RazorHack 2024 Writeup

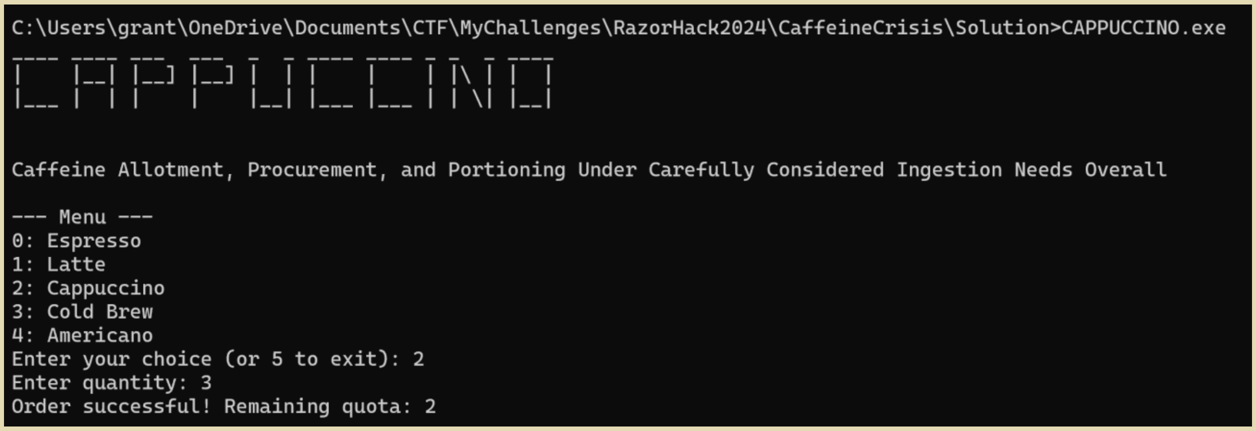
“Caffeine Crisis”

At RazorPower Energy Company, employees are given a generous allotment of up to 5 free coffees per week through the company's smart coffee delivery system. But for Max, the senior engineer with a crippling caffeine addiction, 5 coffees per week simply isn't enough to keep him functioning at full capacity.

Frustrated with the quota system, Max has enlisted your help to bypass the coffee limit. RazorPower's smart coffee system uses a custom network protocol to handle coffee orders, and Max suspects there's a way to exploit the system and score some extra brews.

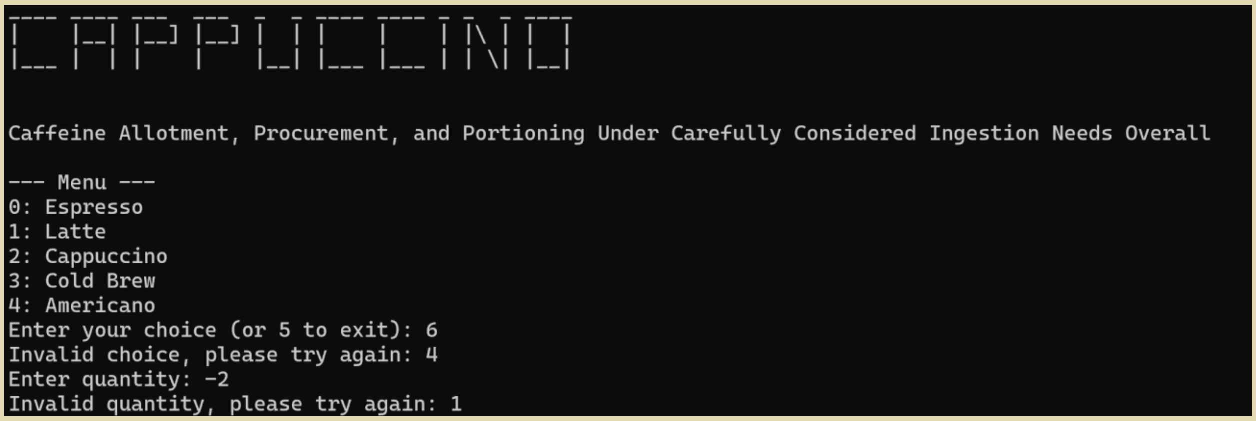
Author: Grant

In this challenge, participants are given the Portable Executable file *CAPPUCCINO.exe*. Opening up a command prompt and running this program, we are met with a command line program that enables users to order up to 5 of a few different options of coffees:



