Computer Security HW2 Write-Up

Account: soyccan
Name: 陳正康
Student ID: B07902143

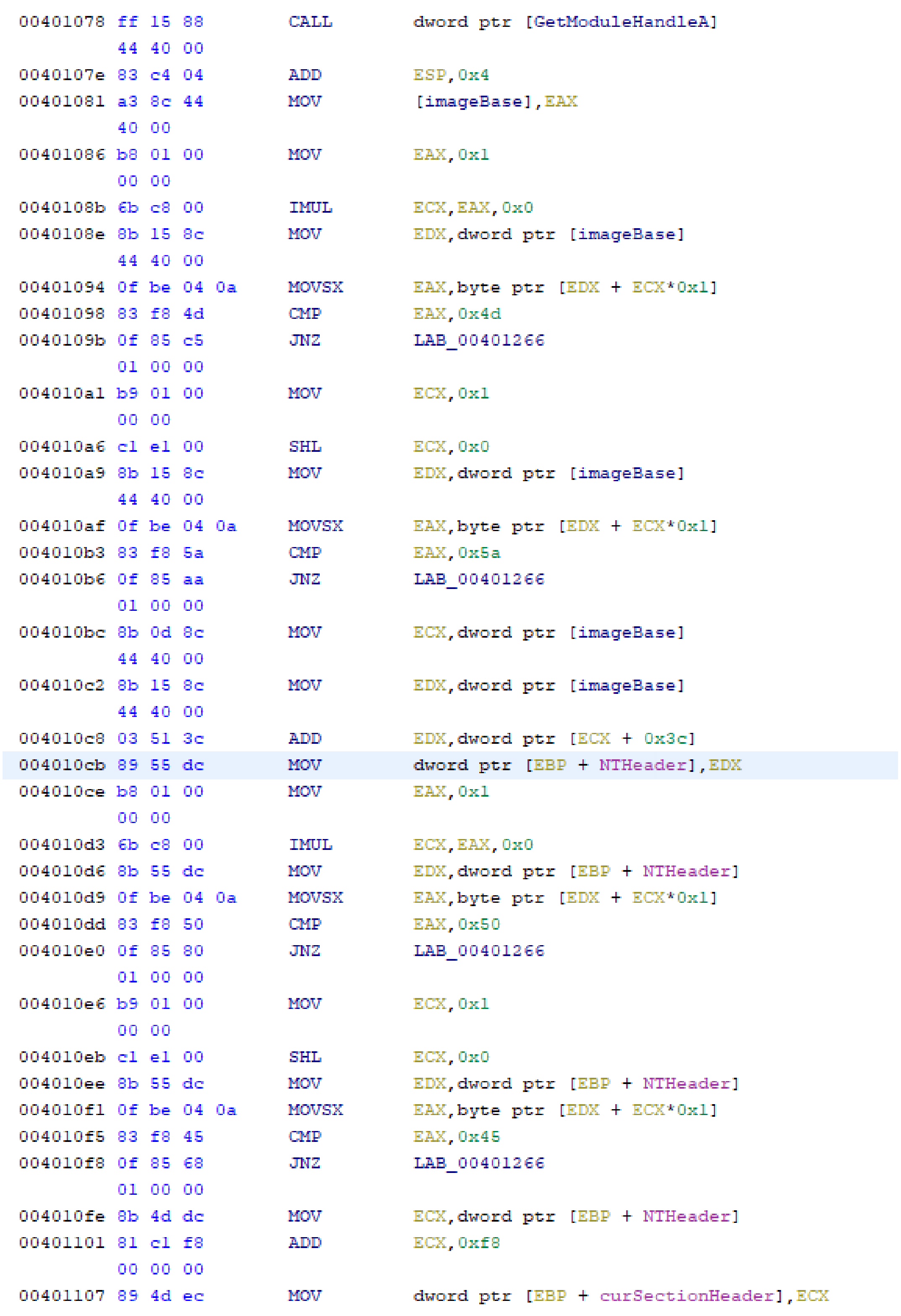
main()

main() function reads 2 variable from input: `seed`, `flag`
then calls 2 functions, let’s say:

- `decrypt_secret(seed)`
- `check_flag(flag)`

decrypt_secret(seed)

