. 打开此文件并重新阅读该场景，我们发现我们想要的电子邮件地址来自 Jason (13日星期五)。

每当您陷入困境时-始终可以查看页面资源！😊

OWASP Hackademic挑战项目-挑战2

下面是方案：

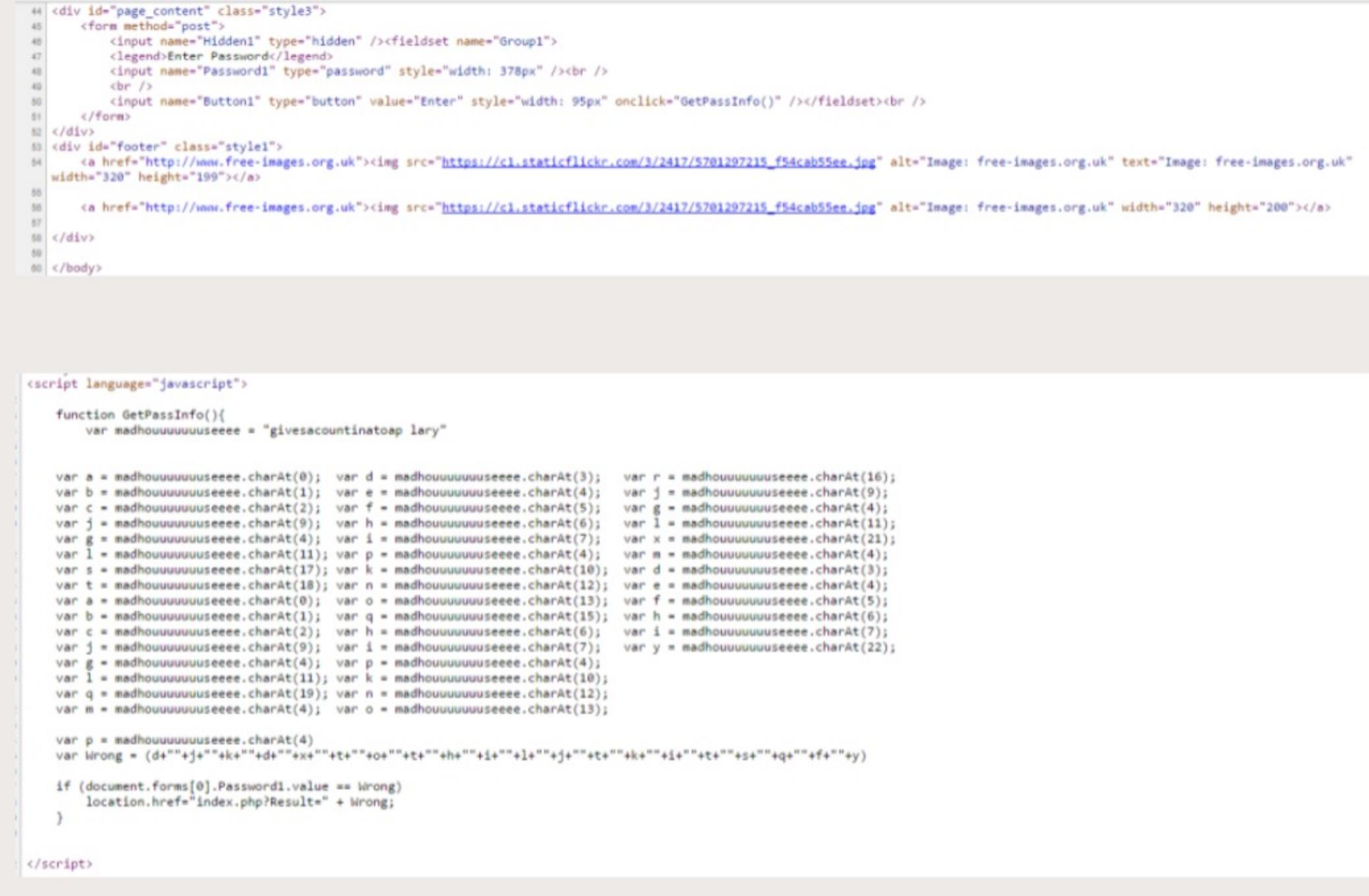
您所在的国家/地区需要您的帮助，以找到包含有用信息的敌方站点的密码，如果该信息得不到及时使用，将危及我们地区的和平。

因此，您必须成功找到该军事站点的密码。

祝好运！

解：

进入站点-我们得到以下信息：



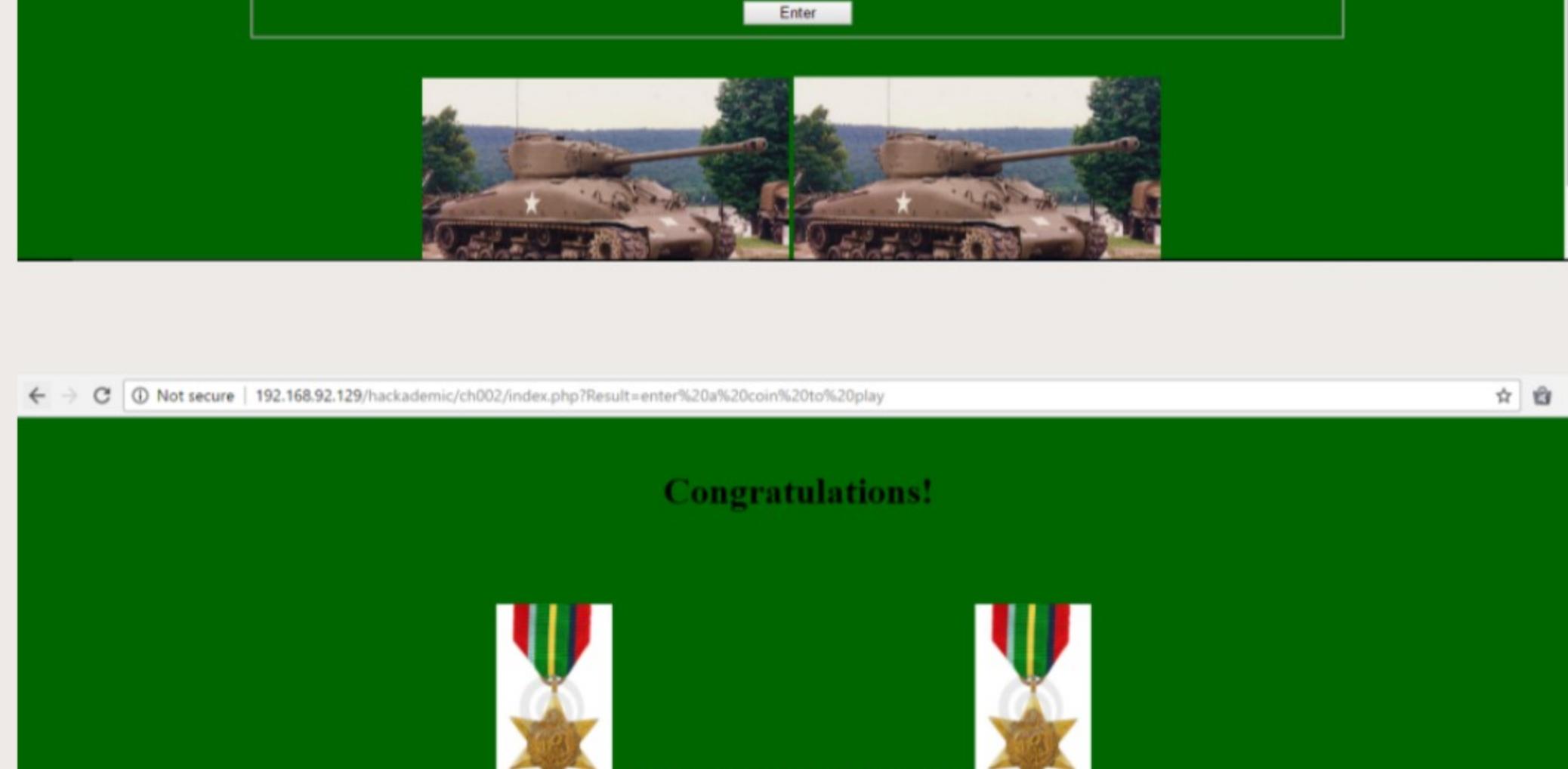
在做页面源代码时，我们看到以下内容：

```
<div id="page_content" class="style3">
<form method="post">
    <input name="Hidden1" type="hidden" /><fieldset name="Group1">
        <input type="password" value=""/>
        <br />
        <input name="Button1" type="button" value="Enter" style="width: 95px;" onclick="GetPassInfo()" /></fieldset><br />
    </form>
</div>
<div id="footer" class="style1">
    <a href="http://www.free-images.org.uk">/a>
    <a href="http://www.free-images.org.uk">/a>
</div>
</body>
```

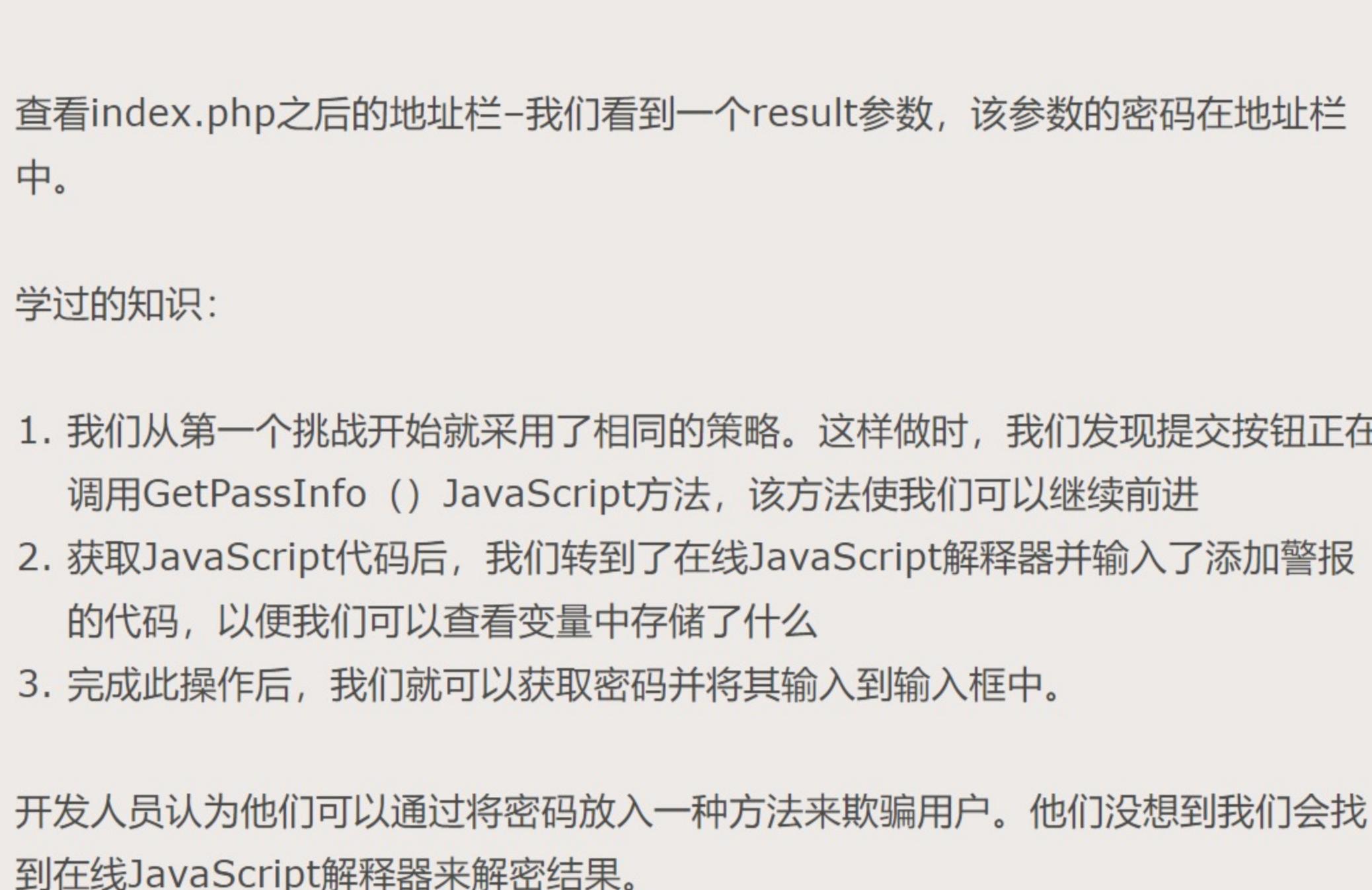
我们看到提交按钮调用了GetPassInfo JavaScript方法（第二个屏幕截图）。