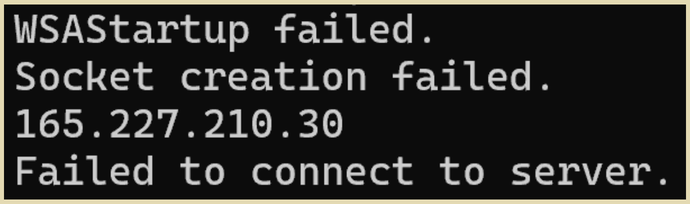
```
C:\Users\grant\OneDrive\Documents\CTF\MyChallenges\RazorHack2024\CaffeineCrisis\Solution>CAPPUCCINO.exe
____ ____ ___  ___  _  _ ____ ____ _ _  _ ____
|    |__| |__] |__] |  | |    |    | |\ | |  |
|___ |  | |    |    |__| |___ |___ | | \| |__|


Caffeine Allotment, Procurement, and Portioning Under Carefully Considered Ingestion Needs Overall

--- Menu ---
0: Espresso
1: Latte
2: Cappuccino
3: Cold Brew
4: Americano
Enter your choice (or 5 to exit): 2
Enter quantity: 3
Order successful! Remaining quota: 2
```

When tasked with exploiting programs in CTFs, it's good to start with trying various inputs that may be "unexpected" by the program. After playing around with various inputs, it becomes apparent the program restricts your inputs to a certain range of numbers for the choice and quantity fields.



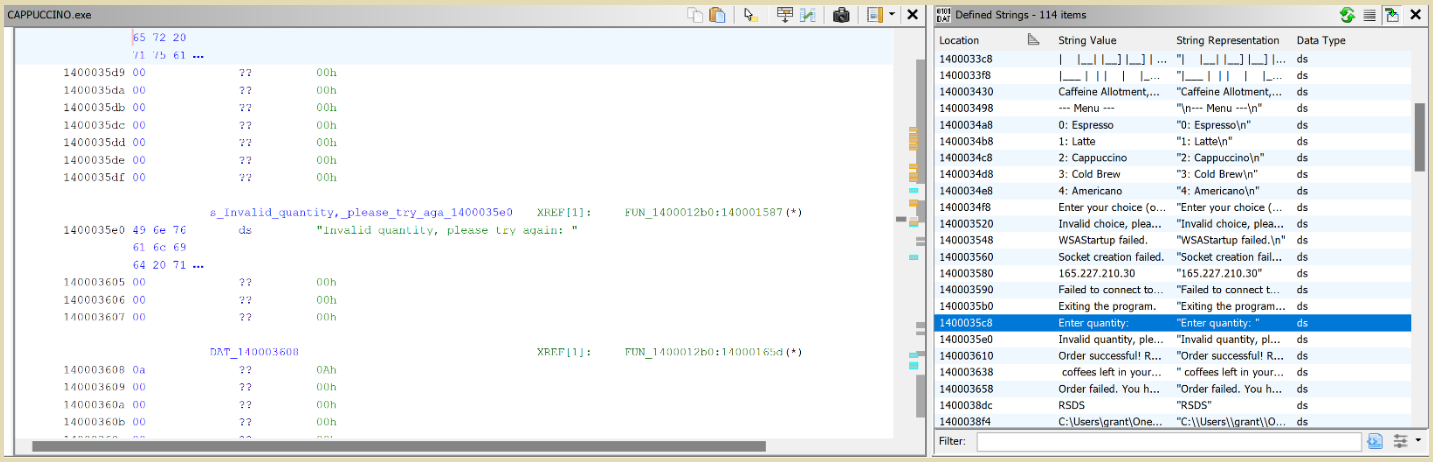
```
____ ____ ___  ___  _  _ ____ ____ _ _  _ ____
|    |__| |__] |__] |  | |    |    | |\ | |  |
|___ |  | |    |    |__| |___ |___ | | \| |__|


Caffeine Allotment, Procurement, and Portioning Under Carefully Considered Ingestion Needs Overall

--- Menu ---
0: Espresso
1: Latte
2: Cappuccino
3: Cold Brew
4: Americano
Enter your choice (or 5 to exit): 6
Invalid choice, please try again: 4
Enter quantity: -2
Invalid quantity, please try again: 1
```

