攻防世界——Web新手题笔记(一个边做题边学习Web相关知 识的菜鸡)

原创

<u>打怪进阶的菜鸡</u> ● 于 2019-11-03 18:05:55 发布 ● 867 ☆ 收藏 7 文章标签: <u>攻防世界 Web 新手题 笔记</u> 版权声明:本文为博主原创文章,遵循 <u>CC 4.0 BY-SA</u>版权协议,转载请附上原文出处链接和本声明。 本文链接: <u>https://blog.csdn.net/weixin_43908872/article/details/102885011</u> 版权

第一题: view_source

学习chrome打开开发者工具的快捷键,用F12即可,Ctrl+u查看源代码,注意flag的格式为Cyberpeace{xxxxxxx}

第二题: get_post

关于HTTP的两种请求方式get和post 具体实现需要一个Hackbar的插件,在Firefox或Chrome添加即可 再根据提示一次完成,即可得出flag

请用GET方式提交一个名为a,值为1的变量 请再以POST方式随便提交一个名为b,值为2的变量 cyberpeace{fcf252e832d303dd9afff269296a0c9f}

	Elements	Console	Hac	kBar	»		: :
LOAD	URL	SPLIT UF	۲L	EXI	ECUTE	URL	SQ
URL http://1	11.198.2	9.45:4254	0/?a	=1			
e E	nable OST	^{ype} blication/x-	-www	/-forr	n-urlen	CO	•
Body b=2							
							908872

HTTP中GET和POST两种请求方式的区别:

GET:Web浏览器将各表单字段名称及其值按照URL参数格式的形式,附在action属性指定的URL地址后一起发送给服务器。(说 白了就是在地址栏可以看到发送的值)

POST:浏览器将把各表单域元素名称及其值作为HTTP消息的实体内容发送给Web服务器,而不是作为URL参数传递。(数据不 会显示在地址栏)

第三题: Robots

HTTP权威指南中专门有一章写关于Web机器人协议内容的。

大概思路就是爬虫的过程中,会不可避免地遇到一些问题。为控制机器人行为,制定了"拒绝机器人访问标准",而这一标准根据 其存储的文件称为robots.txt。思想就是在爬行目标文件之前,先获取robots.txt,验证是否可以进行访问。 所以,打开页面发现什么也没有时,根据题意,访问robot.txt,

O Not secure | 111.198.29.45:56114/robots.txt

就可以发现flag

User-agent: * Disallow: Disallow: f1ag_1s_h3re.php

接下来就是用同样的方式访问这个.php文件,得到flag

(i) Not secure | 111.198.29.45:56114/f1ag 1s h3re.php

Getting Started -... 📴 Learn to Code - f... 🛛 🍖 GoDaddy Domain.

接下来学习一下,robot.txt文件的格式: 1.User-Agent行 每个机器人记录都以一个或多个下列形式的User-Agent 行开始 2.Disallow行和Allow行 紧跟在User-Agent行之后,显式禁止或允许特定机器人使用哪些URL路径。 于是题目就很好理解了,找到页面无法显示的原因即找到flag

第四题: backup

放一个整理得很全的后缀名文章

各种后缀名详解

由此,我们可以知道.bak是备用文件的后缀,加上.bak后缀执行URL

🕞 💼 📔 Elements	Console Sc	ources Network	Performance	HackBar	>>
LOAD URL	SPLIT URL	EXECUTE UF	RL SQLI	• XS	SS •
URL http://111.198.2	29.45:52584/ <u>in</u>	dex.php.bak	https://blog.cscln.c	nat/waixin_43	908872

第五题 cookie

HTTP权威指南第11章,cookie是识别当前用户,实现持久会话的最好方式。cookie中包含了一个由名字=值(name=value)这样的信息构成的任意列表,并通过Set-Cookie或Set-Cookie2HTTP响应首部将其贴到用户身上去。

接下来回到题目,题目问的是cookie,需要清楚从chrome浏览器中如何找到cookie以及它的HTTP response。所以要熟悉 chrome开发者工具界面。

