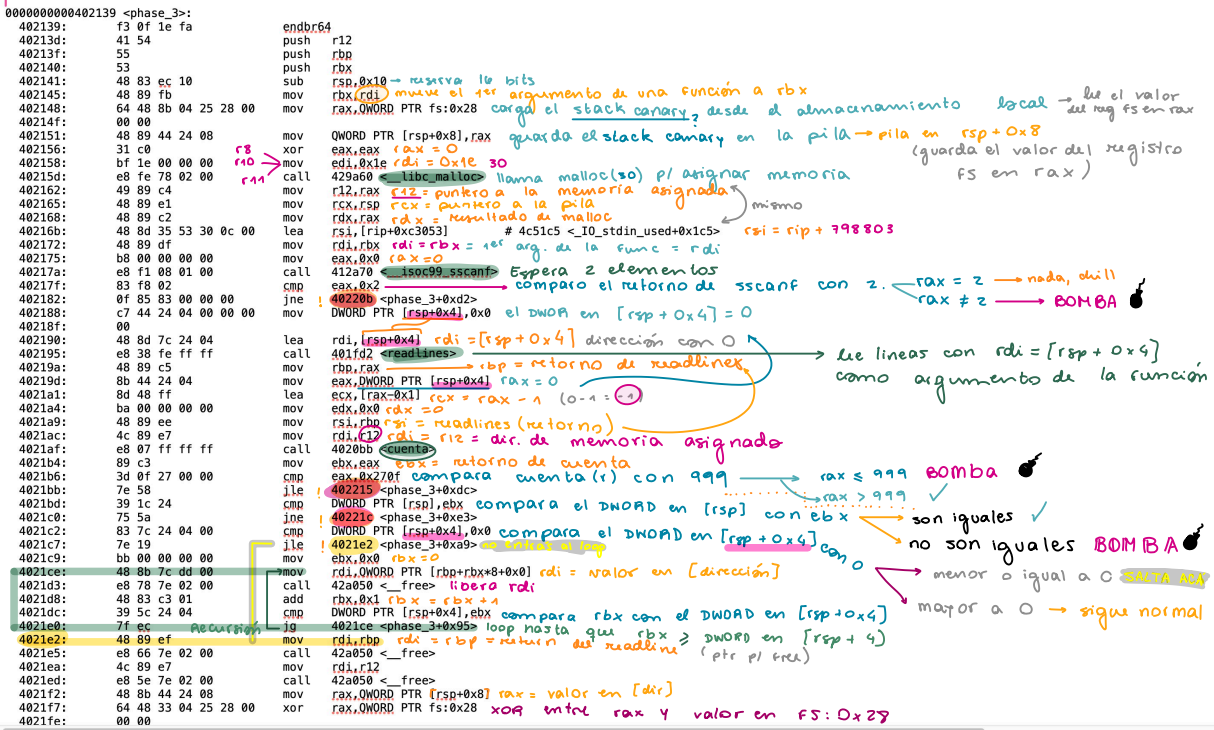
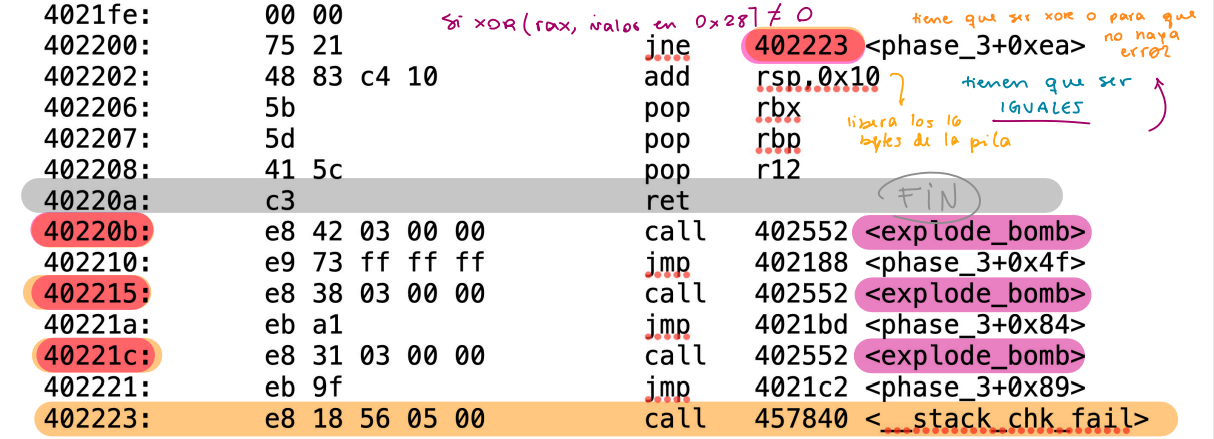