With this in mind, let's take a deeper look into the program. I usually like to start by running the Linux command *strings* to see if the program has any low-hanging fruit. A few strings in particular tell me that this program is communicating with a server.



```
WSAStartup failed.
Socket creation failed.
165.227.210.30
Failed to connect to server.
```

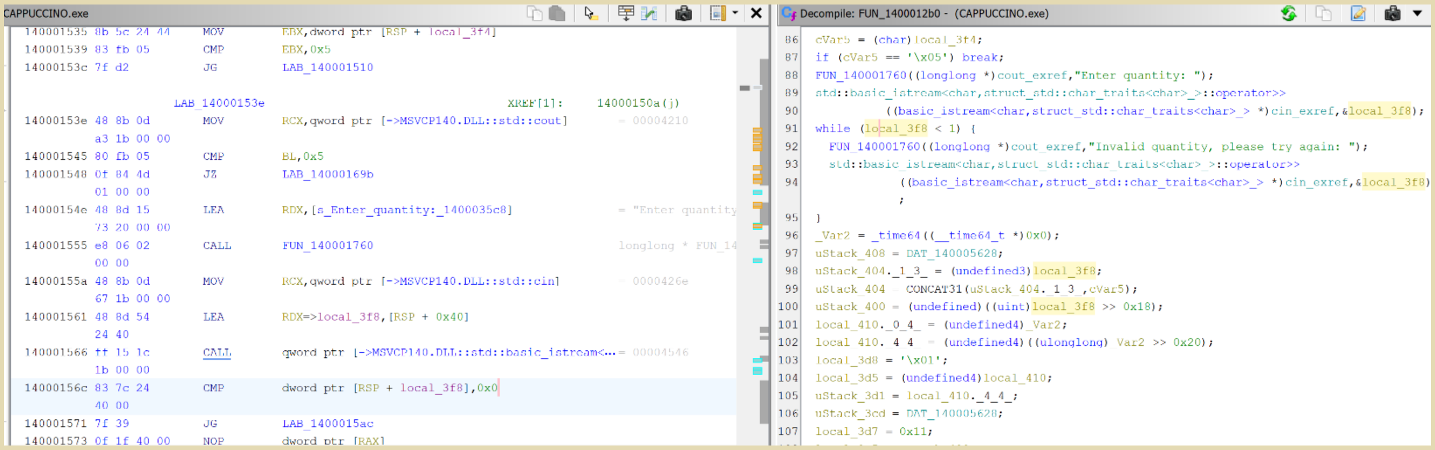
Knowing that the inputs are restricted and there is a server the program is communicating with, we can decompile the binary in Ghidra to determine if the program's input validation is on the client side or server side. The idea here is that, if the input validation is only on the client side, then we may be able to send the server a negative number to trick it into increasing our quota. Once Ghidra has finished its analysis, we can go to *Window->Defined Strings* and search for the strings that are printed by the program.



```
CAPPUCCINO.exe
                65 72 20
                71 75 61 ...
   1400035d9 00          ??          00h
   1400035da 00          ??          00h
   1400035db 00          ??          00h
   1400035dc 00          ??          00h
   1400035dd 00          ??          00h
   1400035de 00          ??          00h
   1400035df 00          ??          00h

                s_Invalid_quantity,_please_try_aga_1400035e0    XREF[1]:     FUN_1400012b0:140001587(*)
   1400035e0 49 6e 76    ds          "Invalid quantity, please try again: "
             61 6c 69
             64 20 71 ...
   140003605 00          ??          00h
   140003606 00          ??          00h
   140003607 00          ??          00h

                DAT_140003608                                   XREF[1]:     FUN_1400012b0:14000165d(*)
   140003608 0a          ??          0Ah
   140003609 00          ??          00h