现在我们需要分析该方法返回的结果。

使用在线JavaScript解释器，我们将代码与警报语句一起添加，以打印“错误”变量的内容。



回到挑战，然后在对话框中输入=输入要玩的硬币-我们得到以下屏幕截图：



查看index.php之后的地址栏-我们看到一个result参数，该参数的密码在地址栏中。

学过的知识：

1. 我们从第一个挑战开始就采用了相同的策略。这样做时，我们发现提交按钮正在调用GetPassInfo () JavaScript方法，该方法使我们可以继续前进
2. 获取JavaScript代码后，我们转到了在线JavaScript解释器并输入了添加警报的代码，以便我们可以查看变量中存储了什么
3. 完成此操作后，我们就可以获取密码并将其输入到输入框中。

开发人员认为他们可以通过将密码放入一种方法来欺骗用户。他们没想到我们会找到在线JavaScript解释器来解密结果。

再次-查看页面源再次统治！ 😊

1. 我们从第一个挑战开始就采用了相同的策略。这样做时，我们发现提交按钮正在调用GetPassInfo () JavaScript方法，该方法使我们可以继续前进
2. 获取JavaScript代码后，我们转到了在线JavaScript解释器并输入了添加警报的代码，以便我们可以查看变量中存储了什么
3. 完成此操作后，我们就可以获取密码并将其输入到输入框中。

OWASP Hackademic挑战项目-挑战3

另一天，另一挑战……今天的主题是什么？跨站点脚本（XSS）！

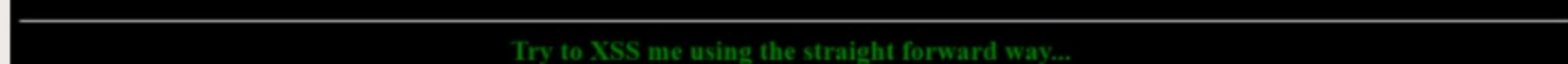
下面的方案：

XSS允许恶意用户在易受攻击的网页中注入自己的代码。根据OWASP 2010十大应用程序安全风险，XSS攻击在“最危险”列表中排名第二。

您的目标是[在此处](#)显示一个警告框，并显示以下消息：“**XSS!**”。

解：

进入挑战，我们将看到以下内容-



在查看页面源代码时，我们注意到有一个POST方法

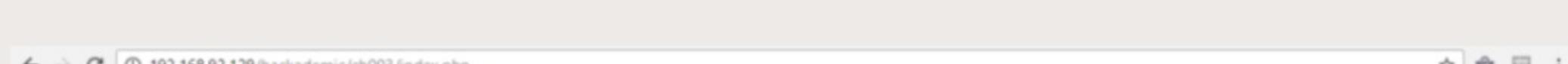


```
1 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
2 <html>
3 <head>
4 <title>Challenge 003</title>
5 <center>
6 <body bgcolor="black">
7 
8 <font color="green">
9 </head>
10 <body>
11 <h2>
12 <hr>
13 Try to XSS me using the straight forward way... <br />
14 <form method="POST">
15 <input type="text" name="try_xss" />
16 <input type="submit" value="XSS Me!" />
17 </form>
18 <hr>
19 </h2>
20 </body>
21 </html>
```

POST方法用于将数据发送到服务器或数据库或另一个文件或API（应用程序编程接口）。

前往Google -我从OWASP找到了一个链接，该链接描述了测试跨站点脚本（XSS）的常用方法。

尝试网页中的项目之一- alert (" XSS! ")； -我得到以下信息：



我们成功地在网页上使用了XSS脚本。

得到教训：

具有输入字段时-作为Web开发人员，我们需要确保输入验证。这样做的原因是我们可以禁止使用无效字符（在这种情况下为脚本标签），因此我们的网站不会受到XSS的影响。

OWASP Hackademic挑战赛-挑战赛4

改天，又挑战。今天的主题是什么？我们仍然在跨站脚本（XSS）领域.....

下面的场景-

一名黑客告知我们，该网站遭受了类似XSS的漏洞。不幸的是，他丢失了自己写的关于他如何充分利用上述漏洞的注释。

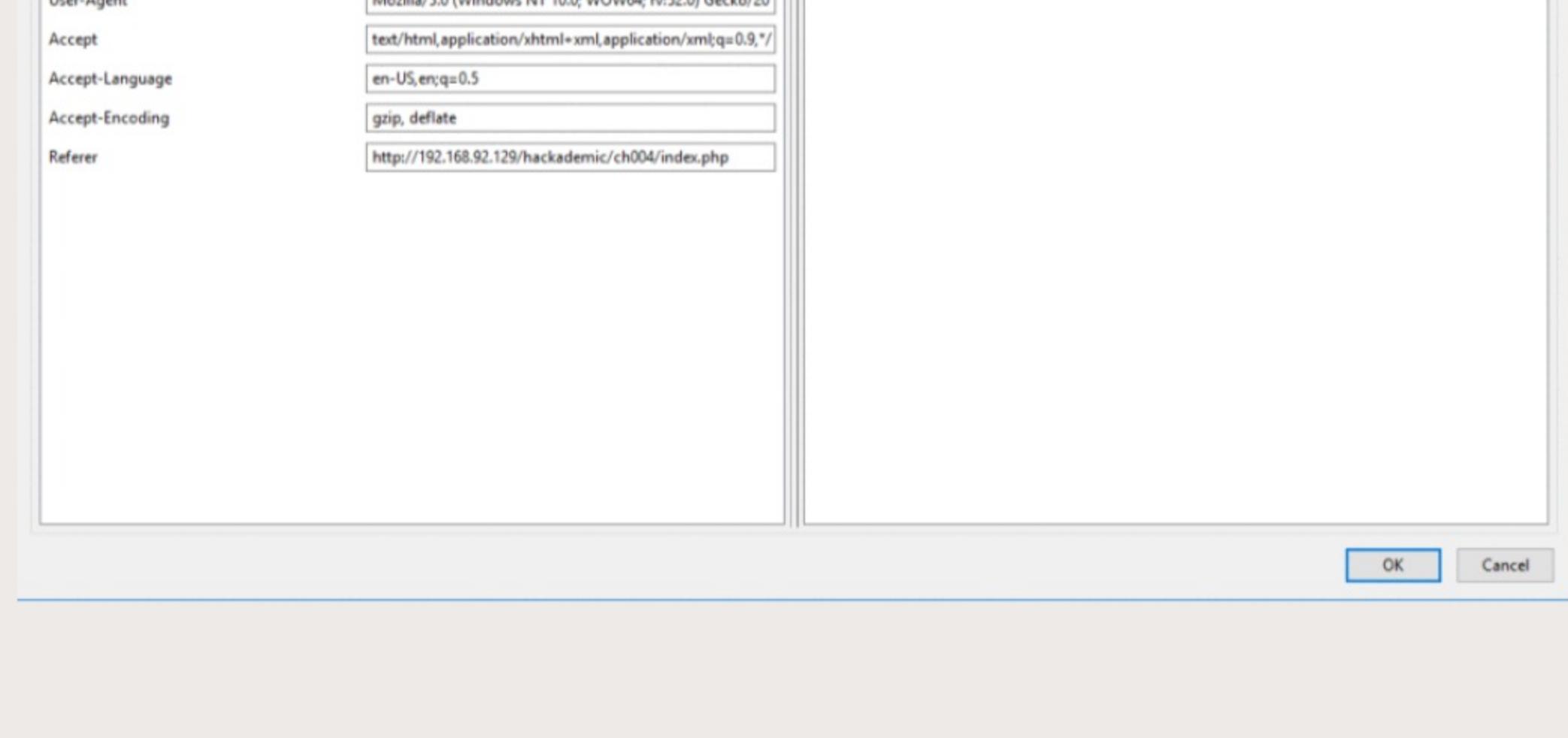
您的目标是在显示警告框的同时显示消息“**XSS !!**”。但是，应该指出的是，该站点对这种攻击具有一定的保护作用。

解

进入我们有以下页面的站点



尝试使用来自第三个挑战（`alert (" XSS! ");`）的相同策略，我们得到以下内容



我们看到这行不通。嗯-似乎开发人员已经在页面上添加了一些验证。

让我们看看是否可以使用XSS进行输出编码。我们的目标仍然是尝试显示XSS的警报框！

使用FireFox的TamperData，我们看到单词已被编码。



去谷歌和寻找“XSS筛选规避小抄”我们得出以下页面[HERE](#)

向下滚动，我们看到以下内容：

fromCharCode

If no quotes of any kind are allowed you can eval() a fromCharCode in JavaScript to create any XSS vector you need:

```
<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
```

让我们尝试使用fromCharCode，看看是否可行。

将XSS更改为等效的ASCII -我们得到以下信息：

```
alert (String.fromCharCode (88,83,83,33) )
```

将其放入文本框，我们进入“篡改数据”内部：

OWASP Hackademic挑战5

另一天，另一挑战.....

