Local Functions, Their Full Disassembly, and Strings from Object Files:

File: Project1.o

Project1.o: file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <main>:

0: 55 push %rbp

1: 48 89 e5 mov %rsp,%rbp

4: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # b <main+0xb>

b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 12 <main+0x12>

12: e8 00 00 00 00 call 17 <main+0x17>

17: e8 00 00 00 00 call 1c <main+0x1c>

1c: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 22 <main+0x22>

22: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 28 <main+0x28>

28: 83 f8 01 cmp $0x1,%eax

2b: 74 13 je 40 <main+0x40>

2d: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 34 <main+0x34>

34: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 3b <main+0x3b>

3b: e8 00 00 00 00 call 40 <main+0x40>

40: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 46 <main+0x46>

46: 83 f8 01 cmp $0x1,%eax

49: 74 02 je 4d <main+0x4d>

4b: eb ca jmp 17 <main+0x17>

4d: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 54 <main+0x54>

54: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 5b <main+0x5b>

5b: e8 00 00 00 00 call 60 <main+0x60>

60: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 67 <main+0x67>

67: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 6e <main+0x6e>

6e: e8 00 00 00 00 call 73 <main+0x73>

73: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 7a <main+0x7a>

7a: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 81 <main+0x81>

81: e8 00 00 00 00 call 86 <main+0x86>

86: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 8d <main+0x8d>

8d: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 94 <main+0x94>

94: e8 00 00 00 00 call 99 <main+0x99>

99: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # a0 <main+0xa0>

a0: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # a7 <main+0xa7>

a7: e8 00 00 00 00 call ac <main+0xac>

ac: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # b3 <main+0xb3>

b3: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # ba <main+0xba>

ba: e8 00 00 00 00 call bf <main+0xbf>

bf: 48 89 c2 mov %rax,%rdx

c2: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # c8 <main+0xc8>

c8: 89 c6 mov %eax,%esi

ca: 48 89 d7 mov %rdx,%rdi

cd: e8 00 00 00 00 call d2 <main+0xd2>

d2: 48 89 c2 mov %rax,%rdx

d5: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # dc <main+0xdc>

dc: 48 89 c6 mov %rax,%rsi

df: 48 89 d7 mov %rdx,%rdi

e2: e8 00 00 00 00 call e7 <main+0xe7>

e7: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # ed <main+0xed>

ed: 83 f8 01 cmp $0x1,%eax

f0: 75 07 jne f9 <main+0xf9>

f2: e8 00 00 00 00 call f7 <main+0xf7>

f7: eb 10 jmp 109 <main+0x109>

f9: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # ff <main+0xff>

ff: 83 f8 02 cmp $0x2,%eax

102: 75 05 jne 109 <main+0x109>

104: e8 00 00 00 00 call 109 <main+0x109>

109: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 10f <main+0x10f>

10f: 83 f8 03 cmp $0x3,%eax

112: 74 05 je 119 <main+0x119>

114: e9 34 ff ff ff jmp 4d <main+0x4d>

119: b8 00 00 00 00 mov $0x0,%eax

11e: 5d pop %rbp

11f: c3 ret

0000000000000120 <\_Z25CheckUserPermissionAccessv>:

120: 55 push %rbp

121: 48 89 e5 mov %rsp,%rbp

124: 53 push %rbx

125: 48 83 ec 48 sub $0x48,%rsp

129: 64 48 8b 04 25 28 00 mov %fs:0x28,%rax

130: 00 00

132: 48 89 45 e8 mov %rax,-0x18(%rbp)

136: 31 c0 xor %eax,%eax

138: 48 8d 45 bb lea -0x45(%rbp),%rax

13c: 48 89 c7 mov %rax,%rdi

13f: e8 00 00 00 00 call 144 <\_Z25CheckUserPermissionAccessv+0x24>

144: 48 8d 55 bb lea -0x45(%rbp),%rdx

148: 48 8d 45 c0 lea -0x40(%rbp),%rax

14c: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 153 <\_Z25CheckUserPermissionAccessv+0x33>