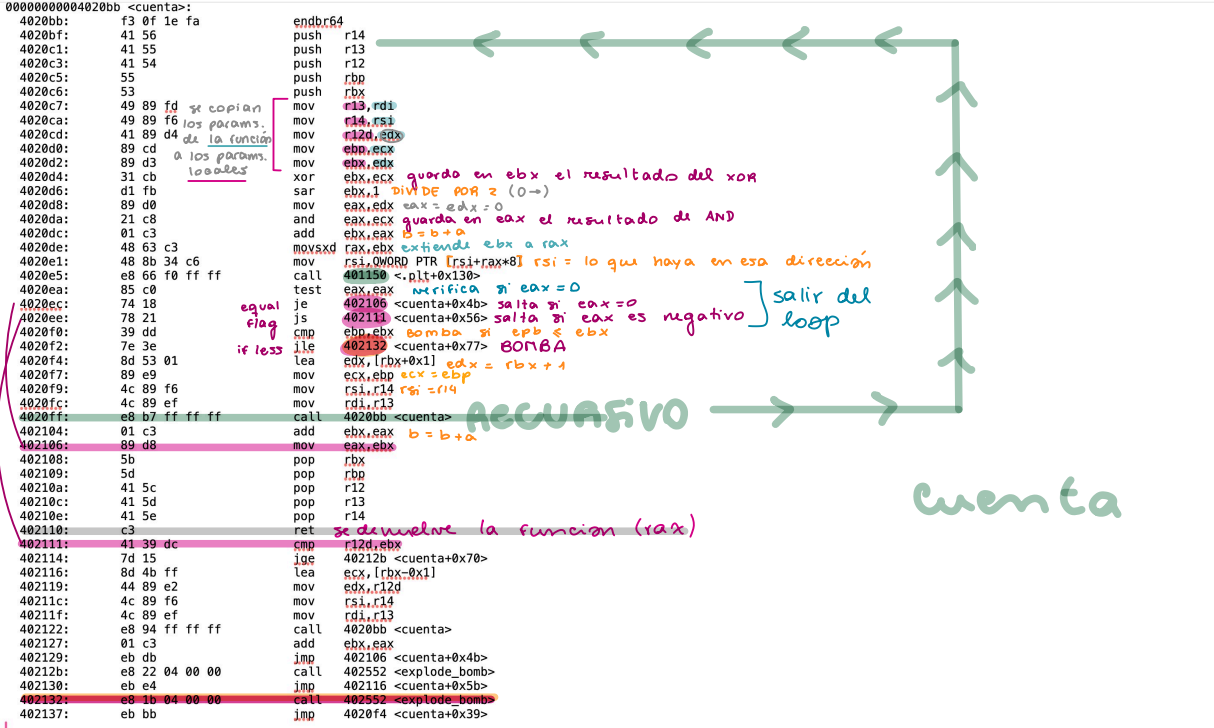