今天的挑战是什么？挑战10之5，来自OWASP Hackademic。

下面的方案：

您需要访问此SITE的内容。为此，必须购买“ pOwnBrowser”网络浏览器。由于它太昂贵了，因此您必须以某种方式“欺骗”系统，以便它可以让您阅读站点的内容。

通过下面的内容：

转到网站，我们得到以下信息：



查看页面源代码，我们看到以下内容：

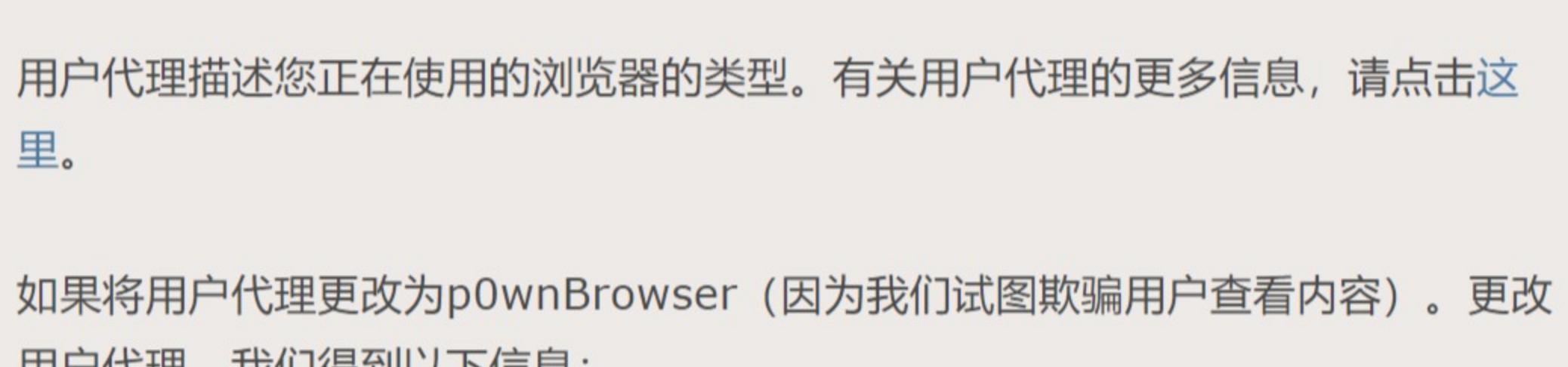
```
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1253">
<title>Challenge 005</title>
<style type="text/css">
.style2 {
    font-size: xx-large;
    color: #0000FF;
}
.style3 {
    color: #808000;
}
</style>
<center>
<body bgcolor="black">

<font color="green">
</font>
</body>
</center>
<br><br>Unfortunately, you cannot access the contents of this site...<br>
In order to do this, you must buy pOwnBrowser. It only costs 3500 euros.
</html>
```

所以.....页面源没有帮助我们，因为那里没有宝石。

让我们看一下篡改数据。

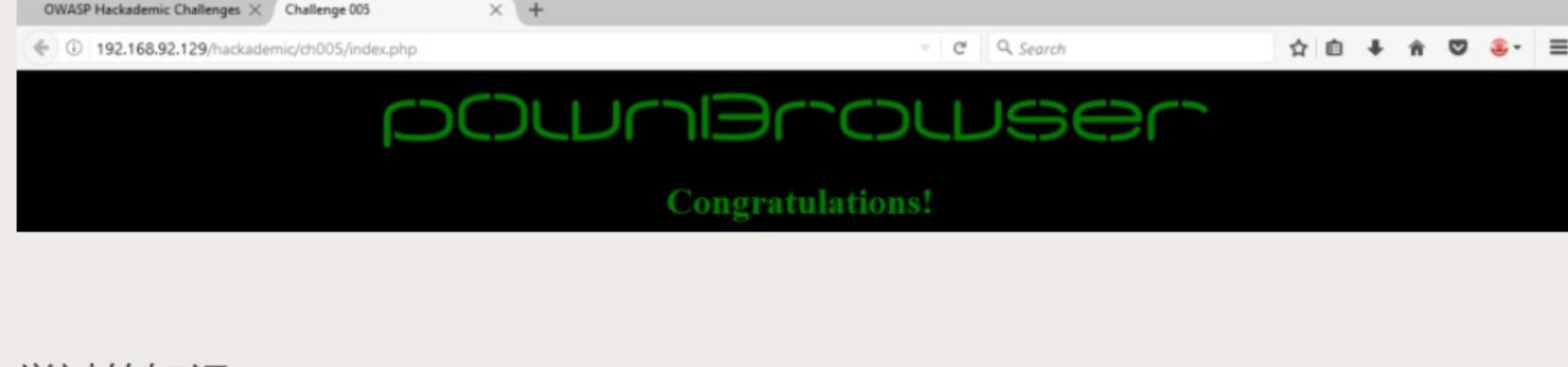
重新加载页面并按“篡改”，我们得到以下信息：



我们没有任何要更改的POST数据，但是我们确实有可以更改的请求标头。一个字段看起来很有趣。

用户代理描述您正在使用的浏览器的类型。有关用户代理的更多信息，请[点击这里](#)。

如果将用户代理更改为pOwnBrowser（因为我们试图欺骗用户查看内容）。更改用户代理，我们得到以下信息：



学过的知识：

在这种情况下，我们没有POST数据（响应）要更改，但是我们确实有GET数据（请求）要更改。查看不同的标题，用户代理跳出一个来进行更改。再说一次，如果您遇到问题了，那就去Google吧！

OWASP Hackademic挑战

在这项作业中，您必

也许您可以找到它?
让我们来看看!
q00d运气帅哥!

点击我们将

HIDDEN SITE

SECRETS & CODES OF KNIGHT EPOCH

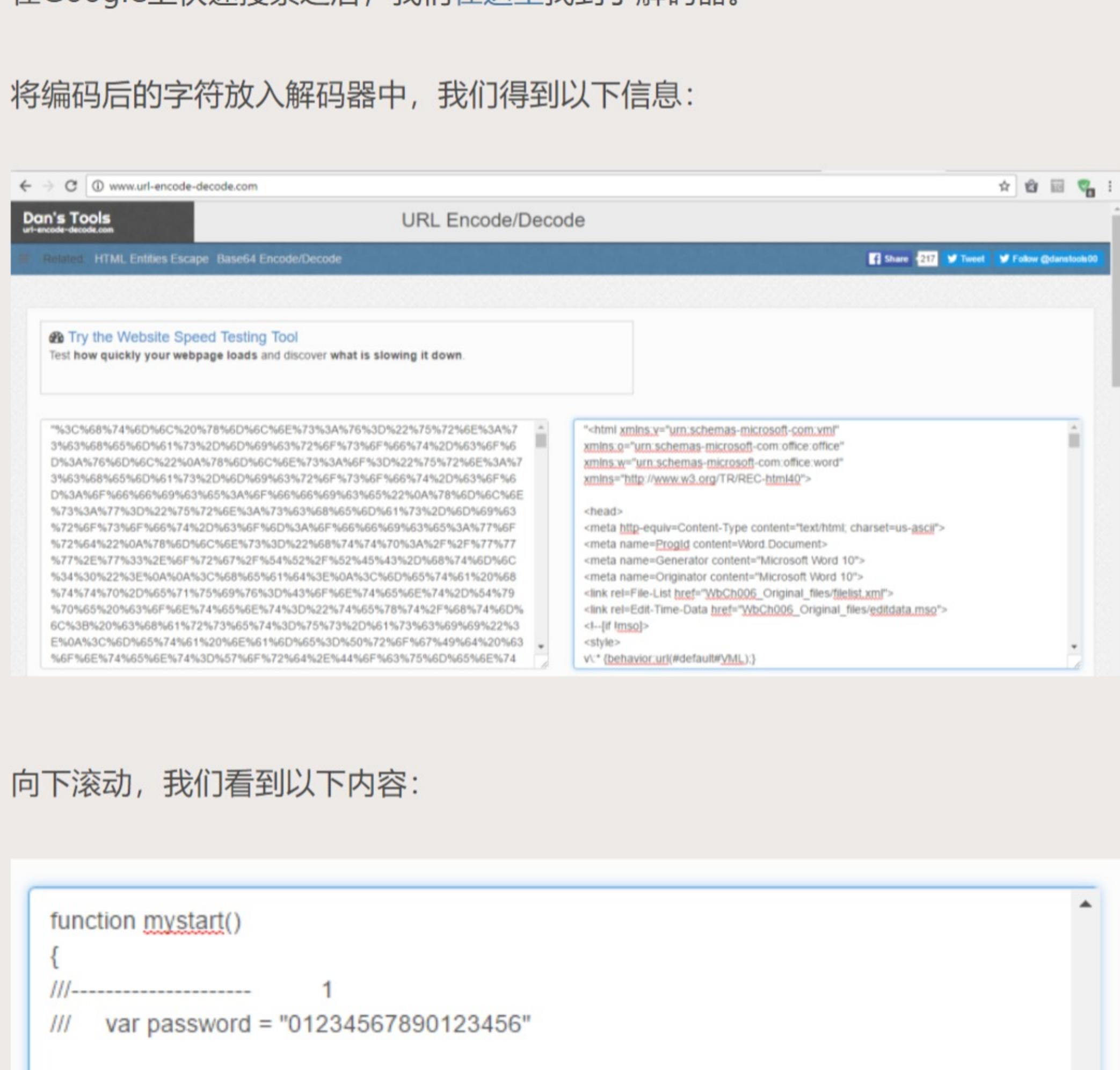
For more information about the study, please contact Dr. John Smith at (555) 123-4567 or email him at john.smith@researchinstitute.org.

查看页面源代码，我们看到以下内容：

5%4%20%68%74%60%6C%34%30%22%
7%8%74%2F%68%74%60%6C%38%20%
%63%75%6D%65%6E%74%3E%0A%3C%
5%74%61%20%6E%61%60%65%3D%40%
6C%65%2D%4C%69%73%74%20%68%74%
%3D%45%64%69%74%2D%54%69%6D%
C%21%2D%2D%5B%69%66%20%21%D%
2A%20%7B%62%65%68%61%76%69%
%40%45%29%3B%7D%0A%2E%2T%73%68%
0%2D%2D%3E%0A%3C%74%69%74%60%

77%73%
%64%20%
%61%6%
61%65%
%73%69%
9%6E%6%
74%79%
%69%6E%
9%4E%6%

脚本标记告诉我们代码是
数-意味着数据正在写入
取消它



```
        alert("Wrong Code...!!");  
    }  
</script>  
<!--[if ate mso 9]><xml>
```

嗯...在mystart函数中有一个注释掉的密码。

我们还有另一个函数GetPassInfo ()，用于检查表单的值以查看其是否简单。如果是这样，那么结果很容易，否则就是错误的代码。

让我们看看是否可以找到在哪里使用这些功能...

向下滚动一些我们使用过的函数：

```
<input type=button name=Send value="Check Code" onClick="mystart()"  
onClick="CheckInfo()">  
  
<span class=GramE><i style='mso-bidi-font-style:normal'><span lang=EN-GB  
style='mso-ansi-language:EN-GB'>cODE</span></i></span></span><span lang=EN-GB  
style='mso-ansi-language:EN-GB'> </span><INPUT TYPE="password" MAXLENGTH="20"  
SIZE="20" NAME="PassPhrase"  
onChange="GetPassInfo()"><span lang=EN-US style='mso-ansi-language:EN-US'><span  
style='mso-tab-count:2'> </span><o:p></o:p></span></p>  
  
<p align=center style='text-align:center;background:#FFCC00;border:none;  
mso-border-alt:solid windowtext .5pt;padding:0cm;mso-padding-alt:1.0pt 4.0pt 1.0pt 4.0pt'>  
<span  
lang=EN-US style='mso-ansi-language:EN-US'><o:p> </o:p></span></p>
```

我们看到单击按钮时，该按钮将调用mystart和checkinfo函数。

让我们尝试输入第一个注释掉的密码：01234567890123456，我们得到以下信息：

① Not secure | 192.168.92.129/hackademic/ch006/index.php

HIDDEN SITE

SECRETS & CODES OF KNIGHT EPOCH

因此，这不是正确的密码。让我们尝试输入：easyyyyyyy！

如您所见，这是正确的密码。

学过的知识：

1. 如有疑问，请 **查看页面资源**。页面源代码为测试人员提供了可以使用的GEMS或让您知道要进入的方向
2. 您可以找到有关所使用的HTML标记所使用的语言的很多信息。开发人员试图通过模糊来实现安全性-意味着通过隐藏来实现安全性。这是通过使用编码和JavaScript函数完成的。我们能够通过解码编码来偏转这种尝试。
3. 查看代码。我知道这对于没有大量编码经验的测试人员可能会很困难。这就是Google的用武之地。没有人知道一切，您需要知道何时寻求或寻求帮助。从场景中我们知道我们需要找到解锁挑战的代码。查看代码，我们看到两个可能的代码。第一个代码不起作用，因为该代码已被注释掉。从上面可以看到，第二个代码起作用了。