153: 48 89 c7 mov %rax,%rdi

156: e8 00 00 00 00 call 15b <\_Z25CheckUserPermissionAccessv+0x3b>

15b: 48 8d 45 bb lea -0x45(%rbp),%rax

15f: 48 89 c7 mov %rax,%rdi

162: e8 00 00 00 00 call 167 <\_Z25CheckUserPermissionAccessv+0x47>

167: c7 45 bc 00 00 00 00 movl $0x0,-0x44(%rbp)

16e: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 175 <\_Z25CheckUserPermissionAccessv+0x55>

175: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 17c <\_Z25CheckUserPermissionAccessv+0x5c>

17c: e8 00 00 00 00 call 181 <\_Z25CheckUserPermissionAccessv+0x61>

181: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 188 <\_Z25CheckUserPermissionAccessv+0x68>

188: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 18f <\_Z25CheckUserPermissionAccessv+0x6f>

18f: e8 00 00 00 00 call 194 <\_Z25CheckUserPermissionAccessv+0x74>

194: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 19b <\_Z25CheckUserPermissionAccessv+0x7b>

19b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 1a2 <\_Z25CheckUserPermissionAccessv+0x82>

1a2: e8 00 00 00 00 call 1a7 <\_Z25CheckUserPermissionAccessv+0x87>

1a7: 48 8d 45 c0 lea -0x40(%rbp),%rax

1ab: 48 89 c6 mov %rax,%rsi

1ae: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 1b5 <\_Z25CheckUserPermissionAccessv+0x95>

1b5: e8 00 00 00 00 call 1ba <\_Z25CheckUserPermissionAccessv+0x9a>

1ba: 48 8d 45 c0 lea -0x40(%rbp),%rax

1be: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 1c5 <\_Z25CheckUserPermissionAccessv+0xa5>

1c5: 48 89 c7 mov %rax,%rdi

1c8: e8 00 00 00 00 call 1cd <\_Z25CheckUserPermissionAccessv+0xad>

1cd: 89 45 bc mov %eax,-0x44(%rbp)

1d0: 83 7d bc 00 cmpl $0x0,-0x44(%rbp)

1d4: 75 07 jne 1dd <\_Z25CheckUserPermissionAccessv+0xbd>

1d6: bb 01 00 00 00 mov $0x1,%ebx

1db: eb 05 jmp 1e2 <\_Z25CheckUserPermissionAccessv+0xc2>

1dd: bb 02 00 00 00 mov $0x2,%ebx

1e2: 48 8d 45 c0 lea -0x40(%rbp),%rax

1e6: 48 89 c7 mov %rax,%rdi

1e9: e8 00 00 00 00 call 1ee <\_Z25CheckUserPermissionAccessv+0xce>

1ee: 89 d8 mov %ebx,%eax

1f0: 48 8b 4d e8 mov -0x18(%rbp),%rcx

1f4: 64 48 33 0c 25 28 00 xor %fs:0x28,%rcx

1fb: 00 00

1fd: 74 3b je 23a <\_Z25CheckUserPermissionAccessv+0x11a>

1ff: eb 34 jmp 235 <\_Z25CheckUserPermissionAccessv+0x115>

201: 48 89 c3 mov %rax,%rbx

204: 48 8d 45 bb lea -0x45(%rbp),%rax

208: 48 89 c7 mov %rax,%rdi

20b: e8 00 00 00 00 call 210 <\_Z25CheckUserPermissionAccessv+0xf0>

210: 48 89 d8 mov %rbx,%rax

213: 48 89 c7 mov %rax,%rdi

216: e8 00 00 00 00 call 21b <\_Z25CheckUserPermissionAccessv+0xfb>

21b: 48 89 c3 mov %rax,%rbx

21e: 48 8d 45 c0 lea -0x40(%rbp),%rax

222: 48 89 c7 mov %rax,%rdi

225: e8 00 00 00 00 call 22a <\_Z25CheckUserPermissionAccessv+0x10a>

22a: 48 89 d8 mov %rbx,%rax

22d: 48 89 c7 mov %rax,%rdi

230: e8 00 00 00 00 call 235 <\_Z25CheckUserPermissionAccessv+0x115>

235: e8 00 00 00 00 call 23a <\_Z25CheckUserPermissionAccessv+0x11a>

23a: 48 83 c4 48 add $0x48,%rsp

23e: 5b pop %rbx

23f: 5d pop %rbp

240: c3 ret

0000000000000241 <\_Z11DisplayInfov>:

241: 55 push %rbp

242: 48 89 e5 mov %rsp,%rbp

245: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 24c <\_Z11DisplayInfov+0xb>

24c: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 253 <\_Z11DisplayInfov+0x12>

253: e8 00 00 00 00 call 258 <\_Z11DisplayInfov+0x17>

258: 48 89 c2 mov %rax,%rdx

25b: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 262 <\_Z11DisplayInfov+0x21>

262: 48 89 c6 mov %rax,%rsi

265: 48 89 d7 mov %rdx,%rdi

268: e8 00 00 00 00 call 26d <\_Z11DisplayInfov+0x2c>

26d: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 274 <\_Z11DisplayInfov+0x33>

274: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 27b <\_Z11DisplayInfov+0x3a>

27b: e8 00 00 00 00 call 280 <\_Z11DisplayInfov+0x3f>

280: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 287 <\_Z11DisplayInfov+0x46>

287: 48 89 c7 mov %rax,%rdi

28a: e8 00 00 00 00 call 28f <\_Z11DisplayInfov+0x4e>

28f: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 296 <\_Z11DisplayInfov+0x55>

296: 48 89 c7 mov %rax,%rdi

299: e8 00 00 00 00 call 29e <\_Z11DisplayInfov+0x5d>

29e: 48 89 c2 mov %rax,%rdx

2a1: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 2a7 <\_Z11DisplayInfov+0x66>

2a7: 89 c6 mov %eax,%esi

2a9: 48 89 d7 mov %rdx,%rdi

2ac: e8 00 00 00 00 call 2b1 <\_Z11DisplayInfov+0x70>

2b1: 48 89 c2 mov %rax,%rdx

2b4: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 2bb <\_Z11DisplayInfov+0x7a>

2bb: 48 89 c6 mov %rax,%rsi

2be: 48 89 d7 mov %rdx,%rdi

2c1: e8 00 00 00 00 call 2c6 <\_Z11DisplayInfov+0x85>

2c6: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2cd <\_Z11DisplayInfov+0x8c>

2cd: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 2d4 <\_Z11DisplayInfov+0x93>

2d4: e8 00 00 00 00 call 2d9 <\_Z11DisplayInfov+0x98>

2d9: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2e0 <\_Z11DisplayInfov+0x9f>

2e0: 48 89 c7 mov %rax,%rdi

2e3: e8 00 00 00 00 call 2e8 <\_Z11DisplayInfov+0xa7>

2e8: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 2ef <\_Z11DisplayInfov+0xae>

2ef: 48 89 c7 mov %rax,%rdi

2f2: e8 00 00 00 00 call 2f7 <\_Z11DisplayInfov+0xb6>

2f7: 48 89 c2 mov %rax,%rdx

2fa: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 300 <\_Z11DisplayInfov+0xbf>

300: 89 c6 mov %eax,%esi

302: 48 89 d7 mov %rdx,%rdi

305: e8 00 00 00 00 call 30a <\_Z11DisplayInfov+0xc9>

30a: 48 89 c2 mov %rax,%rdx

30d: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 314 <\_Z11DisplayInfov+0xd3>

314: 48 89 c6 mov %rax,%rsi

317: 48 89 d7 mov %rdx,%rdi

31a: e8 00 00 00 00 call 31f <\_Z11DisplayInfov+0xde>

31f: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 326 <\_Z11DisplayInfov+0xe5>

326: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 32d <\_Z11DisplayInfov+0xec>

32d: e8 00 00 00 00 call 332 <\_Z11DisplayInfov+0xf1>

332: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 339 <\_Z11DisplayInfov+0xf8>

339: 48 89 c7 mov %rax,%rdi

33c: e8 00 00 00 00 call 341 <\_Z11DisplayInfov+0x100>

341: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 348 <\_Z11DisplayInfov+0x107>

348: 48 89 c7 mov %rax,%rdi

34b: e8 00 00 00 00 call 350 <\_Z11DisplayInfov+0x10f>

350: 48 89 c2 mov %rax,%rdx

353: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 359 <\_Z11DisplayInfov+0x118>