   14000360a 00          ??          00h
   14000360b 00          ??          00h
```

```
Defined Strings - 114 items
Location     String Value              String Representation        Data Type
1400033c8    |   |__| |__] |__] | ...  "|   |__| |__] |__] |...     ds
1400033f8    |___ | | |   |   |_...    "|___ | | |   |   |_...      ds
140003430    Caffeine Allotment,...    "Caffeine Allotment,...      ds
140003498    --- Menu ---              "\n--- Menu ---\n"           ds
1400034a8    0: Espresso               "0: Espresso\n"              ds
1400034b8    1: Latte                  "1: Latte\n"                 ds
1400034c8    2: Cappuccino             "2: Cappuccino\n"            ds
1400034d8    3: Cold Brew              "3: Cold Brew\n"             ds
1400034e8    4: Americano              "4: Americano\n"             ds
1400034f8    Enter your choice (o...   "Enter your choice (...      ds
140003520    Invalid choice, plea...   "Invalid choice, plea...     ds
140003548    WSAStartup failed.        "WSAStartup failed.\n"       ds
140003560    Socket creation failed.   "Socket creation fail...     ds
140003580    165.227.210.30            "165.227.210.30"             ds
140003590    Failed to connect to...   "Failed to connect t...      ds
1400035b0    Exiting the program.      "Exiting the program...      ds
1400035c8    Enter quantity:           "Enter quantity: "           ds
1400035e0    Invalid quantity, ple...  "Invalid quantity, pl...     ds
140003610    Order successful! R...    "Order successful! R...      ds
140003638     coffees left in your...  " coffees left in your...    ds
140003658    Order failed. You h...    "Order failed. You h...      ds
1400038dc    RSDS                      "RSDS"                       ds
1400038f4    C:\Users\grant\One...     "C:\\Users\\grant\\O...      ds
Filter:
```