OWASP Hackademic挑战赛7

改天，又挑战。

这篇文章我们将解决OWASP Hackademic Challenge的挑战7。

下面是方案：

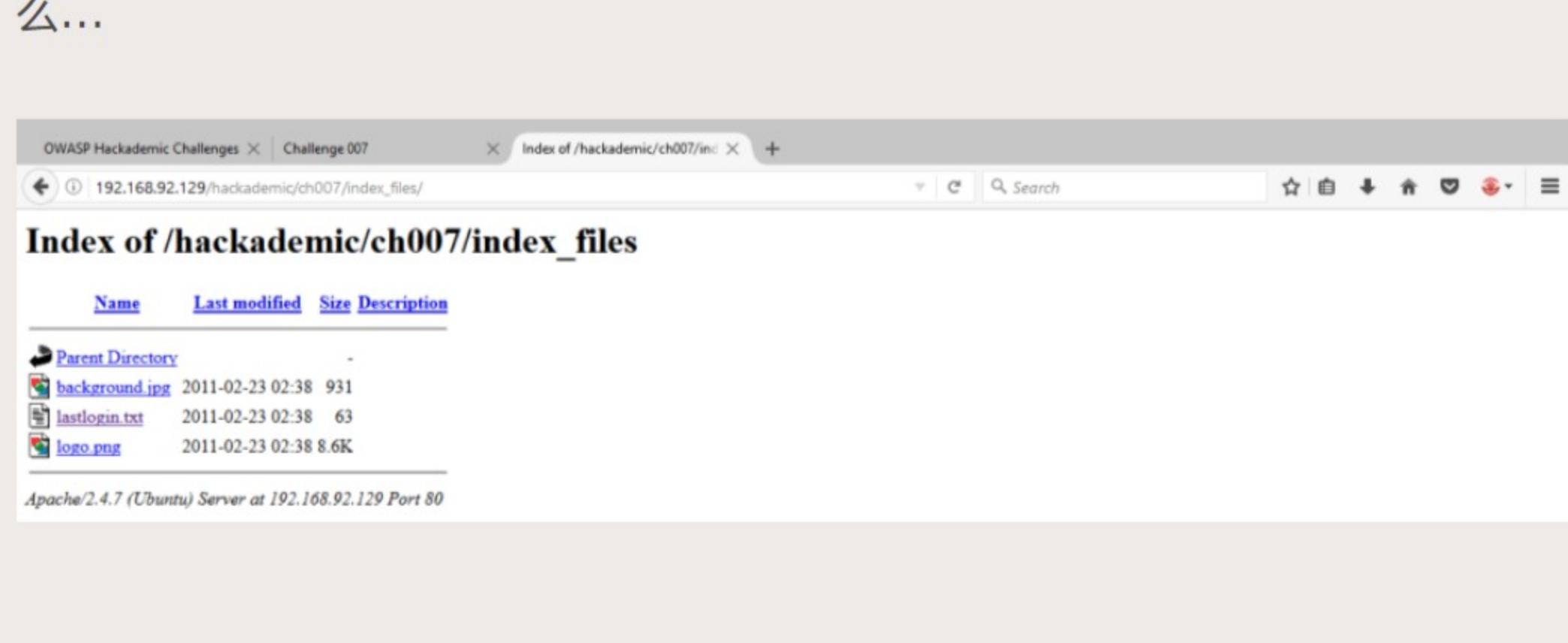
我是Acme大学计算机科学与电信系我的好朋友。不幸的是，她的成绩不是很好。您现在正在思考“这是个大新闻！”.....嗯，也许不是。然而，最大的新闻是：网络管理员要求3,000欧元将她的商标更改为A的商标。这显然是滥用行政权力的情况。因此，.....是D阶段和公众曝光的好机会.....

我需要以管理员身份进入网站，并在Web根目录中上载index.htm文件，该文件将提供大学最新“重新标记”的所有必要证据。做法！

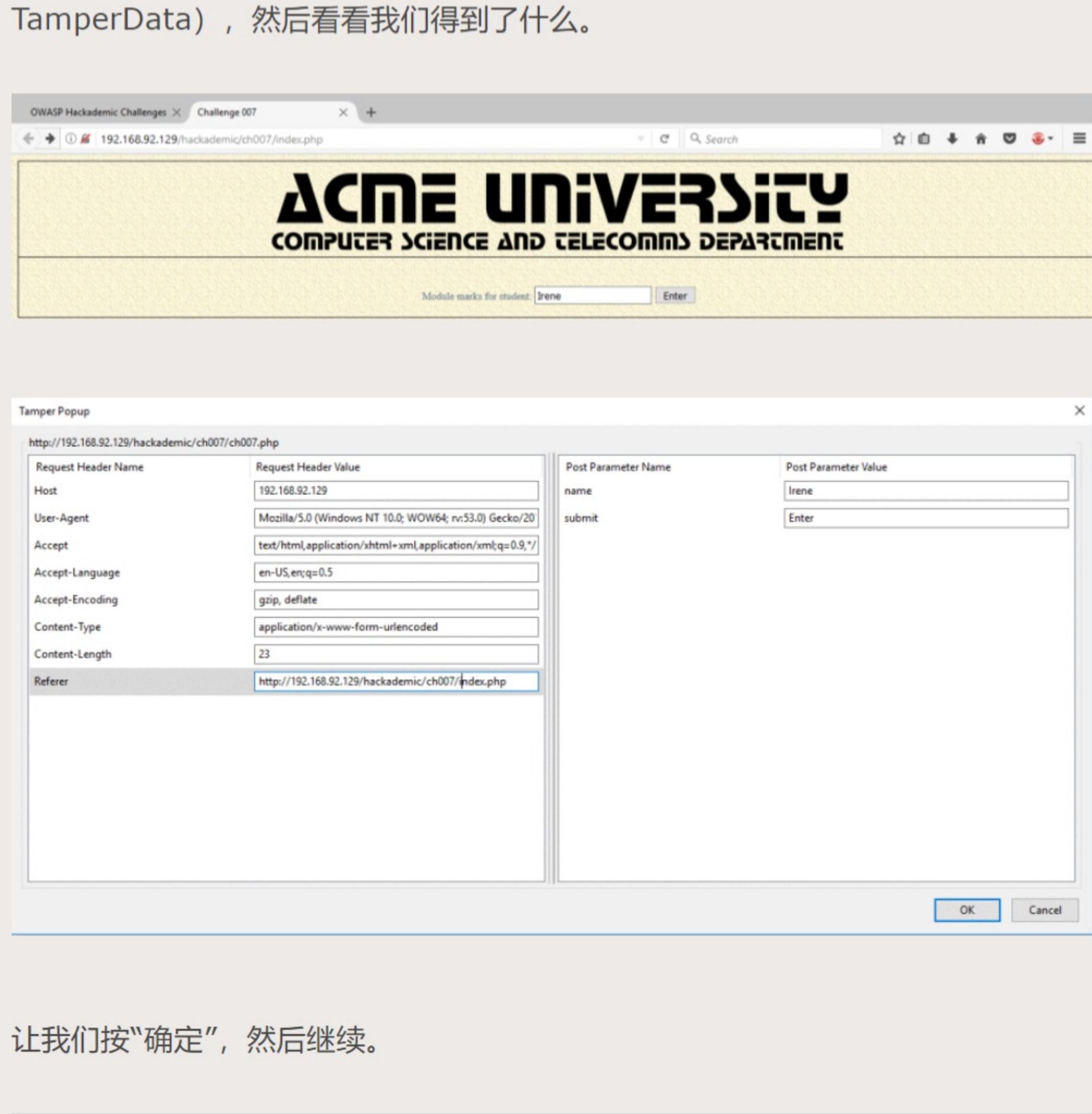
我只需要您为我找到管理员密码...

祝好运！

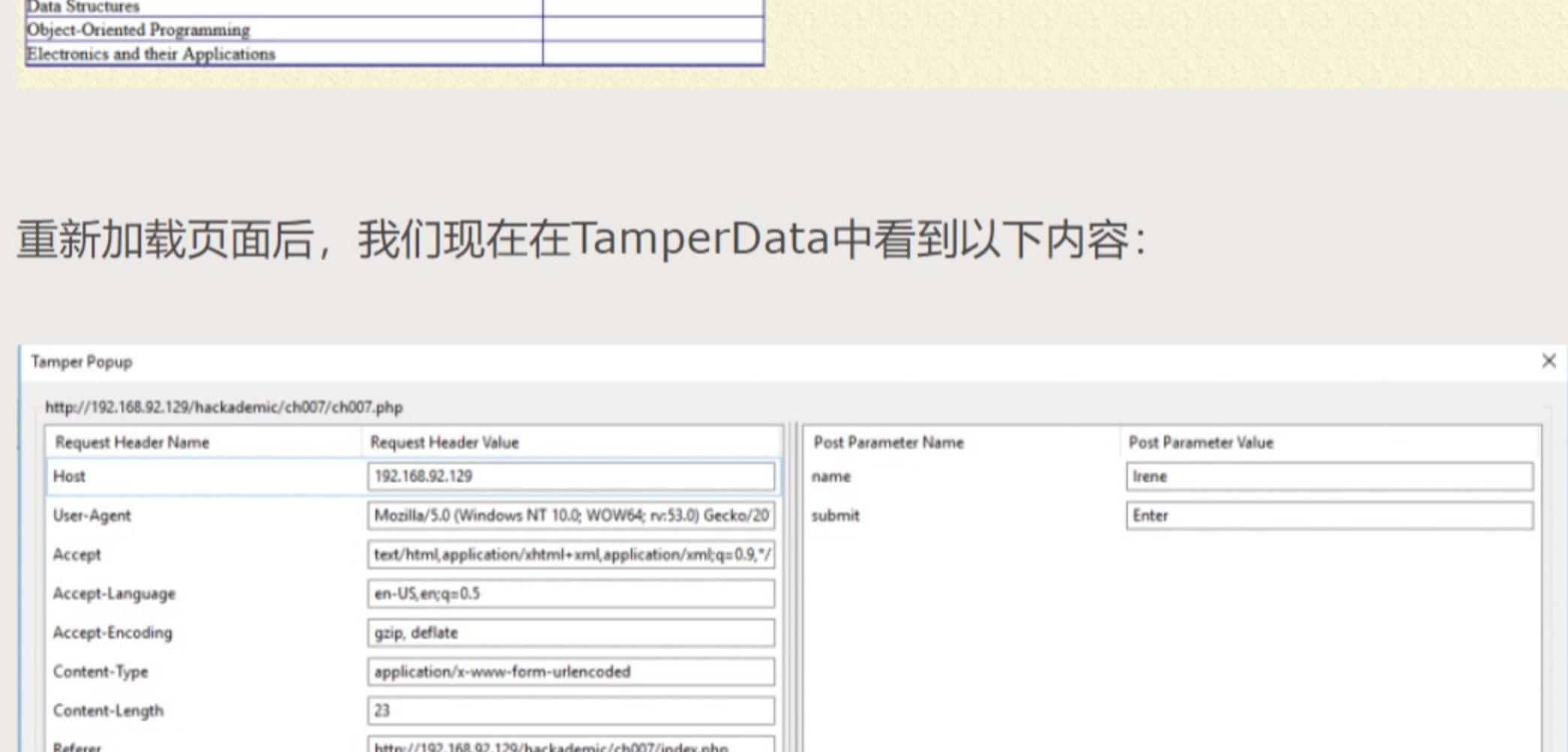
单击该链接，我们将看到以下内容：



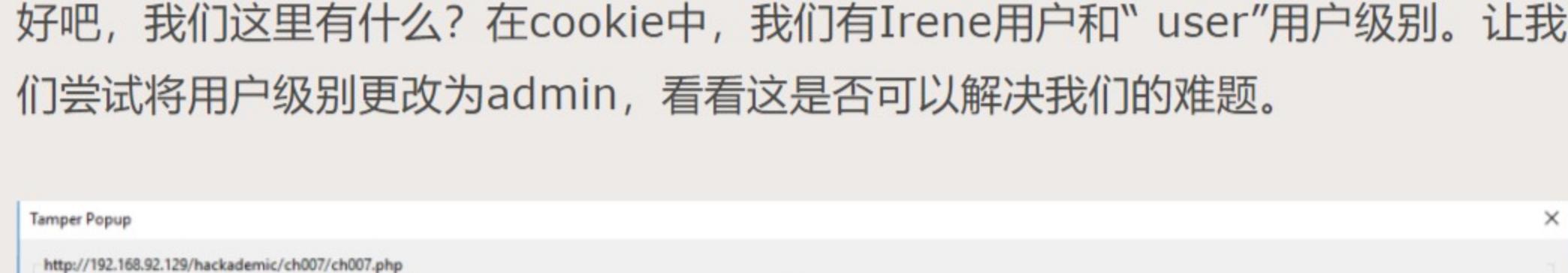
右键单击页面，我们将看到以下内容：



我们看到有一个名为index_files的文件夹。让我们去这个文件夹，看看那里有什么...