359: 89 c6 mov %eax,%esi

35b: 48 89 d7 mov %rdx,%rdi

35e: e8 00 00 00 00 call 363 <\_Z11DisplayInfov+0x122>

363: 48 89 c2 mov %rax,%rdx

366: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 36d <\_Z11DisplayInfov+0x12c>

36d: 48 89 c6 mov %rax,%rsi

370: 48 89 d7 mov %rdx,%rdi

373: e8 00 00 00 00 call 378 <\_Z11DisplayInfov+0x137>

378: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 37f <\_Z11DisplayInfov+0x13e>

37f: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 386 <\_Z11DisplayInfov+0x145>

386: e8 00 00 00 00 call 38b <\_Z11DisplayInfov+0x14a>

38b: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 392 <\_Z11DisplayInfov+0x151>

392: 48 89 c7 mov %rax,%rdi

395: e8 00 00 00 00 call 39a <\_Z11DisplayInfov+0x159>

39a: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3a1 <\_Z11DisplayInfov+0x160>

3a1: 48 89 c7 mov %rax,%rdi

3a4: e8 00 00 00 00 call 3a9 <\_Z11DisplayInfov+0x168>

3a9: 48 89 c2 mov %rax,%rdx

3ac: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 3b2 <\_Z11DisplayInfov+0x171>

3b2: 89 c6 mov %eax,%esi

3b4: 48 89 d7 mov %rdx,%rdi

3b7: e8 00 00 00 00 call 3bc <\_Z11DisplayInfov+0x17b>

3bc: 48 89 c2 mov %rax,%rdx

3bf: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 3c6 <\_Z11DisplayInfov+0x185>

3c6: 48 89 c6 mov %rax,%rsi

3c9: 48 89 d7 mov %rdx,%rdi

3cc: e8 00 00 00 00 call 3d1 <\_Z11DisplayInfov+0x190>

3d1: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3d8 <\_Z11DisplayInfov+0x197>

3d8: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 3df <\_Z11DisplayInfov+0x19e>

3df: e8 00 00 00 00 call 3e4 <\_Z11DisplayInfov+0x1a3>

3e4: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3eb <\_Z11DisplayInfov+0x1aa>

3eb: 48 89 c7 mov %rax,%rdi

3ee: e8 00 00 00 00 call 3f3 <\_Z11DisplayInfov+0x1b2>

3f3: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 3fa <\_Z11DisplayInfov+0x1b9>

3fa: 48 89 c7 mov %rax,%rdi

3fd: e8 00 00 00 00 call 402 <\_Z11DisplayInfov+0x1c1>

402: 48 89 c2 mov %rax,%rdx

405: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 40b <\_Z11DisplayInfov+0x1ca>

40b: 89 c6 mov %eax,%esi

40d: 48 89 d7 mov %rdx,%rdi

410: e8 00 00 00 00 call 415 <\_Z11DisplayInfov+0x1d4>

415: 48 89 c2 mov %rax,%rdx

418: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 41f <\_Z11DisplayInfov+0x1de>

41f: 48 89 c6 mov %rax,%rsi

422: 48 89 d7 mov %rdx,%rdi

425: e8 00 00 00 00 call 42a <\_Z11DisplayInfov+0x1e9>

42a: 90 nop

42b: 5d pop %rbp

42c: c3 ret

000000000000042d <\_Z20ChangeCustomerChoicev>:

42d: 55 push %rbp

42e: 48 89 e5 mov %rsp,%rbp

431: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 438 <\_Z20ChangeCustomerChoicev+0xb>

438: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 43f <\_Z20ChangeCustomerChoicev+0x12>

43f: e8 00 00 00 00 call 444 <\_Z20ChangeCustomerChoicev+0x17>

444: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 44b <\_Z20ChangeCustomerChoicev+0x1e>

44b: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 452 <\_Z20ChangeCustomerChoicev+0x25>

452: e8 00 00 00 00 call 457 <\_Z20ChangeCustomerChoicev+0x2a>

457: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 45e <\_Z20ChangeCustomerChoicev+0x31>

45e: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 465 <\_Z20ChangeCustomerChoicev+0x38>