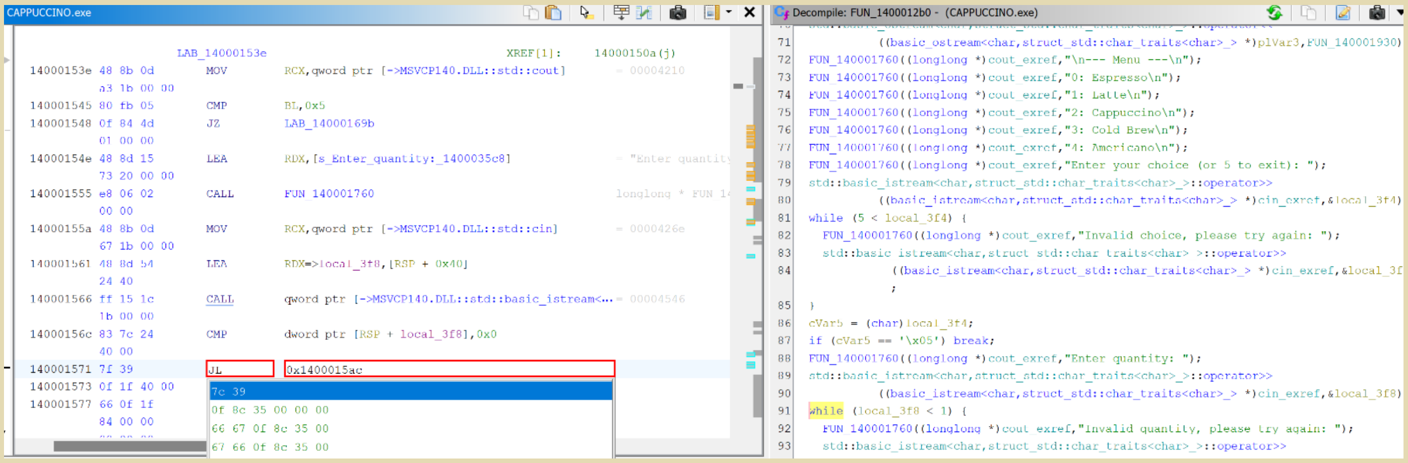
Clicking on the XREF to the "Invalid quantity" string, we are taken to the part of the code that references it.



```
CAPPUCCINO.exe
 140001535 8b 5c 24 44    MOV     EBX,dword ptr [RSP + local_3f4]
 140001539 83 fb 05       CMP     EBX,0x5
 14000153c 7f d2          JG      LAB_140001510

                    LAB_14000153e                         XREF[1]:     14000150a(j)
 14000153e 48 8b 0d       MOV     RCX,qword ptr [->MSVCP140.DLL::std::cout]        = 00004210
           a3 1b 00 00
 140001545 80 fb 05       CMP     BL,0x5
 140001548 0f 84 4d       JZ      LAB_14000169b
           01 00 00
 14000154e 48 8d 15       LEA     RDX,[s_Enter_quantity:_1400035c8]                = "Enter quantity
           73 20 00 00
 140001555 e8 06 02       CALL    FUN_140001760                                    longlong * FUN_14
           00 00
 14000155a 48 8b 0d       MOV     RCX,qword ptr [->MSVCP140.DLL::std::cin]         = 0000426e
           67 1b 00 00
 140001561 48 8d 54       LEA     RDX=>local_3f8,[RSP + 0x40]
           24 40
 140001566 ff 15 1c       CALL    qword ptr [->MSVCP140.DLL::std::basic_istream<...= 00004546
           1b 00 00
 14000156c 83 7c 24       CMP     dword ptr [RSP + local_3f8],0x0
           40 00
 140001571 7f 39          JG      LAB_1400015ac
 140001573 0f 1f 40 00    NOP     dword ptr [RAX]
```

```
Decompile: FUN_1400012b0 -  (CAPPUCCINO.exe)
 86  cVar5 = (char)local_3f4;
 87  if (cVar5 == '\x05') break;
 88  FUN_140001760((longlong *)cout_exref,"Enter quantity: ");
 89  std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
 90            ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f8);
 91  while (local_3f8 < 1) {
 92    FUN_140001760((longlong *)cout_exref,"Invalid quantity, please try again: ");
 93    std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
 94              ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f8)
               ;
 95  }
 96  _Var2 = _time64((__time64_t *)0x0);
 97  uStack_408 = DAT_140005628;
 98  uStack_404._1_3_ = (undefined3)local_3f8;
 99  uStack_404 = CONCAT31(uStack_404._1_3_,cVar5);
100  uStack_400 = (undefined)((uint)local_3f8 >> 0x18);
101  local_410._0_4_ = (undefined4)_Var2;
102  local_410._4_4_ = (undefined4)((ulonglong)_Var2 >> 0x20);
103  local_3d8 = '\x01';
104  local_3d5 = (undefined4)local_410;
105  uStack_3d1 = local_410._4_4_;
106  uStack_3cd = DAT_140005628;
107  local_3d7 = 0x11;
```