还是用hackbar, execute题目的URL, 从network下找到cookie

🕞 💼 🛛 Elements Console	Sources Network	Performance	Memory	HackBar »	: ×					
🔴 🛇 🍸 🔍 🗌 Preserve	e log 🔲 Disable cache	Online 🔻	★ ★		\$					
Filter	Hide data URLs All XH	R JS CSS Img	Media Font	Doc WS Manifest	Other					
20 ms 40 ms 60 m	s 80 ms 100 ms	s 120 ms	140 ms	160 ms 180 ms	200 ms 2					
Name	× Headers Preview	v Response	Cookies Tim	ning						
Til.198.29.45										
bootstrap min css	Content-Length: 27	76								
	Content-Type: tex	Content-Type: text/html								
glyphicons-hallings-regular.w	Date: Sun, 03 Nov	Date: Sun, 03 Nov 2019 08:07:49 GMT								
glyphicons-halflings-regular.ttf	Keen-Alive: timeou	Keep-Alive: timeout=5, max=99								
	Company Analysis (2									
	Server: Apache/2.4	4.7 (UDUNLU)								
	Set-Cookie: look-here=cookie.php									
	Vary: Accept-Enco	ding								
4 requests 582 B transferred 9	X-Powered-By: PHF	/5.5.9-1ubuntu	14.26		•					
Console					at/weixin_43908 37 2					

打开cookie.php,找到flag

	Image: Construction Image: Constr	sole Sources Network Performance Memory Application ≫ : serve log □ Disable cache Online ▼	×				
	10 ms 20 ms 30	0 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms	110 1				
	Name	× Headers Preview Response Cookies Timing					
See the http://www.see	cookie.php	Response Headers view source Connection: Keep-Alive					
oce the http response	bootstrap.min.css						
	glyphicons-halflings-regular.v	Content-Encoding: gzip					
		Content-Length: 253					
		Content-Type: text/html					
		Date: Sun, 03 Nov 2019 08:12:00 GMT					
	flag: cyberpeace{0577ad6a893c99e9390b29dd9b5320d5}						
	4 requests 578 B transferred 5						
	Console		×				
	🕩 🛇 top	▼ 💿 djmoeo 🛞 Default levels ▼ hitps://blog.csdn.net/veizin_435b	.				

第六题 disabled_button

```
这一题做的时候很简单,就把disabled删了,按钮一按,flag就出来了。
深究disabled的原理:
在表单的提交中,
disabled:对于所有的表单元素都有效,包括select,radio,checkbox,button等。如果一个输入项的disabled设为true,则该表单
输入项不能获取焦点,用户的所有操作(鼠标点击和键盘输入等)对该输入项都无效,最重要的一点是当提交表单时,这个表单
输入项将不会被提交。
```

通常会放在一起比较的还有readonly,但这个只针对input里面的textarea和(text/password)

第七题 simple_js

这个题,其实一眼看过去就可以感觉到flag的位置在哪,关键是怎么把它转化成flag的格式。先看源代码:

```
<html>
<head>
                 <title>JS</title>
                {crite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss/rite/ss
                                                                 = pass_enc.split(',');
var tab2 = pass.split(',');var i,j,k,l=0,m,n,o,p = "";i = 0;j = tab.length;
    k = j + (1) + (n=0);
    n = tab2.length;
    for(i = (o=0); i < (k = j = n); i++ ){o = tab[i-1];p += String.fromCharCode((o = tab2[i]));</pre>
                                                                                                                                  if(i == 5)break;}
                                                                                                    for(i = (o=0); i < (k = j = n); i++ ){</pre>
                                                                                                    o = tab[i-1];
if(i > 5 && i < k-1)</pre>
                                                                                                                                                                        p += String.fromCharCode((o = tab2[i]));
                                 p += String.fromCharCode(tab2[17]);
                                 pass = p;return pass;
                  ,
String["fromCharCode"]
h = window.prompt('Enter password');
alert( dechiffre(h) );
</script>
</head
</html>
```

发现关键问题在pass的值和dechiffre函数中那一串16进制数是否吻合,(当然,题目给出密码输不对,那一定是不吻合的),这个地方就是bug,把那一串16进制数转为ASCII码在根据题目的flag格式,就可以得到flag。