465: e8 00 00 00 00 call 46a <\_Z20ChangeCustomerChoicev+0x3d>

46a: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 471 <\_Z20ChangeCustomerChoicev+0x44>

471: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 478 <\_Z20ChangeCustomerChoicev+0x4b>

478: e8 00 00 00 00 call 47d <\_Z20ChangeCustomerChoicev+0x50>

47d: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 483 <\_Z20ChangeCustomerChoicev+0x56>

483: 83 f8 01 cmp $0x1,%eax

486: 75 0e jne 496 <\_Z20ChangeCustomerChoicev+0x69>

488: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 48e <\_Z20ChangeCustomerChoicev+0x61>

48e: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 494 <\_Z20ChangeCustomerChoicev+0x67>

494: eb 62 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

496: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 49c <\_Z20ChangeCustomerChoicev+0x6f>

49c: 83 f8 02 cmp $0x2,%eax

49f: 75 0e jne 4af <\_Z20ChangeCustomerChoicev+0x82>

4a1: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4a7 <\_Z20ChangeCustomerChoicev+0x7a>

4a7: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4ad <\_Z20ChangeCustomerChoicev+0x80>

4ad: eb 49 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4af: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4b5 <\_Z20ChangeCustomerChoicev+0x88>

4b5: 83 f8 03 cmp $0x3,%eax

4b8: 75 0e jne 4c8 <\_Z20ChangeCustomerChoicev+0x9b>

4ba: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4c0 <\_Z20ChangeCustomerChoicev+0x93>

4c0: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4c6 <\_Z20ChangeCustomerChoicev+0x99>

4c6: eb 30 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4c8: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4ce <\_Z20ChangeCustomerChoicev+0xa1>

4ce: 83 f8 04 cmp $0x4,%eax

4d1: 75 0e jne 4e1 <\_Z20ChangeCustomerChoicev+0xb4>

4d3: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4d9 <\_Z20ChangeCustomerChoicev+0xac>

4d9: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4df <\_Z20ChangeCustomerChoicev+0xb2>

4df: eb 17 jmp 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4e1: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4e7 <\_Z20ChangeCustomerChoicev+0xba>

4e7: 83 f8 05 cmp $0x5,%eax

4ea: 75 0c jne 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4ec: 8b 05 00 00 00 00 mov 0x0(%rip),%eax # 4f2 <\_Z20ChangeCustomerChoicev+0xc5>

4f2: 89 05 00 00 00 00 mov %eax,0x0(%rip) # 4f8 <\_Z20ChangeCustomerChoicev+0xcb>

4f8: 90 nop

4f9: 5d pop %rbp

4fa: c3 ret

00000000000004fb <\_Z41\_\_static\_initialization\_and\_destruction\_0ii>:

4fb: 55 push %rbp

4fc: 48 89 e5 mov %rsp,%rbp

4ff: 48 83 ec 10 sub $0x10,%rsp

503: 89 7d fc mov %edi,-0x4(%rbp)

506: 89 75 f8 mov %esi,-0x8(%rbp)

509: 83 7d fc 01 cmpl $0x1,-0x4(%rbp)

50d: 75 32 jne 541 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x46>

50f: 81 7d f8 ff ff 00 00 cmpl $0xffff,-0x8(%rbp)

516: 75 29 jne 541 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x46>

518: 48 8d 3d 00 00 00 00 lea 0x0(%rip),%rdi # 51f <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x24>

51f: e8 00 00 00 00 call 524 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x29>

524: 48 8d 15 00 00 00 00 lea 0x0(%rip),%rdx # 52b <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x30>

52b: 48 8d 35 00 00 00 00 lea 0x0(%rip),%rsi # 532 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x37>

532: 48 8b 05 00 00 00 00 mov 0x0(%rip),%rax # 539 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x3e>

539: 48 89 c7 mov %rax,%rdi

53c: e8 00 00 00 00 call 541 <\_Z41\_\_static\_initialization\_and\_destruction\_0ii+0x46>

541: 90 nop

542: c9 leave

543: c3 ret

0000000000000544 <\_GLOBAL\_\_sub\_I\_username>:

