①

-O0

c: %ebp-0x4 - push %ebx - valor armazenado em (ebp / 0xc) - movl 0x0, 0xc
result: %ebp - 0x9c - movl $0x0, -0x9c(%ebp)
n: %ebp + 0x8 - vem como argumento, visual com (divl 0x8(%ebp))

-O2

c: %ebp-0xc - push %ebx, xor %ebx,%ebx - c=0
result: %ebp-0x8 - push %esi, xor %esi,%esi - result=0
n: %ebp+0x8 - movl 0x8(%ebp),%edi) - edi=n

②

0x08048 40F: cmpl $0x1F, -0xc(%ebp)
0x08 04841 3: jle 0x8048417
0x08048415: jmp 0x8048438
:
0x... : lea -0xc(%ebp), %eax
0x... : incl (%eax)

// 0x1F = 32 = TAM, -0xc = valor de c, logo = c<TAM
// caso a condição se verifique entra no ciclo
// caso contrário sai do ciclo
...

// le o valor de ebp-0xc e aponta eax para ele (valor de c)
// incrementa o valor do endereço de eax (c++)

③ Esta funcionalidade permite obter os 6 valores presentes nos endereços a partir de ebp-12, com isto sabemos que:
ebx - 0x007d3ff4
esi - 0x0057 3ca0
edi - 0x00000000
main - 0xbffffeas8
retorno - 0x080484a3
arg n - 0x00000014

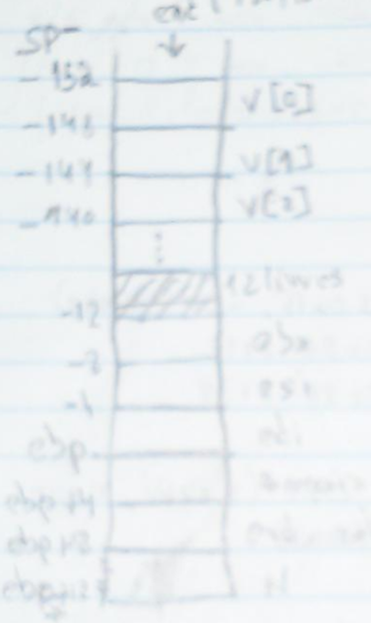
(4)

(ii) após breakpoint
esp (Reajustar) , logo , [illegible] antes [illegible]
xor %ebx , %eax)

(iii) call (5 bytes) que invocou calcula(n)
end retorno − 5 = 0x080481a3 − 5
= 0x0804849e

(5)

end ([illegible] x/35xw)



v[] = 32×4
= 128
mov 0x98(ebp)
= 152
− 12
= 140

(6)

(i) 140 células reservadas
(ii) v[2] = 0x00000007 = 7 //

(7) TAM = 4
reservados 4×4 = 16, logo vamos apenas assumir
4 valores para obter o resultado, v[0] + v[1] + v[2] + v[3]
= 4 + 7 + 12 + 10 = 33 //

(8) 0x804843F + 2 = 0x8048441

0x8048534 − 0x8048441 = F3 //