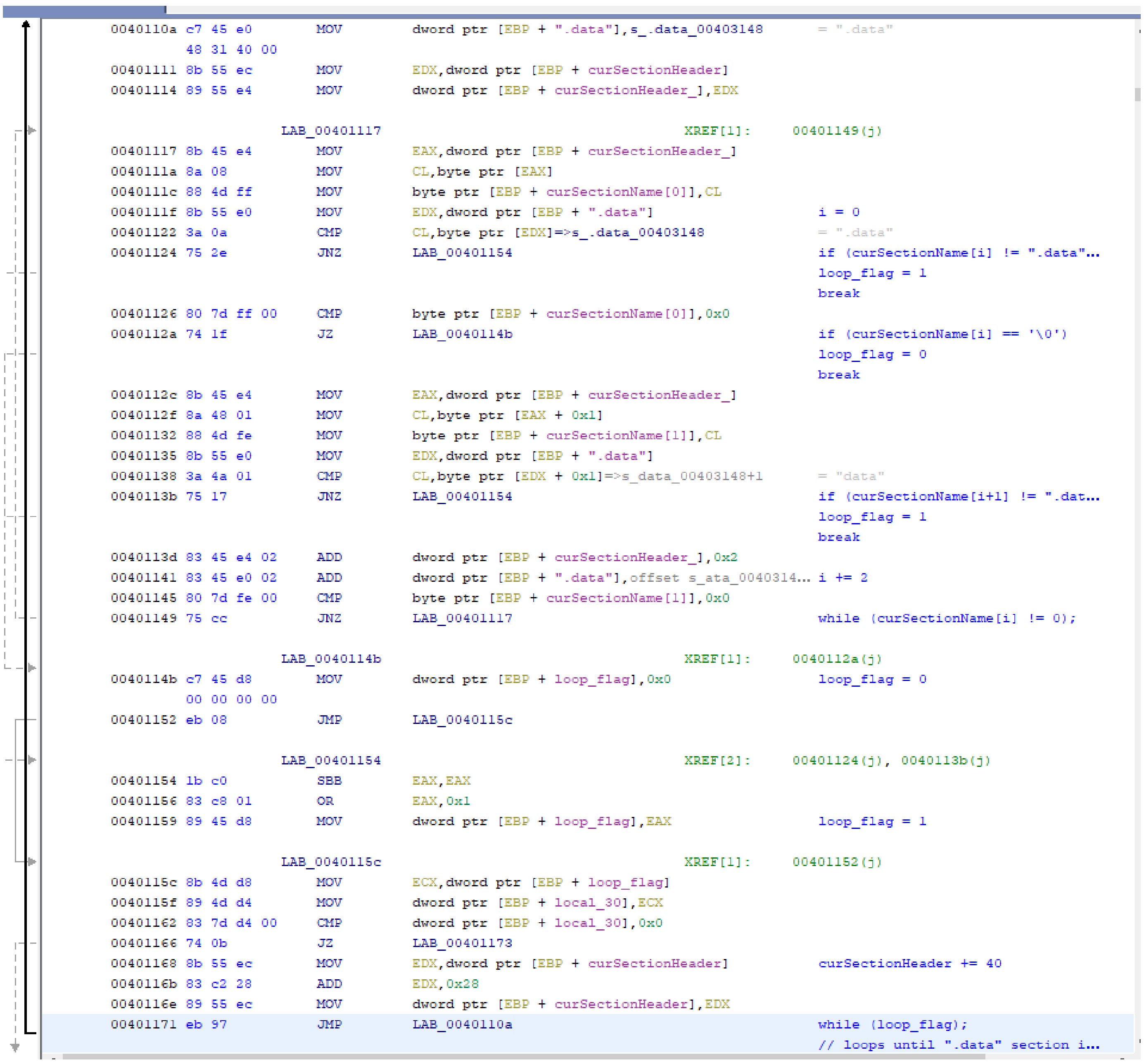
part 1: locate section headers



```
00401078 ff 15 88        CALL      dword ptr [GetModuleHandleA]
         44 40 00
0040107e 83 c4 04        ADD       ESP,0x4
00401081 a3 8c 44        MOV       [imageBase],EAX
         40 00
00401086 b8 01 00        MOV       EAX,0x1
         00 00
0040108b 6b c8 00        IMUL      ECX,EAX,0x0
0040108e 8b 15 8c        MOV       EDX,dword ptr [imageBase]
         44 40 00
00401094 0f be 04 0a     MOVSX     EAX,byte ptr [EDX + ECX*0x1]
00401098 83 f8 4d        CMP       EAX,0x4d
0040109b 0f 85 c5        JNZ       LAB_00401266
         01 00 00
004010a1 b9 01 00        MOV       ECX,0x1
         00 00
004010a6 c1 e1 00        SHL       ECX,0x0
004010a9 8b 15 8c        MOV       EDX,dword ptr [imageBase]
         44 40 00
004010af 0f be 04 0a     MOVSX     EAX,byte ptr [EDX + ECX*0x1]
004010b3 83 f8 5a        CMP       EAX,0x5a
004010b6 0f 85 aa        JNZ       LAB_00401266
         01 00 00
004010bc 8b 0d 8c        MOV       ECX,dword ptr [imageBase]
         44 40 00
004010c2 8b 15 8c        MOV       EDX,dword ptr [imageBase]
         44 40 00
004010c8 03 51 3c        ADD       EDX,dword ptr [ECX + 0x3c]
004010cb 89 55 dc        MOV       dword ptr [EBP + NTHeader],EDX
004010ce b8 01 00        MOV       EAX,0x1
         00 00
004010d3 6b c8 00        IMUL      ECX,EAX,0x0
004010d6 8b 55 dc        MOV       EDX,dword ptr [EBP + NTHeader]
004010d9 0f be 04 0a     MOVSX     EAX,byte ptr [EDX + ECX*0x1]
004010dd 83 f8 50        CMP       EAX,0x50
004010e0 0f 85 80        JNZ       LAB_00401266
         01 00 00
004010e6 b9 01 00        MOV       ECX,0x1
         00 00
004010eb c1 e1 00        SHL       ECX,0x0
004010ee 8b 55 dc        MOV       EDX,dword ptr [EBP + NTHeader]
004010f1 0f be 04 0a     MOVSX     EAX,byte ptr [EDX + ECX*0x1]
004010f5 83 f8 45        CMP       EAX,0x45
004010f8 0f 85 68        JNZ       LAB_00401266
         01 00 00
004010fe 8b 4d dc        MOV       ECX,dword ptr [EBP + NTHeader]
00401101 81 c1 f8        ADD       ECX,0xf8
         00 00 00
00401107 89 4d ec        MOV       dword ptr [EBP + curSectionHeader],ECX
```