544: 55 push %rbp

545: 48 89 e5 mov %rsp,%rbp

548: be ff ff 00 00 mov $0xffff,%esi

54d: bf 01 00 00 00 mov $0x1,%edi

552: e8 a4 ff ff ff call 4fb <\_Z41\_\_static\_initialization\_and\_destruction\_0ii>

557: 5d pop %rbp

558: c3 ret

Strings and the Functions they belong to:

String 'Bob Jones' (offset 1456) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Sarah Davis' (offset 1472) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Amy Friendly' (offset 1488) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Johnny Smith' (offset 1504) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Carol Spears' (offset 1520) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Hello! Welcome to our Investment Company' (offset 1608) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Invalid Password. Please try again' (offset 1656) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'What would you like to do?' (offset 1692) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'DISPLAY the client list (enter 1)' (offset 1720) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'CHANGE a client's choice (enter 2)' (offset 1760) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Exit the program.. (enter 3)' (offset 1796) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'You chose ' (offset 1826) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Enter your username: ' (offset 1838) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Enter your password: ' (offset 1861) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Client's Name Service Selected (1 = Brokerage, 2 = Retirement)' (offset 1888) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'selected option ' (offset 1960) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Enter the number of the client that you wish to change' (offset 2000) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Please enter the client's new service choice (1 = Brokerage, 2 = Retirement)' (offset 2056) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0' (offset 2177) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'zPLR' (offset 2289) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'Project1.cpp' (offset 3921) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZStL19piecewise\_construct' (offset 3934) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZStL8\_\_ioinit' (offset 3961) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_Z41\_\_static\_initialization\_and\_destruction\_0ii' (offset 3976) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_GLOBAL\_\_sub\_I\_username' (offset 4024) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'name1' (offset 4048) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'name2' (offset 4054) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'name3' (offset 4060) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'name4' (offset 4066) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'name5' (offset 4072) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'num1' (offset 4078) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'num2' (offset 4083) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'num3' (offset 4088) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'num4' (offset 4093) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'num5' (offset 4098) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'changechoice' (offset 4103) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'newservice' (offset 4116) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'password' (offset 4127) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'answer' (offset 4136) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'main' (offset 4143) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZSt4cout' (offset 4148) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_GLOBAL\_OFFSET\_TABLE\_' (offset 4158) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc' (offset 4180) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_Z25CheckUserPermissionAccessv' (offset 4236) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZSt3cin' (offset 4267) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSirsERi' (offset 4276) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSolsEi' (offset 4287) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_' (offset 4297) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSolsEPFRSoS\_E' (offset 4356) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_Z11DisplayInfov' (offset 4373) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_Z20ChangeCustomerChoicev' (offset 4390) belongs to function \_GLOBAL\_\_sub\_I\_username

String 'DW.ref.\_\_gxx\_personality\_v0' (offset 4416) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSaIcEC1Ev' (offset 4444) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEC1EPKcRKS3\_' (offset 4457) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSaIcED1Ev' (offset 4522) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZStrsIcSt11char\_traitsIcEERSt13basic\_istreamIT\_T0\_ES6\_PS3\_' (offset 4535) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZStrsIcSt11char\_traitsIcESaIcEERSt13basic\_istreamIT\_T0\_ES7\_RNSt7\_\_cxx1112basic\_stringIS4\_S5\_T1\_EE' (offset 4595) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNKSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEE7compareEPKc' (offset 4694) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev' (offset 4761) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_Unwind\_Resume' (offset 4819) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_\_stack\_chk\_fail' (offset 4834) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSt8ios\_base4InitC1Ev' (offset 4851) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_\_dso\_handle' (offset 4875) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_ZNSt8ios\_base4InitD1Ev' (offset 4888) belongs to function \_GLOBAL\_\_sub\_I\_username

String '\_\_cxa\_atexit' (offset 4912) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.symtab' (offset 8841) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.strtab' (offset 8849) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.shstrtab' (offset 8857) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.rela.text' (offset 8867) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.data' (offset 8878) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.bss' (offset 8884) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.rodata' (offset 8889) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.gcc\_except\_table' (offset 8897) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.rela.init\_array' (offset 8915) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.rela.data.rel.local.DW.ref.\_\_gxx\_personality\_v0' (offset 8932) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.comment' (offset 8981) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.note.GNU-stack' (offset 8990) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.rela.eh\_frame' (offset 9006) belongs to function \_GLOBAL\_\_sub\_I\_username