0000000000402139 <phase_3>:
402139: f3 0f 1e fa endbr64
4013d: 41 54 push r12
4013f: 55 push rbp
402140: 53 push rbx
402141: 48 83 ec 10 sub rsp,0x10 → reserva 16 bits
402145: 48 89 fb mov rbx,rdi mueve el 1er argumento de una función a rbx
402148: 64 48 8b 04 25 28 00 mov rax,QWORD PTR fs:0x28 carga el stack canary desde el almacenamiento local → lee el valor del reg fs en rax
40214f: 00 00
402151: 48 89 44 24 08 mov QWORD PTR [rsp+0x8],rax guarda el stack canary en la pila → pila en rsp + 0x8 (guarda el valor del registro fs en rax)
402156: 31 c0 xor eax,eax rax = 0
r8 r10 r11
402158: bf 1e 00 00 00 mov edi,0x1e rdi = 0x1e 30
40215d: e8 fe 78 02 00 call 429a60 <__libc_malloc> llama malloc(30) p/ asignar memoria
402162: 49 89 c4 mov r12,rax r12 = puntero a la memoria asignada
402165: 48 89 e1 mov rcx,rsp rcx = puntero a la pila
402168: 48 89 c2 mov rdx,rax rdx = resultado de malloc
mismo
40216b: 48 8d 35 53 30 0c 00 lea rsi,[rip+0xc3053] # 4c51c5 <_IO_stdin_used+0x1c5> rsi = rip + 798803
402172: 48 89 df mov rdi,rbx rdi = rbx = 1er arg. de la func = rdi
402175: b8 00 00 00 00 mov eax,0x0 rax = 0
40217a: e8 f1 08 01 00 call 412a70 <__isoc99_sscanf> Espera 2 elementos
40217f: 83 f8 02 cmp eax,0x2 comparo el retorno de sscanf con 2.
rax = 2 → nada, chill
rax ≠ 2 → BOMBA
402182: 0f 85 83 00 00 00 jne ! 40220b <phase_3+0xd2>
402188: c7 44 24 04 00 00 00 00 mov DWORD PTR [rsp+0x4],0x0 el DWOR en [rsp + 0x4] = 0
40218f: 00
402190: 48 8d 7c 24 04 lea rdi,[rsp+0x4] rdi = [rsp + 0x4] dirección con 0
402195: e8 38 fe ff ff call 401fd2 <readlines>
Lee lineas con rdi = [rsp + 0x4] como argumento de la función
40219a: 48 89 c5 mov rbp,rax rbp = retorno de readlines
40219d: 8b 44 24 04 mov eax,DWORD PTR [rsp+0x4] rax = 0
4021a1: 8d 48 ff lea ecx,[rax-0x1] rcx = rax - 1 (0 - 1 = -1)
4021a4: ba 00 00 00 00 mov edx,0x0 rdx = 0
4021a9: 48 89 ee mov rsi,rbp rsi = readlines (retorno)
4021ac: 4c 89 e7 mov rdi,r12 rdi = r12 = dir. de memoria asignada
4021af: e8 07 ff ff ff call 4020bb <cuenta>
4021b4: 89 c3 mov ebx,eax ebx = retorno de cuenta
4021b6: 3d 0f 27 00 00 cmp eax,0x270f compara cuenta(r) con 999
rax ≤ 999 BOMBA
4021bb: 7e 58 jle ! 402215 <phase_3+0xdc>
rax > 999
4021bd: 39 1c 24 cmp DWORD PTR [rsp],ebx compara el DWORD en [rsp] con ebx
son iguales
4021c0: 75 5a jne ! 40221c <phase_3+0xe3>
no son iguales BOMBA
4021c2: 83 7c 24 04 00 cmp DWORD PTR [rsp+0x4],0x0 compara el DWORD en [rsp + 0x4] con 0
4021c7: 7e 19 jle ! 4021e2 <phase_3+0xa9>
menor o igual a 0 SALTA ACA
4021c9: bb 00 00 00 00 mov ebx,0x0 ebx = 0
4021ce: 48 8b 7c dd 00 mov rdi,QWORD PTR [rbp+rbx+8+0x0] rdi = valor en [dirección]
4021d3: e8 78 7e 02 00 call 42a050 <__free> libera rdi
4021d8: 48 83 c3 01 add rbx,0x1 rbx = rbx + 1
4021dc: 39 5c 24 04 cmp DWORD PTR [rsp+0x4],ebx compara rbx con el DWORD en [rsp+0x4]
4021e0: 7f ec jg 4021ce <phase_3+0x95> loop hasta que rbx ≥ DWORD en [rsp + 4]
recursión
mayor a 0 → sigue normal
(ptr p/ free)
4021e2: 48 89 ef mov rdi,rbp rdi = rbp = return del readline
4021e5: e8 66 7e 02 00 call 42a050 <__free>
4021ea: 4c 89 e7 mov rdi,r12
4021ed: e8 5e 7e 02 00 call 42a050 <__free>
4021f2: 48 8b 44 24 08 mov rax,QWORD PTR [rsp+0x8] rax = valor en [dir]
4021f7: 64 48 33 04 25 28 00 xor rax,QWORD PTR fs:0x28 xor entre rax y valor en FS: 0x28
4021fe: 00 00