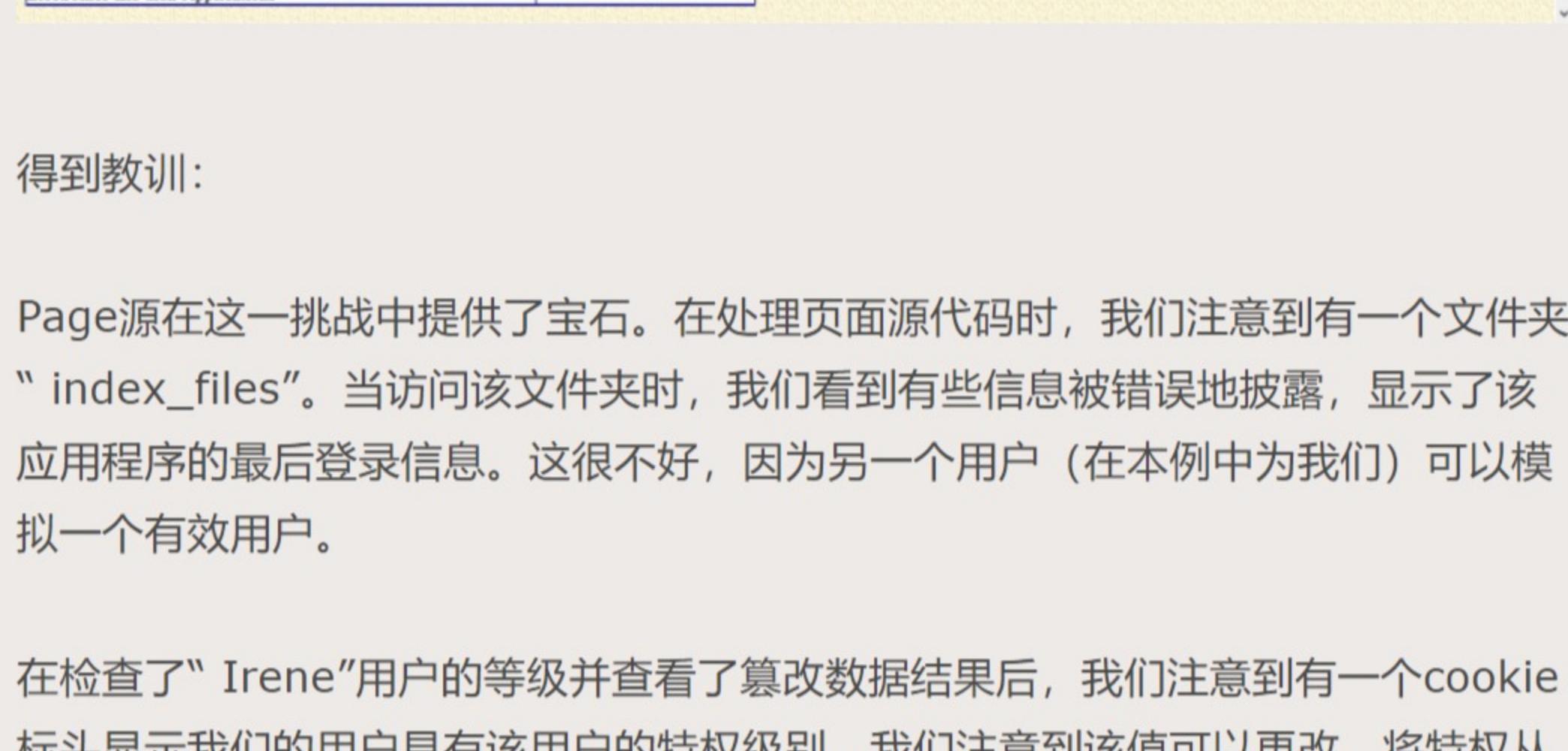
好吧，看这里有什么.....有一个lastlogin.txt，单击该文件，我们得到以下信息：



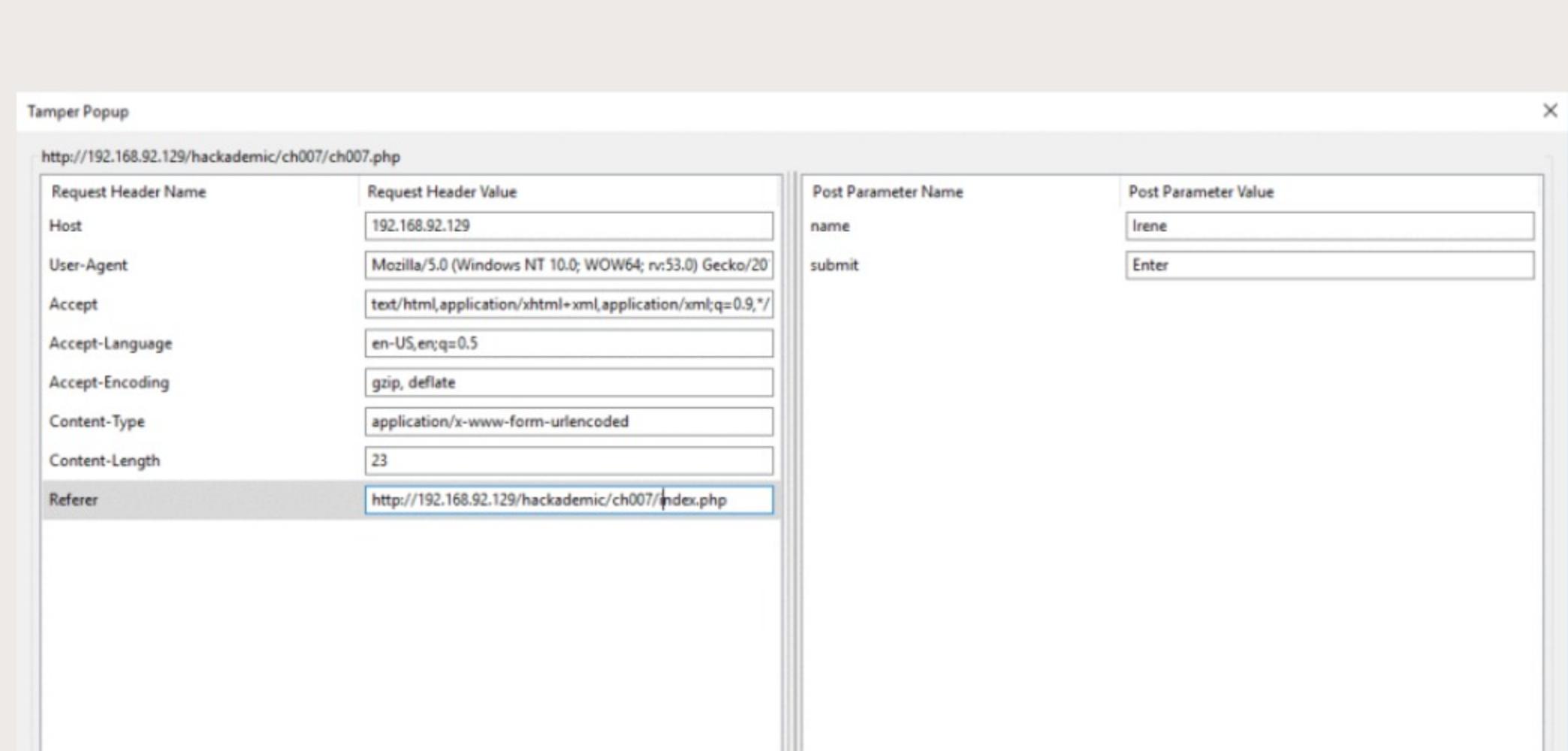
我们看到Irene是有效的用户。让我们回到开头，将Irene添加到文本框中（启用TamperData），然后看看我们得到了什么。