String '.group' (offset 9021) belongs to function \_GLOBAL\_\_sub\_I\_username

The assembly code for the given C++ code exemplifies a conventional framework used for managing input, decision-making, and iteration. The main() function initiates the assembly by configuring the stack with instructions such as push %rbp and mov %rsp, %rbp. This is prevalent in assembly since it establishes the stack frame for the function's local variables and registers. The software then loads data with the lea command, which retrieves effective addresses associated with strings such as the menu prompts.  
  
In C++, the line where the program displays the menu via std::cout aligns with these lea instructions, followed by the call instruction that invokes the function tasked with rendering data to the console. Upon user input, the selection is saved in a variable and then used in comparison statements in C++. In assembly, this is shown via instructions such as cmp $0x1, %eax, where the value in the eax register (which contains the user input) is compared to the number 1. If the user picks option 1, a conditional jump (je) redirects control to the code responsible for showing the client list. This procedure is repeated for further inputs, with each comparison using distinct values such as cmp $0x2, %eax and cmp $0x3, %eax to address alternative scenarios for modifying a client’s selection or terminating the program.  
  
The ChangeCustomerChoice() method in C++ handles input in which the user picks a customer number. The assembly code similarly demonstrates this by comparing the user's input saved in registers, such as cmp $0x1, %eax for client 1, and branching to the relevant code section that outputs the related service modification. The comparisons for clients 2 through 5 adhere to a same structure, using the cmp instruction followed by jumps (jne) that guide the execution to distinct code blocks based on the client number.  
  
In CheckUserPermissionAccess(), the logic for verifying a username and password is seen in assembly via memory and register transactions. The input is first retained in memory and subsequently contrasted with predetermined values. This constitutes a streamlined authentication verification. The assembly code for this function displays many mov instructions, including movl $0x0,-0x44(%rbp), which allocates memory for the password comparison result. The function thereafter executes a sequence of call instructions, presumably denoting the functions tasked with comparing the input against stored credentials, however the intricate management of the string comparisons remains obscured behind these calls in the disassembly.  
  
The DisplayInfo() method outputs the client list in C++. This is executed in assembly using numerous lea instructions, which load the addresses of the strings denoting the customers' names and their services, followed by invocations of the output function. These actions consist of basic load and call sequences, illustrating the function's simplicity, since it produces structured data devoid of intricate logic. Each client input and service option are recorded in registers, followed by a call to output the information.

#include <iostream>

int main() {

int userChoice = 0; // Variable to store user input

do {

std::cout << "What would you like to do?" << std::endl;

std::cout << "1. DISPLAY the client list (enter 1)" << std::endl;

std::cout << "2. CHANGE a client's choice (enter 2)" << std::endl;

std::cout << "3. Exit the program (enter 3)" << std::endl;

std::cin >> userChoice;

if (userChoice == 1) {

std::cout << "Displaying the client list..." << std::endl;

// Call to the display client list function

} else if (userChoice == 2) {

std::cout << "Changing a client's choice..." << std::endl;

// Call to change client's choice function

} else if (userChoice != 3) {

std::cout << "Invalid option, please try again." << std::endl;

}

} while (userChoice != 3);

std::cout << "Exiting the program..." << std::endl;

return 0;

}

void ChangeCustomerChoice() {

int choice = 0;

std::cout << "Enter the number of the client that you wish to change: ";

std::cin >> choice;

if (choice == 1) {

std::cout << "Client 1 selected. New service: Brokerage." << std::endl;

} else if (choice == 2) {

std::cout << "Client 2 selected. New service: Retirement." << std::endl;

} else if (choice == 3) {

std::cout << "Client 3 selected. New service: Brokerage." << std::endl;

} else if (choice == 4) {

std::cout << "Client 4 selected. New service: Retirement." << std::endl;

} else if (choice == 5) {

std::cout << "Client 5 selected. New service: Brokerage." << std::endl;

} else {

std::cout << "Invalid client number." << std::endl;