si xor(rax, valor en 0x28) ≠ 0
tiene que ser xor o para que no haya error
tienen que ser IGUALES
libera los 16 bytes de la pila
FIN
4021fe: 00 00
402200: 75 21 jne 402223 <phase_3+0xea>
402202: 48 83 c4 10 add rsp,0x10
402206: 5b pop rbx
402207: 5d pop rbp
402208: 41 5c pop r12
40220a: c3 ret
40220b: e8 42 03 00 00 call 402552 <explode_bomb>
402210: e9 73 ff ff ff jmp 402188 <phase_3+0x4f>
402215: e8 38 03 00 00 call 402552 <explode_bomb>
40221a: eb a1 jmp 4021bd <phase_3+0x84>
40221c: e8 31 03 00 00 call 402552 <explode_bomb>
402221: eb 9f jmp 4021c2 <phase_3+0x89>
402223: e8 18 56 05 00 call 457840 <__stack_chk_fail>
0000000000402228 <phase_4>:


0000000000004020bb <cuenta>:
4020bb: f3 0f 1e fa endbr64
4020bf: 41 56 push r14
4020c1: 41 55 push r13
4020c3: 41 54 push r12
4020c5: 55 push rbp
4020c6: 53 push rbx
4020c7: 49 89 fd se copian
4020ca: 49 89 f6 los params.
4020cd: 41 89 d4 de la función
4020d0: 89 cd a los params.
4020d2: 89 d3 locales
4020d4: 31 cb xor ebx,ecx guarda en ebx el resultado del xor
4020d6: d1 fb sar ebx,1 DIVIDE POR 2 (0→)
4020d8: 89 d0 mov eax,edx eax = edx = 0
4020da: 21 c8 and eax,ecx guarda en eax el resultado de AND
4020dc: 01 c3 add ebx,eax b = b + a
4020de: 48 63 c3 movsxd rax,ebx extiende ebx a rax
4020e1: 48 8b 34 c6 mov rsi,QWORD PTR [rsi+rax*8] rsi = lo que haya en esa dirección
4020e5: e8 66 f0 ff ff call 401150 <.plt+0x130>
4020ea: 85 c0 test eax,eax verifica si eax = 0
4020ec: 74 18 je 402106 <cuenta+0x4b> salta si eax=0
4020ee: 78 21 js 402111 <cuenta+0x56> salta si eax es negativo
4020f0: 39 dd cmp ebp,ebx BOMBA si ebp < ebx
4020f2: 7e 3e jle 402132 <cuenta+0x77> BOMBA
4020f4: 8d 53 01 lea edx,[rbx+0x1] edx = rbx + 1
4020f7: 89 e9 mov ecx,ebp ecx = ebp
4020f9: 4c 89 f6 mov rsi,r14 rsi = r14
4020fc: 4c 89 ef mov rdi,r13
4020ff: e8 b7 ff ff ff call 4020bb <cuenta>
402104: 01 c3 add ebx,eax b = b + a
402106: 89 d8 mov eax,ebx
402108: 5b pop rbx
402109: 5d pop rbp
40210a: 41 5c pop r12
40210c: 41 5d pop r13
40210e: 41 5e pop r14
402110: c3 ret se devuelve la función (rax)
402111: 41 39 dc cmp r12d,ebx
402114: 7d 15 jge 40212b <cuenta+0x70>
402116: 8d 4b ff lea ecx,[rbx-0x1]
402119: 44 89 e2 mov edx,r12d
40211c: 4c 89 f6 mov rsi,r14
40211f: 4c 89 ef mov rdi,r13
402122: e8 94 ff ff ff call 4020bb <cuenta>
402127: 01 c3 add ebx,eax
402129: eb db jmp 402106 <cuenta+0x4b>
40212b: e8 22 04 00 00 call 402552 <explode_bomb>
402130: eb e4 jmp 402116 <cuenta+0x5b>
402132: e8 1b 04 00 00 call 402552 <explode_bomb>
402137: eb bb jmp 4020f4 <cuenta+0x39>
equal flag
if less
salir del loop
RECURSIVO
cuenta

Qué sabemos:

★ Son 2 elementos

son 13 y 44

EXPLOTA

① si no son 2 argumentos → solución: ingresar 2 elementos ✓

cuenta(r) < 999 → solución: buscar número que cuenta(n) > 999 ②

DWORD en [rsp+4] ≠ ebx → solución: DWORD en [rsp+4] = ebx ③

(a)

Rax xor (valor en fs:0x28) ≠ 0 → solución: Rax xor (valor en fs:0x28) = 0 ④

(cuenta)

rbp < ebx → solución: rbp > ebx ⑤