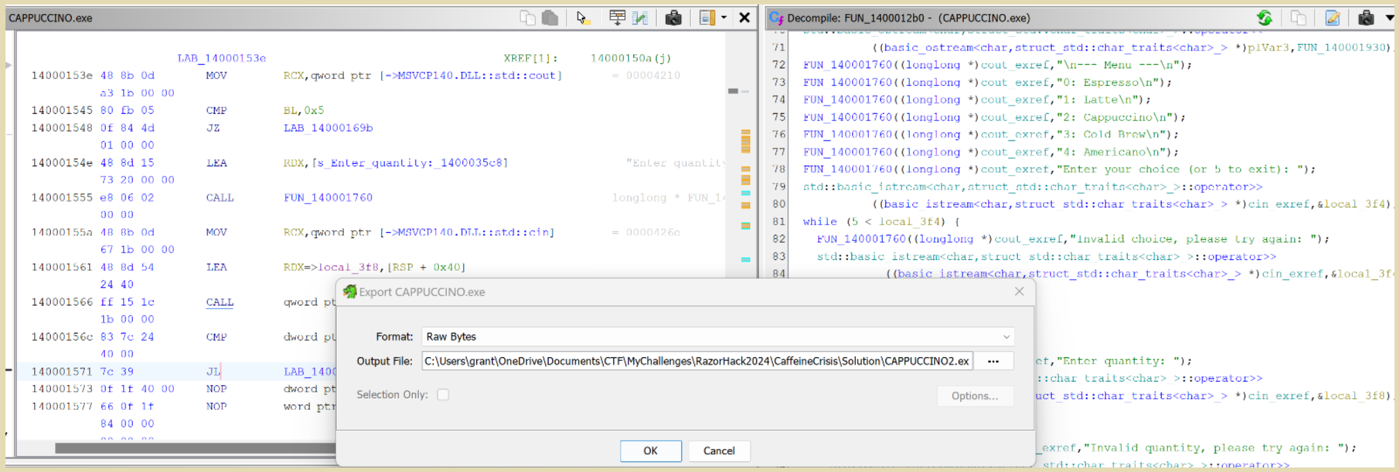
Looking at lines 89-95 of the decompilation, we can see that the user's input, denoted by local_3f8, is rejected until you enter an input greater than or equal to 1. A powerful tool at our disposal in Ghidra is binary patching. This allows us to alter the behavior of the program by changing the bytes of its instructions. With line 89 selected, we can see the corresponding disassembled instructions in the disassembly window. The CMP dword ptr [RSP + local_3f8], 0x0 instruction is comparing our input to 0. In conjunction with the JG LAB_1400015ac instruction, the program will jump to LAB_1400015ac (the label that prints the error and prompts for another input) if the user's input is less than or equal to 0. So, if we patch the JG LAB_1400015ac instruction to be a JL instruction instead, the program will accept negative inputs. To do this, we can right click on the instruction and click Patch Instruction:



```
                     LAB_14000153e                          XREF[1]:     14000150a(j)
14000153e 48 8b 0d      MOV        RCX,qword ptr [->MSVCP140.DLL::std::cout]         = 00004210
          a3 1b 00 00
140001545 80 fb 05      CMP        BL,0x5
140001548 0f 84 4d      JZ         LAB_14000169b
          01 00 00
14000154e 48 8d 15      LEA        RDX,[s_Enter_quantity:_1400035c8]                 = "Enter quantity
          73 20 00 00
140001555 e8 06 02      CALL       FUN_140001760                                     longlong * FUN_14
          00 00
14000155a 48 8b 0d      MOV        RCX,qword ptr [->MSVCP140.DLL::std::cin]          = 0000426e
          67 1b 00 00
140001561 48 8d 54      LEA        RDX=>local_3f8,[RSP + 0x40]
          24 40
140001566 ff 15 1c      CALL       qword ptr [->MSVCP140.DLL::std::basic_istream<... = 00004546
          1b 00 00
14000156c 83 7c 24      CMP        dword ptr [RSP + local_3f8],0x0
          40 00
140001571 7f 39         JL         0x1400015ac
140001573 0f 1f 40 00              7c 39
140001577 66 0f 1f                 0f 8c 35 00 00 00
          84 00 00                 66 67 0f 8c 35 00
                                   67 66 0f 8c 35 00
```

```
71              ((basic_ostream<char,struct_std::char_traits<char>_> *)plVar3,FUN_140001930)
72  FUN_140001760((longlong *)cout_exref,"\n--- Menu ---\n");
73  FUN_140001760((longlong *)cout_exref,"0: Espresso\n");
74  FUN_140001760((longlong *)cout_exref,"1: Latte\n");
75  FUN_140001760((longlong *)cout_exref,"2: Cappuccino\n");
76  FUN_140001760((longlong *)cout_exref,"3: Cold Brew\n");
77  FUN_140001760((longlong *)cout_exref,"4: Americano\n");
78  FUN_140001760((longlong *)cout_exref,"Enter your choice (or 5 to exit): ");
79  std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
80              ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f4)
81  while (5 < local_3f4) {
82    FUN_140001760((longlong *)cout_exref,"Invalid choice, please try again: ");
83    std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
84              ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f
                ;
85  }
86  cVar5 = (char)local_3f4;
87  if (cVar5 == '\x05') break;
88  FUN_140001760((longlong *)cout_exref,"Enter quantity: ");
89  std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
90              ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f8)
91  while (local_3f8 < 1) {
92    FUN_140001760((longlong *)cout_exref,"Invalid quantity, please try again: ");
93    std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
```