}

}

bool CheckUserPermissionAccess() {

std::string username, password;

std::cout << "Enter your username: ";

std::cin >> username;

std::cout << "Enter your password: ";

std::cin >> password;

// Placeholder logic for user permission validation

if (username == "admin" && password == "password") {

std::cout << "Access granted." << std::endl;

return true;

} else {

std::cout << "Invalid username or password." << std::endl;

return false;

}

}

void DisplayInfo() {

std::cout << "Client's Name Service Selected (1 = Brokerage, 2 = Retirement)" << std::endl;

std::cout << "----------------------------------------" << std::endl;

std::cout << "Bob Jones 1" << std::endl;

std::cout << "Sarah Davis 2" << std::endl;

std::cout << "Amy Friendly 1" << std::endl;

std::cout << "Johnny Smith 2" << std::endl;

std::cout << "Carol Spears 1" << std::endl;

}

The C++ code exhibits many possible security flaws concerning user input management, authentication, and validation. This document provides an analysis of each function, including potential security vulnerabilities, methods of exploitation, and corresponding mitigations.  
  
The main function lacks input validation. The statement std::cin >> userChoice; fails to verify whether the input is a proper integer. This may lead to unforeseen behavior or program failures if a non-integer input, such as a string, is supplied. A further worry is the potential for buffer overrun. While std::cin is generally secure for fundamental inputs, the absence of length validations or input restrictions in more complex scenarios (such as when storing in strings) may result in buffer overflows. The resolution involves implementing input validation checks and sanitizing inputs prior to processing. For instance, verify if std::cin acquires a valid integer:

if (!(std::cin >> userChoice)) {

std::cerr << "Invalid input!" << std::endl;

std::cin.clear(); // Clear error flag

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // Ignore invalid input

continue;

}

The ChangeCustomerChoice function lacks input validation while receiving the user's option. Providing a non-numeric input or an unforeseen number may result in complications. Moreover, there exists the possibility of information leaking. Storing and displaying sensitive client information without authentication or authorization checks may result in data leaking. There is no verification of the user's identity or their authorization to implement modifications. The resolution involves implementing enough input validation to permit just numeric values within a certain range (e.g., 1-5), such as:

if (choice < 1 || choice > 5) {

std::cout << "Invalid client number." << std::endl;

return;

}

Additionally, implement access control verifications to confirm the user's authorization to modify client information.  
  
The CheckUserPermissionAccess method has many security vulnerabilities. The use of hardcoded credentials (admin, password) poses a considerable security threat. This facilitates attackers in deducing or uncovering the login credentials, so obtaining illegal access. The password storage is inadequate, since the code compares the password as a plaintext string. Passwords in a legitimate system should be hashed rather than saved in plaintext. Furthermore, there is an absence of rate limiting or brute force protection, indicating that there is no system in place to restrict the number of login attempts, rendering it vulnerable to brute-force assaults. Ultimately, there is a lack of input sanitization. The username and password fields lack sanitization, potentially resulting in injection attacks, such as SQL injection, particularly if the system subsequently interfaces with a database. To mitigate this issue, substitute hardcoded credentials with a secure authentication framework, use password hashing and salting techniques (e.g., bcrypt), enforce rate-limiting or account lockout after a certain number of unsuccessful login attempts, and sanitize inputs to avert injection attacks.  
  
The DisplayInfo function has a vulnerability related to the exposure of sensitive information. This function shows client names and services without access restriction or validation. An unauthorized individual might use this to get confidential information. Implementing appropriate access control measures prior to the presentation of sensitive material is essential to ensure that only authorized individuals may access client data. Furthermore, refrain from embedding sensitive data directly into the code.  
  
To enhance security throughout the codebase, it is essential to provide stringent input validation in all functions that take user input, including range checks and sanitization procedures. Prior to permitting users to modify sensitive data or execute operations, verify their authentication and authorization for such activities. Refrain from embedding sensitive information like as passwords, client data, or configuration settings directly in the source code. Consistently save passwords securely by using hashing algorithms such as bcrypt or Argon2, and avoid storing them in plain text. Ultimately, mitigate brute-force login attempts by restricting the amount of retries, locking the account after a certain number of unsuccessful attempts, or using captchas.