第八题 xff_referer

这个题目看到很多writeup都用了burp suite来操作,只可惜我这个菜鸟级选手总是没办法实现他们的效果,于是便用了另一种方法:

XFF和Referer都是HTTP的首部,XFF是X-Forwarded-For的缩写,是用于描述客户端的IP地址,Referer是描述用户是从这个页面上依照链接跳转过来的。两个都是请求类型的首部。

知道了这个,用Hackbar就可以实现了

	🕞 🚹 🛛 Elemente	s Console So	urces Network	Performance M	emory Applic	ation Secu	irity Audits	HackBar	: ×
	LOAD URL	SPLIT URL	EXECUTE URI	L SQLI -	XSS -	LFI 🕶	SSTI -	ENCOD	ING -
	URL http://111.198.2	29.45:46932/							
cyberpeace{2674f6e5e1be7a67eb9d75dec6cff9a4}	Enable P	OST		A	DD HEADER				
					Name Referer	•	Value https://ww	w.google	×
					Name X-Forwarde	d-For 🔻	Value 123.123.12	23.123	×
	Console								43908

第九题 weak_auth

弱密码问题,弱密码这个应该算是常识,纯数字就比较弱,所以很多人都是输入123456,flag就拿到了。 所以我也没有具体的去尝试其他方法,不过看到另一种思路就是使用bp弄一个字典来试密码,这个时间关系还没有具体实践 过......

不过就是有一个答题经验弱密码可以用一些常见密码试试。

第十题 webshell

用中国菜刀试一试好像就可以了,但是我用Linux就没有中国菜刀,CKnife也没下,这题就没做。

第十一题 command_execution

这题用的是Linux中的一些命令,所以需要学一些Linux基础。

这里放一个讲解十个常用的运算符的文章

运算符英文教程

'&'是让命令在后台进行,'l'管道和python里的含义是类似的,前一个的输出流是后一个输入流,'ls'十分重要的命令,细分了很多 命令。

用管道命令和ls-a找目录,一个一个试发现了flag文件在home目录下,当然也可以一开始先试几个比较特别的文件,比如 home,因为在Linux系统中是主目录。(题外话:为什么这题会需要用Linux命令呢?因为题目环境是设置在Ubuntu中的)

PING

请输入需要ping的地址

PING

```
ping -c 3 127.0.0.1 | ls /home -a
.
.
flag.txt
https://bloc.csdn.net/weixin_439088
```

最后用cat命令+文件路径读取文件,得到flag

PING

请输入需要ping的地址

PING

ping -c 3 127.0.0.1 & cat /home/flag.txt
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.056 ms
cyberpeace{ff825e79a89f6017c13cc4fe2751e126}64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.042 ms

--- 127.0.0.1 ping statistics ---3 packets transmitted, 3 received, 0% packet loss, time 1998ms rtt min/avg/max/mdev = 0.042/0.048/0.056/0.005 ms

https://blog.csdn.net/weixin_43908872

第十二题 simple_php

```
<?php
show_source(__FILE__);
include("config.php");
$a=@$_GET['a'];
$b=@$_GET['b'];
if($a==0 and $a){
    echo $flag1;
}
if(is_numeric($b)){
    exit();
}
if($b>1234){
    echo $flag2;
}
?>ttps://blog.csdn.net/weixin_43908872
```

由代码知道要求a==0且a不为空,那当a为一个字符串就可以了

PHP中'0'==0的bool值是true,可以令a='0',b的要求是不能是数字且要大于1234,所以随便令b等于一个大于1234的字符串即可

php<br show_source(FILE); include("config.php"); \$a=@\$_GET['a'];	LOAD URL	s Console So	EXECUTE URI	Performance	Memory A	pplication	Security	Audits	HackBar ENCOD	i ×
<pre>\$D=#05_GE1['0']; if(\$a==0 and \$a){ echo \$flag1; } if(is_numeric(\$b)){ evit();</pre>	URL http://111.198.2	29.45:30566/?a	="0"&b= <u>1235i</u>							
<pre>} } if(\$b>1234){ echo \$flag2; } </pre>	Enable P	OST			ADD HEAI	DER				
Cyberpeace {647E37C7627CC3E4019EC69324F66C7C}										