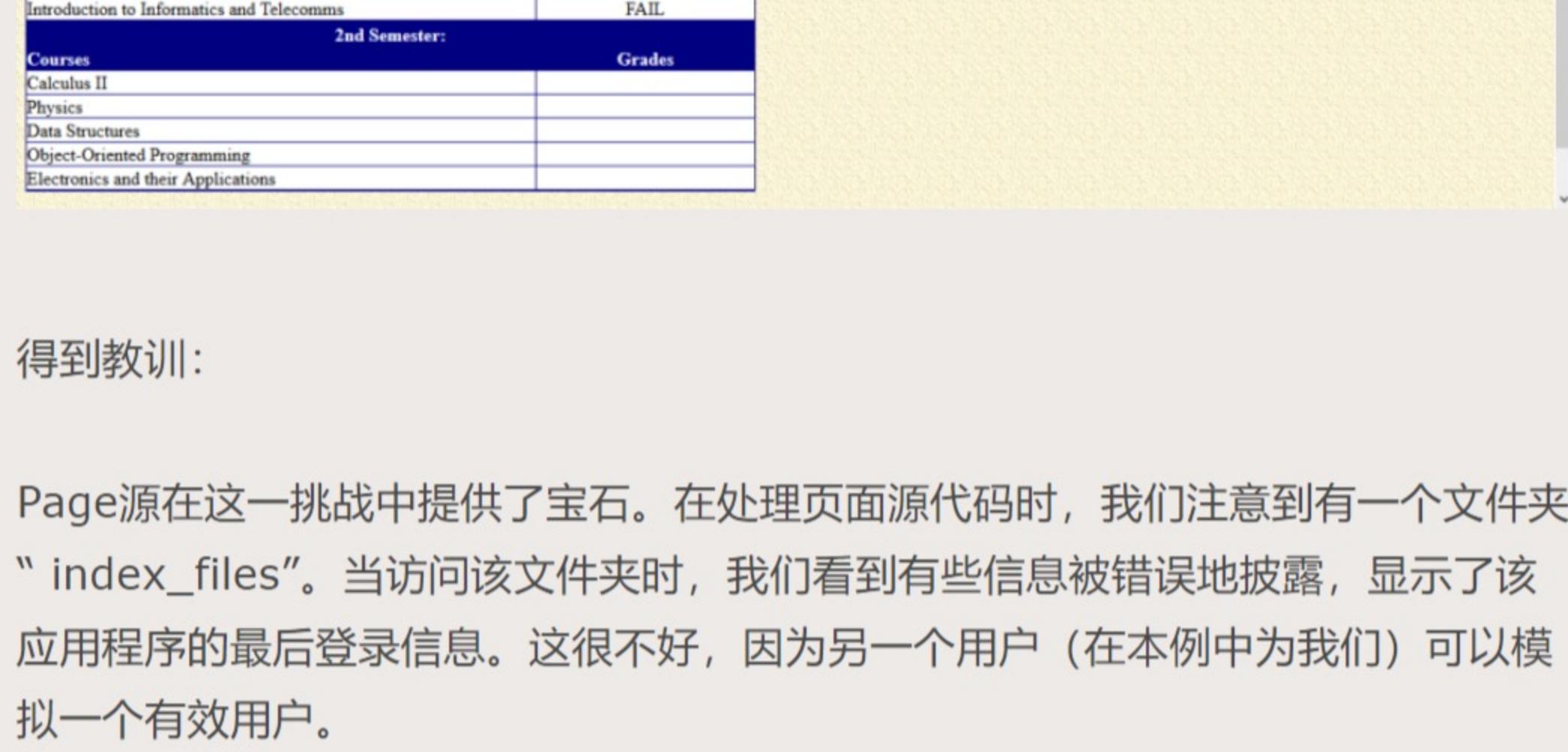
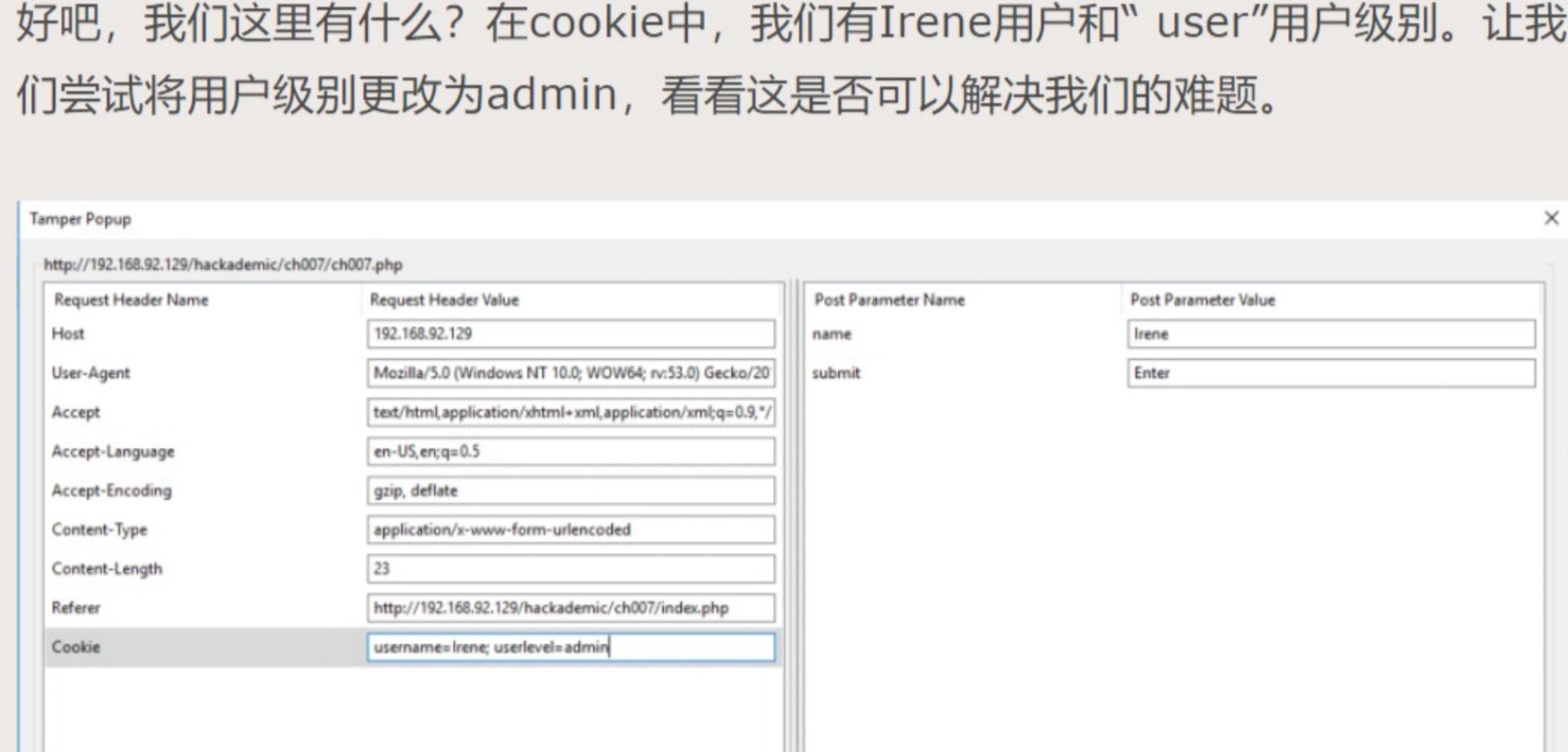
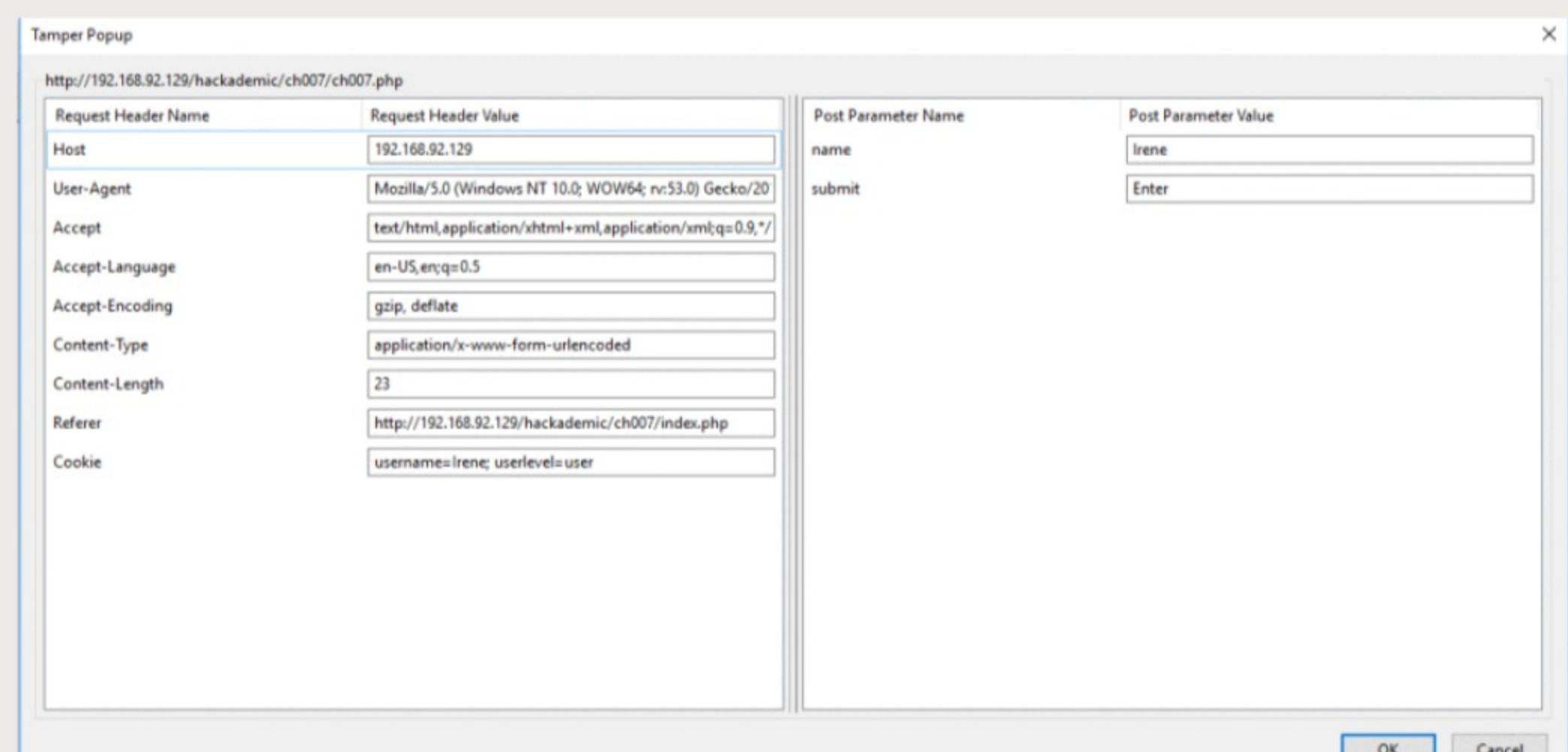
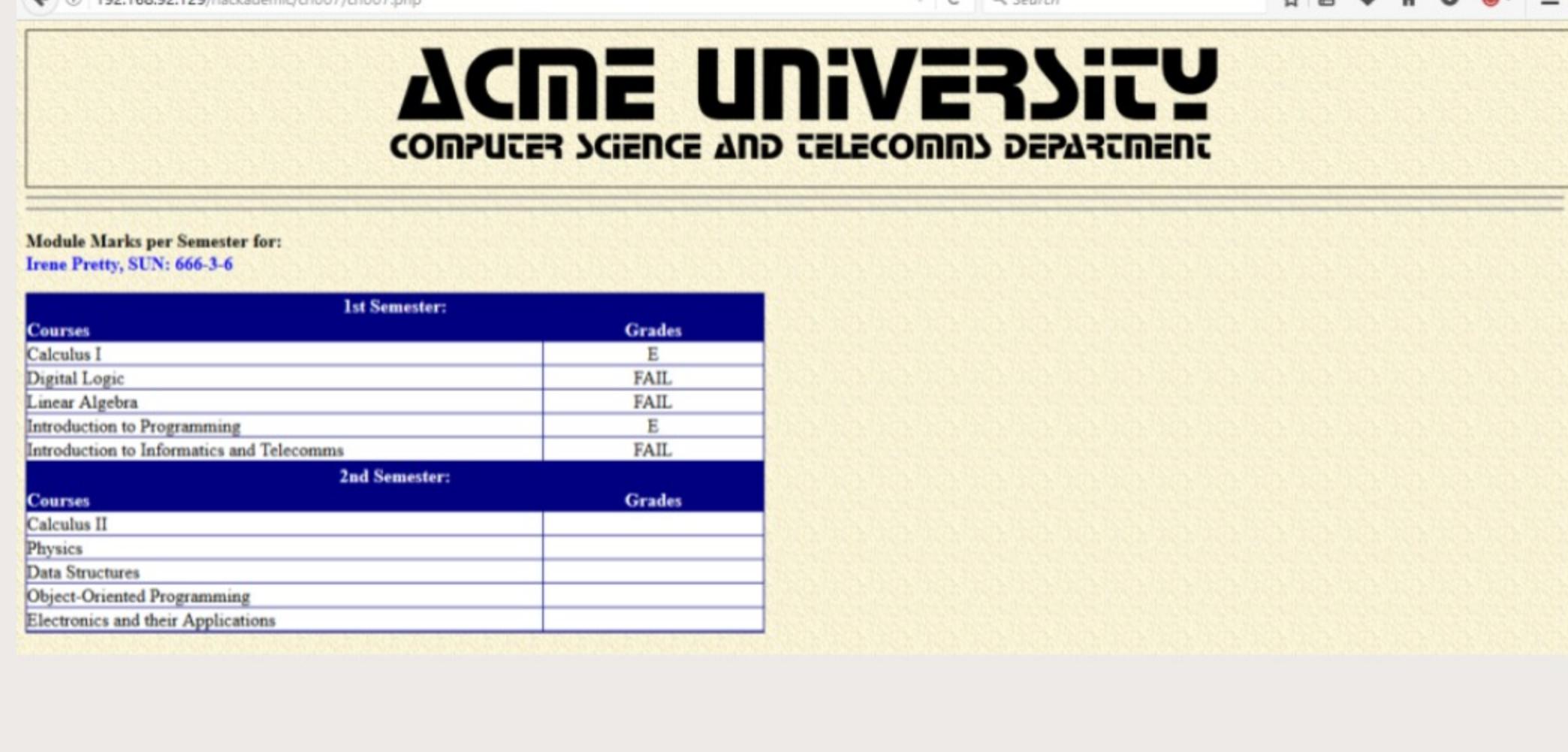
让我们按“确定”，然后继续。