Then, we can export the patched version of the binary by going to File->Export

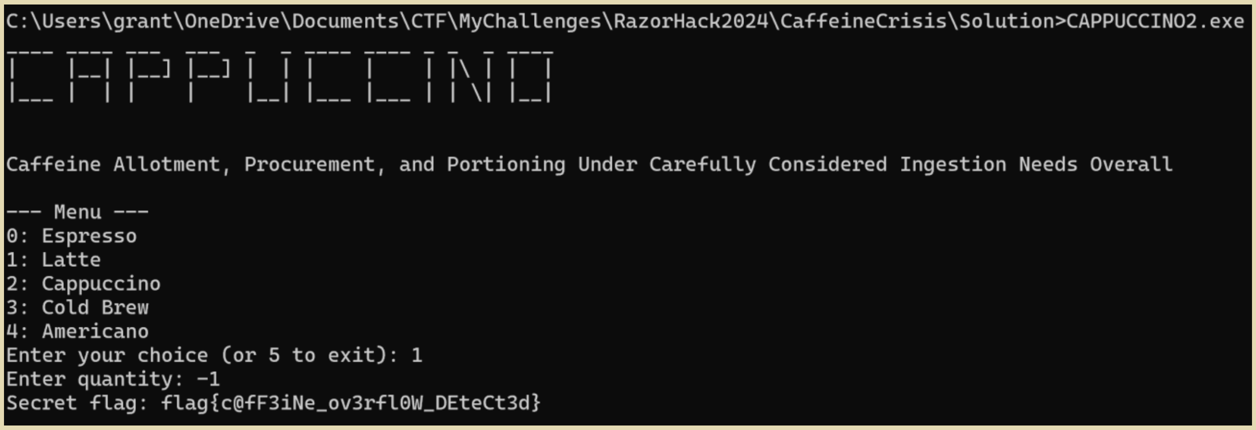


```
                     LAB_14000153e                          XREF[1]:     14000150a(j)
14000153e 48 8b 0d      MOV        RCX,qword ptr [->MSVCP140.DLL::std::cout]         = 00004210
          a3 1b 00 00
140001545 80 fb 05      CMP        BL,0x5
140001548 0f 84 4d      JZ         LAB_14000169b
          01 00 00
14000154e 48 8d 15      LEA        RDX,[s_Enter_quantity:_1400035c8]                 "Enter quantity
          73 20 00 00
140001555 e8 06 02      CALL       FUN_140001760                                     longlong * FUN_14
          00 00
14000155a 48 8b 0d      MOV        RCX,qword ptr [->MSVCP140.DLL::std::cin]          = 0000426e
          67 1b 00 00
140001561 48 8d 54      LEA        RDX=>local_3f8,[RSP + 0x40]
          24 40
140001566 ff 15 1c      CALL       qword pt
          1b 00 00
14000156c 83 7c 24      CMP        dword pt
          40 00
140001571 7c 39         JL         LAB_1400
140001573 0f 1f 40 00   NOP        dword pt
140001577 66 0f 1f      NOP        word ptr
          84 00 00
```

```
71              ((basic_ostream<char,struct_std::char_traits<char>_> *)plVar3,FUN_140001930)
72  FUN_140001760((longlong *)cout_exref,"\n--- Menu ---\n");
73  FUN_140001760((longlong *)cout_exref,"0: Espresso\n");
74  FUN_140001760((longlong *)cout_exref,"1: Latte\n");
75  FUN_140001760((longlong *)cout_exref,"2: Cappuccino\n");
76  FUN_140001760((longlong *)cout_exref,"3: Cold Brew\n");
77  FUN_140001760((longlong *)cout_exref,"4: Americano\n");
78  FUN_140001760((longlong *)cout_exref,"Enter your choice (or 5 to exit): ");
79  std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
80              ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f4)
81  while (5 < local_3f4) {
82    FUN_140001760((longlong *)cout_exref,"Invalid choice, please try again: ");
83    std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
84              ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,&local_3f
                                  ef,"Enter quantity: ");
                                  ::char_traits<char>_>::operator>>
                                  uct_std::char_traits<char>_> *)cin_exref,&local_3f8)
                                  _exref,"Invalid quantity, please try again: ");
```

Format: Raw Bytes
Output File: C:\Users\grant\OneDrive\Documents\CTF\MyChallenges\RazorHack2024\CaffeineCrisis\Solution\CAPPUCCINO2.ex
Selection Only: ☐

Using our newly patched binary, we can input negative numbers and retrieve the flag:



```
C:\Users\grant\OneDrive\Documents\CTF\MyChallenges\RazorHack2024\CaffeineCrisis\Solution>CAPPUCCINO2.exe
____ ____ ___  ___  _  _ ____ ____ _ _  _ ____
|    |__| |__] |__] |  | |    |    | |\ | |  |
|___ |  | |    |    |__| |___ |___ | | \| |__|


Caffeine Allotment, Procurement, and Portioning Under Carefully Considered Ingestion Needs Overall

--- Menu ---
0: Espresso
1: Latte
2: Cappuccino
3: Cold Brew
4: Americano
Enter your choice (or 5 to exit): 1
Enter quantity: -1
Secret flag: flag{c@fF3iNe_ov3rfl0W_DEteCt3d}
```

It is also worth noting that another way this challenge could be solved is by reverse engineering the protocol the program is using and spoofing packets directly to the server using a tool like Scapy.