1. call `GetModuleHandleA` to get address where image base is loaded (starts with “MZ”)
2. get address of NT header from `*(imageBase + 0x3c)` (starts with “PE”)
3. get address of first section header from `NTHeader + 0xf8`

part 2: search for .data section

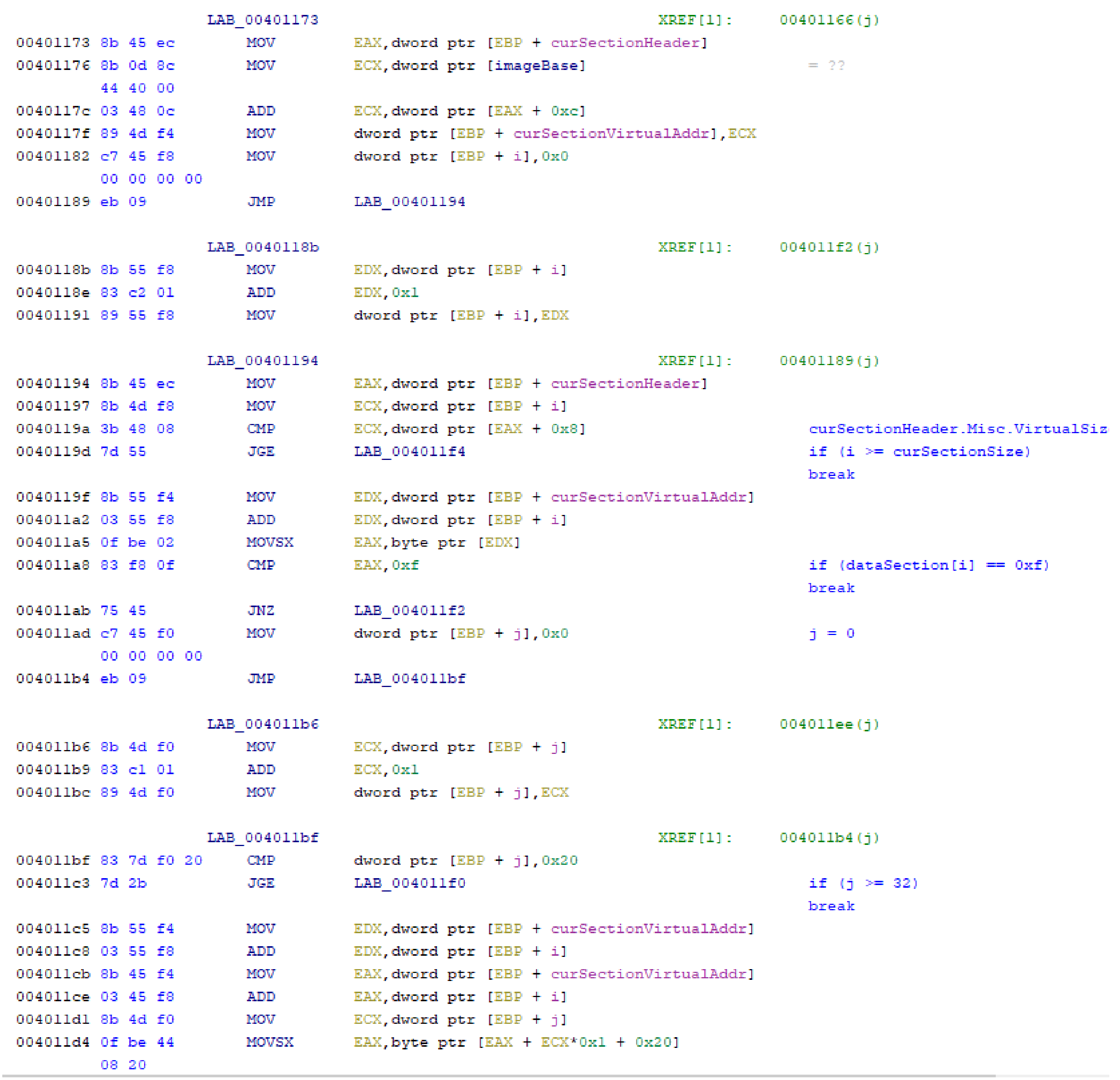


```
0040110a c7 45 e0      MOV     dword ptr [EBP + ".data"],s_.data_00403148        = ".data"
         48 31 40 00
00401111 8b 55 ec      MOV     EDX,dword ptr [EBP + curSectionHeader]
00401114 89 55 e4      MOV     dword ptr [EBP + curSectionHeader_],EDX

                  LAB_00401117                               XREF[1]:     00401149(j)
00401117 8b 45 e4      MOV     EAX,dword ptr [EBP + curSectionHeader_]
0040111a 8a 08         MOV     CL,byte ptr [EAX]
0040111c 88 4d ff      MOV     byte ptr [EBP + curSectionName[0]],CL
0040111f 8b 55 e0      MOV     EDX,dword ptr [EBP + ".data"]                   i = 0
00401122 3a 0a         CMP     CL,byte ptr [EDX]=>s_.data_00403148             = ".data"
00401124 75 2e         JNZ     LAB_00401154                                    if (curSectionName[i] != ".data"…
                                                                               loop_flag = 1
                                                                               break
00401126 80 7d ff 00   CMP     byte ptr [EBP + curSectionName[0]],0x0
0040112a 74 1f         JZ      LAB_0040114b                                    if (curSectionName[i] == '\0')
                                                                               loop_flag = 0
                                                                               break
0040112c 8b 45 e4      MOV     EAX,dword ptr [EBP + curSectionHeader_]
0040112f 8a 48 01      MOV     CL,byte ptr [EAX + 0x1]
00401132 88 4d fe      MOV     byte ptr [EBP + curSectionName[1]],CL
00401135 8b 55 e0      MOV     EDX,dword ptr [EBP + ".data"]