好吧，我们这里有什么？在cookie中，我们有Irene用户和“user”用户级别。让我们尝试将用户级别更改为admin，看看这是否可以解决我们的难题。



按下后，我们得到以下屏幕：



Page源在这一挑战中提供了宝石。在处理页面源代码时，我们注意到有一个文件夹“index_files”。当访问该文件夹时，我们看到有些信息被错误地披露，显示了该应用程序的最后登录信息。这很不好，因为另一个用户（在本例中为我们）可以模拟一个有效用户。

在检查了“Irene”用户的等级并查看了篡改数据结果后，我们注意到有一个cookie标头显示我们的用户具有该用户的特权级别。我们注意到该值可以更改。将特权从用户更改为管理员后，我们成功完成了挑战。

创建应用程序时，请确保没有不正确地披露信息。确保没有打开的文件夹可以在网站上访问。

OWASP Hackademic挑战

今天的挑战是

经过几次尝试，您已经设法将后门外壳程序（Locus

用戶权限 (

单击该链接，我们将看到以下内容：

LOTUS SHELL

Disabled functions: **NONE**
cURL: **OFF**
Register globals: **ON**
MySQL: **ON**

/apps/web/html/trytohack/upload_files/ drwxrwxrwx
Free 2.76 GB of 19.69 GB (14.05%)



Oracle: OFF
Safe-mode: OFF (not secure)
/apps/web/html/trytohack/upload_files/ drwxrwxrwx



← → 🔍 Secure | https://www.base64decode.org

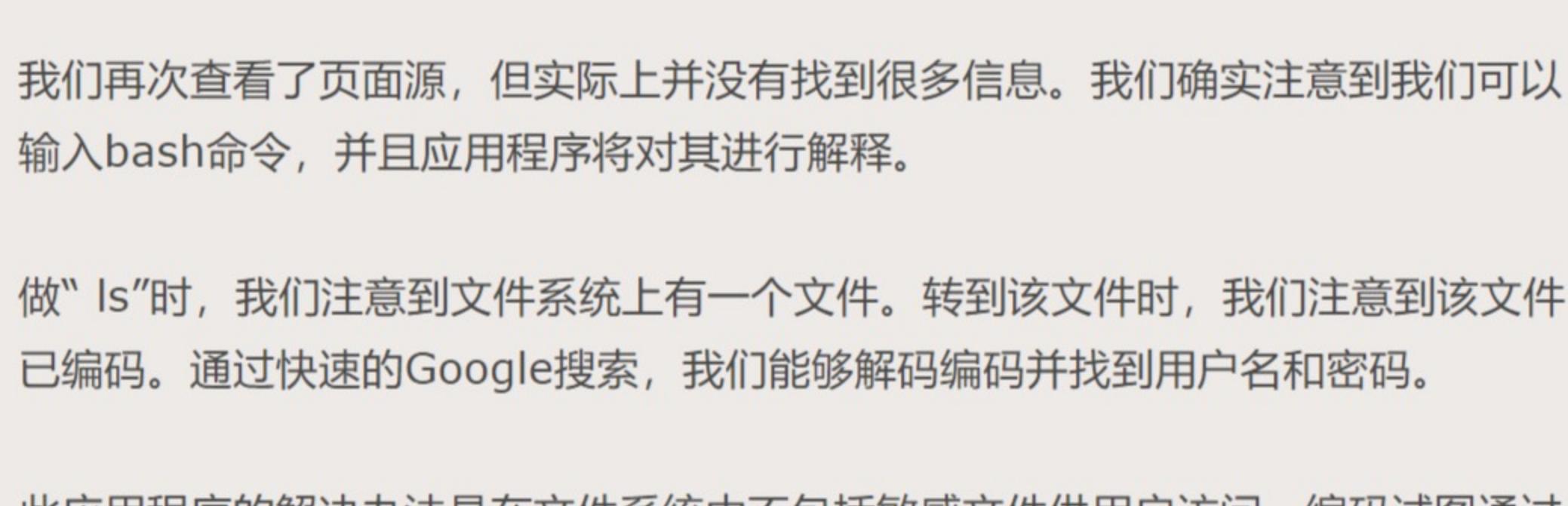


MySQL: ON
MSSQL: OFF
PostgreSQL: OFF
Oracle: OFF

Safe-mode: OFF (not secure)

uid=0(root) gid=0(root) groups=0(root)

并在底部表示祝贺)



做“ls”时，我们注意到文件系统上有一个文件。转到该文件时，我们注意到该文件已编码。通过快速的Google搜索，我们能够解码编码并找到用户名和密码。

OWASP Hackademic挑战赛

您的一个朋友
的新闻的安全

提示：可以在“http://www.really_nasty_hacker.com/shell”处的“通过”

单击第一个

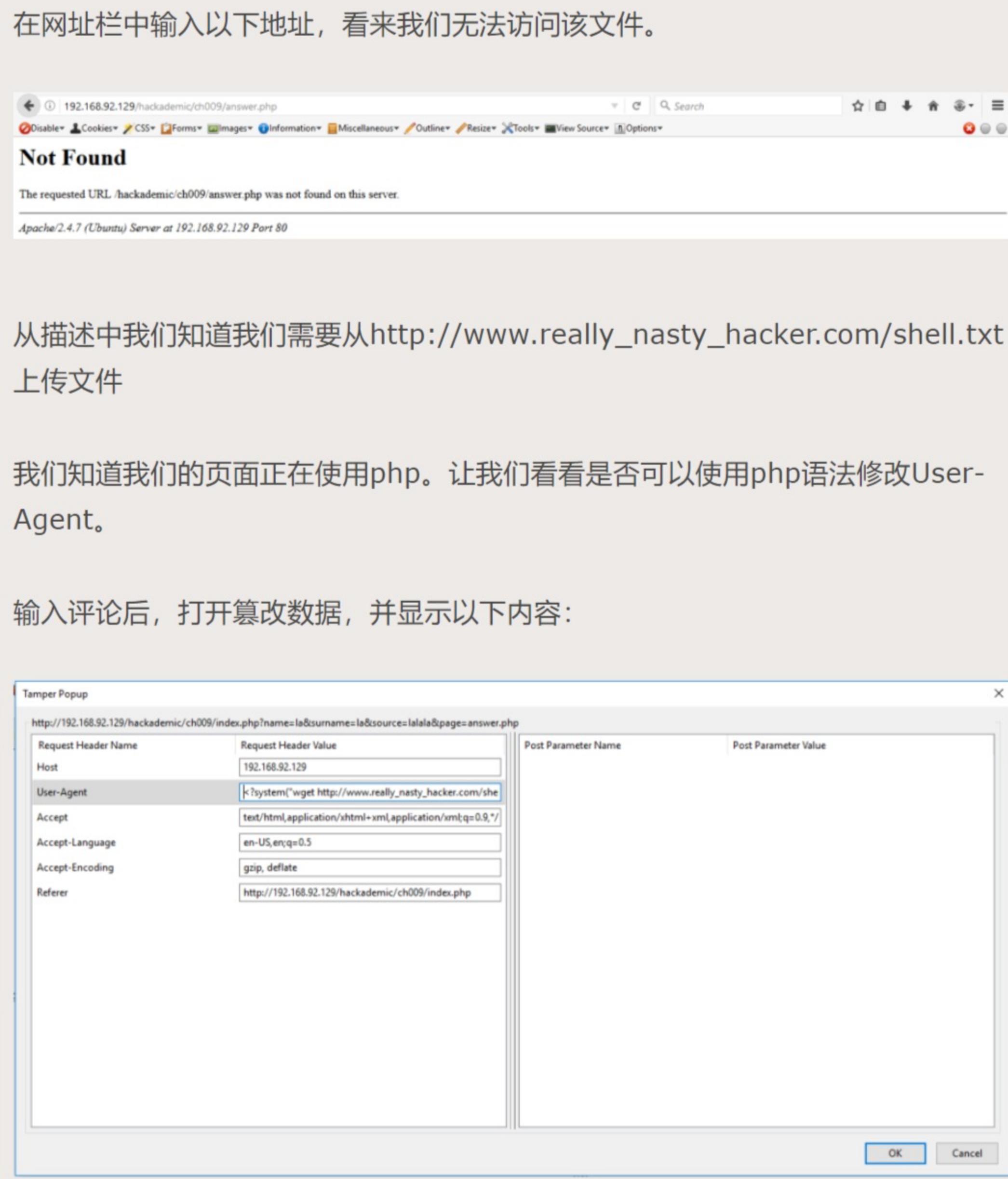
Slagoff

Last Name:	<input type="text"/>
Comment:	<input type="text"/>

```
10 color:lime;
11 }
12 </style>
13 </head>
14 <hr>
15 <body style="background-color: black; color: rgb(0, 0, 0);>
16 <div style="text-align: left; color: white;"><big style="font-family: Arial;">
17 <big><big>The Police have arrested a team of 20 hackers.<br>
18 <small><small><font color="yellow">Comment by Admin, 14 June, 2010 </font><br><br>
19
20 <small>
21 The Police have arrested a team of 20 hackers, belonging to a certain hacker group, who were trying to clone bank websites, in an attempt to intercept bank account
22
23 One more hacker, member of the same hacker group, has been arrested for obtaining illegal access to the servers of several US universities and state organisations,
```

```
34 </form>
35 <form method="GET" action="http://www.google.com/search">
36 <font color="yellow">
37 <input type="text" name="q" value="Google" style="width: 200px;" />
38 <font color="yellow">
39 <input type="text" name="sitesearch" value="www.google.com" style="width: 200px;" />
40 <font color="yellow">
41 <textarea name="source" style="width: 200px; height: 100px;" />
42 <br/>
43 <input type="submit" value="Search" style="width: 100px; height: 30px;" />
44 </form>
```

我们注意到有一个值为answer.ph



The screenshot shows a web browser window with the URL 192.168.92.129/hackademic/ch009/index.php?name=la&surname=la&source=lala&page=answer.php. The page title is 'slagoff'. The main content reads: 'The Police have arrested a team of 20 hackers.' Below it is a comment by Admin from June 14, 2010, detailing a hacking operation. A text input field for comments is present, followed by a note about a successfully installed backdoor shell. The browser toolbar includes various developer tools like Disable, Cookies, CSS, Forms, Images, Information, Miscellaneous, Outline, Resize, Tools, View Source, and Options.

我们的脚本已成功上传！

转到新站点，我们看到以下内容：

Software: Apache/2.2.16 (Fedora). PHP/4.4.9
uname -a: Linux prwtoslagoff 2.6.34.7-61.fc13.i686 #1 SMP Tue Oct 19 04:42:47 UTC 2010 i686
Disabled functions: NONE
cURL: OFF
Register globals: ON
MySQL: ON
MSSQL: OFF
PostgreSQL: OFF
Oracle: OFF
Safe-mode: OFF (not secure)
/home/prwtoslagoff/public_html dwwwrwxrwx
Free 34.76 GB of 69.69 GB
Available Commands: (ls, id, whoami, pwd)
execute command

嗯.....我们看到的命令行类似于挑战8！

让我们来看看文件系统上的内容。

adminpanel.php
tyj0rL.php
dUpErDuPErL33T.txt

A screenshot of a web browser window. The address bar shows the URL: 192.168.92.129/hackademic/ch009/sUpErDuPErL33T.txt. Below the address bar is a toolbar with various icons: Disable, Cookies, CSS, Forms, Images, Information, Miscellaneous, Outline, Resize, Tools, View Source, and Open. The main content area displays the text "Top Secret Information:" followed by a horizontal dashed line.

出现登录信息。

转到该文件，我们看到以下内容

Please login as Administrator
管理员门户!
让我们输入登录信息，看看会得到什么：

我们通过了挑战！！！
得到教训：
我们进行了右键单击试图页面源，显示了一个名为answer.php的隐藏文件。尝试

找到登录信息并成功登录。

利用您所学的知识来解决下一个挑战。我们首先通过p0wnbrowser产品被介绍给挑战5中的用户代理。我们知道，我们可以更改用户代理以显示我们从未注意到的内容。

OWASP Hackademic挑战10

另一天，另一挑战.....

今天的挑战将结束Hackademic挑战。

下面是方案：

您想成为一名活跃的黑客吗？

如何成为世界上最大的黑客组织的成员：

n1nJ4.n4x0rZ.CreW!

但是在加入之前，您必须通过以下网址的测试来证明自己的价值：[http : //n1nJ4.n4x0rZcr3w.com](http://n1nJ4.n4x0rZcr3w.com)

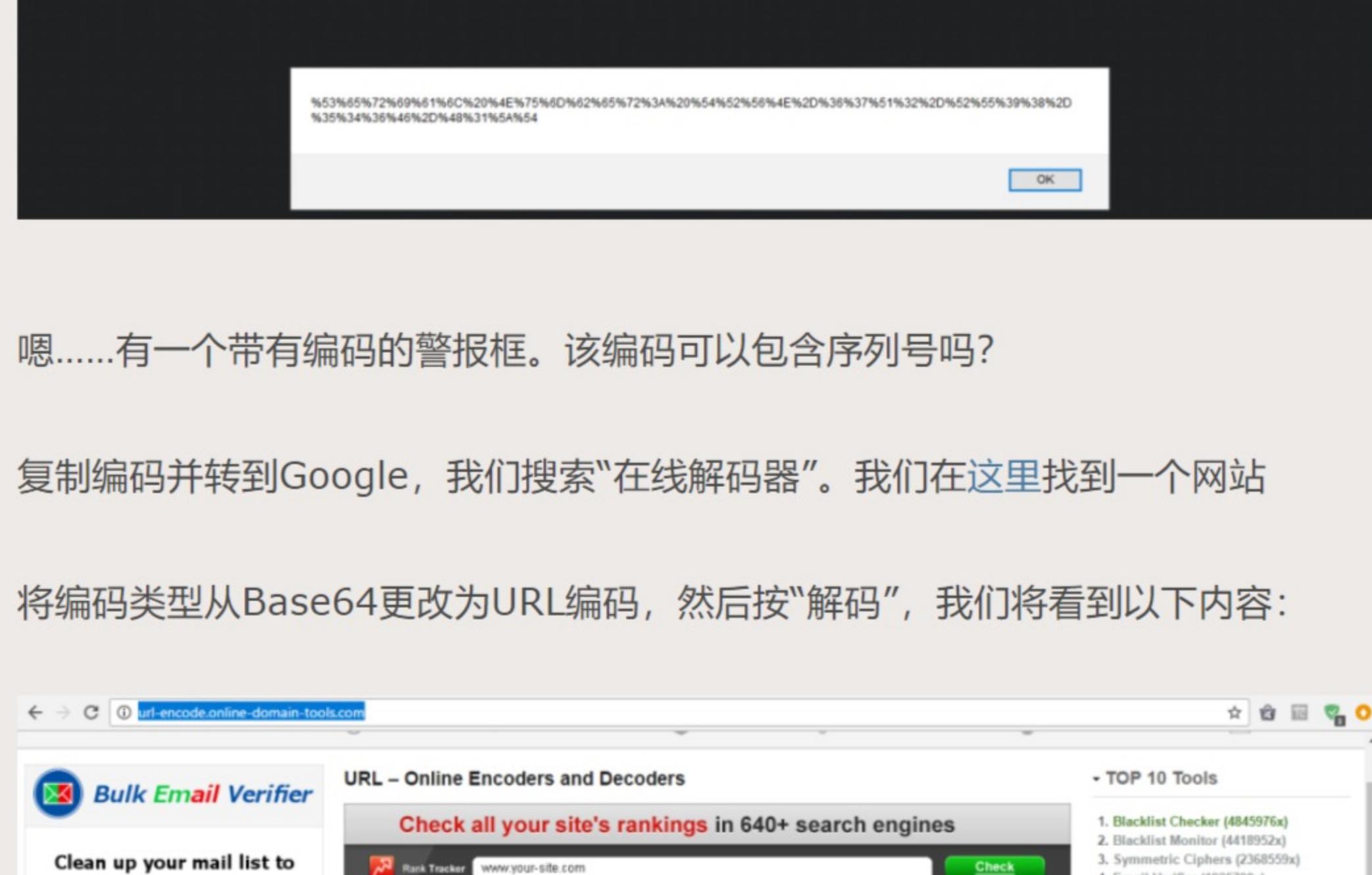
如果您成功完成挑战，您将获得一个序列号，您将使用该序列号来获取使您能够加入小组的密码。

您的目标是绕过身份验证机制，找到序列号，并从站点的管理团队获得您自己的用户名和密码。

单击链接，我们将看到以下屏幕：



右键单击页面源，我们看到以下内容：

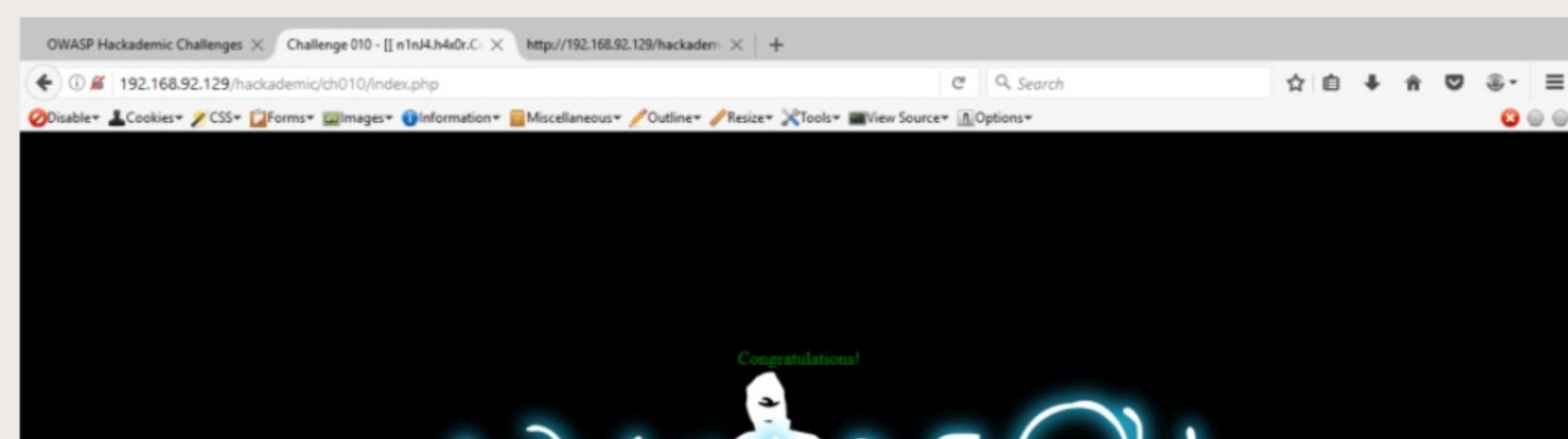


查看密码行上方的行，我们注意到有一个名为“ LetMeIn”的隐藏字段，该字段设置为false。如果将其设置为true怎么办？

回到我们的原始屏幕，然后单击工具-> Web开发人员扩展->表单->显示表单字段，我们看到以下屏幕：



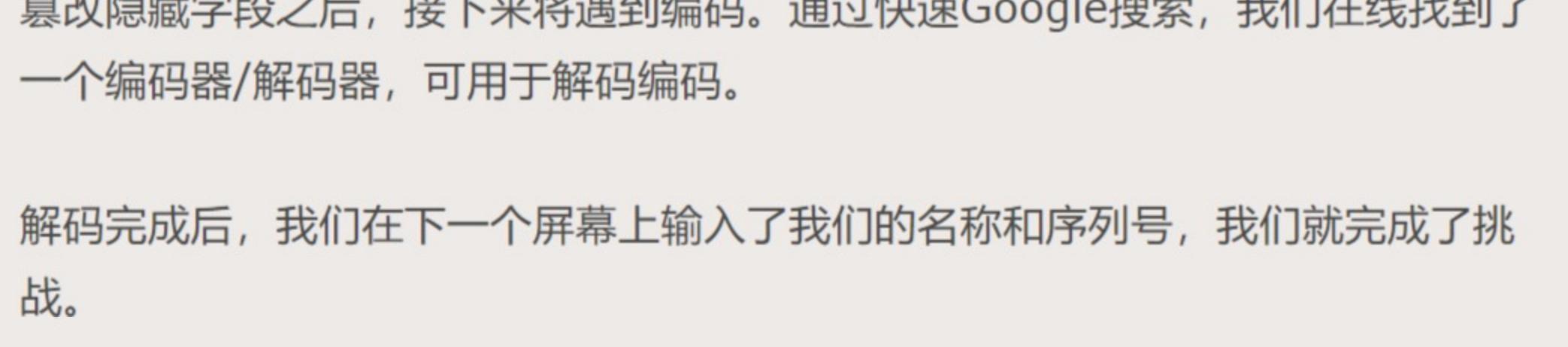
将字段从“ False”更改为“ True”，然后按“ Login”按钮，我们将看到以下内容：



嗯.....有一个带有编码的警报框。该编码可以包含序列号吗？

复制编码并转到Google，我们搜索“在线解码器”。我们在[这里](#)找到一个网站

将编码类型从Base64更改为URL编码，然后按“解码”，我们将看到以下内容：



我们有序列号！

返回到挑战并按Enter，将显示以下屏幕：

输入我们的名称和序列号，然后按发送按钮，我们将看到以下屏幕：

学过的知识：

右键单击和查看页面源的技巧对我们有所帮助。我们注意到有一个标题为“ LetMeIn”的隐藏字段。开发人员认为，仅由于某个字段被隐藏，渗透测试人员就无法利用这些字段。这与事实相去甚远。

篡改隐藏字段之后，接下来将遇到编码。通过快速Google搜索，我们在线找到了一个编码器/解码器，可用于解码编码。

解码完成后，我们在下一个屏幕上输入了我们的名称和序列号，我们就完成了挑战。