00401138 3a 4a 01      CMP     CL,byte ptr [EDX + 0x1]=>s_data_00403148+1      = "data"
0040113b 75 17         JNZ     LAB_00401154                                    if (curSectionName[i+1] != ".dat…
                                                                               loop_flag = 1
                                                                               break
0040113d 83 45 e4 02   ADD     dword ptr [EBP + curSectionHeader_],0x2
00401141 83 45 e0 02   ADD     dword ptr [EBP + ".data"],offset s_ata_0040314… i += 2
00401145 80 7d fe 00   CMP     byte ptr [EBP + curSectionName[1]],0x0
00401149 75 cc         JNZ     LAB_00401117                                    while (curSectionName[i] != 0);

                  LAB_0040114b                               XREF[1]:     0040112a(j)
0040114b c7 45 d8      MOV     dword ptr [EBP + loop_flag],0x0                 loop_flag = 0
         00 00 00 00
00401152 eb 08         JMP     LAB_0040115c

                  LAB_00401154                               XREF[2]:     00401124(j), 0040113b(j)
00401154 1b c0         SBB     EAX,EAX
00401156 83 c8 01      OR      EAX,0x1
00401159 89 45 d8      MOV     dword ptr [EBP + loop_flag],EAX                 loop_flag = 1

                  LAB_0040115c                               XREF[1]:     00401152(j)
0040115c 8b 4d d8      MOV     ECX,dword ptr [EBP + loop_flag]
0040115f 89 4d d4      MOV     dword ptr [EBP + local_30],ECX
00401162 83 7d d4 00   CMP     dword ptr [EBP + local_30],0x0
00401166 74 0b         JZ      LAB_00401173
00401168 8b 55 ec      MOV     EDX,dword ptr [EBP + curSectionHeader]          curSectionHeader += 40
0040116b 83 c2 28      ADD     EDX,0x28
0040116e 89 55 ec      MOV     dword ptr [EBP + curSectionHeader],EDX
00401171 eb 97         JMP     LAB_0040110a                                    while (loop_flag);
                                                                               // loops until ".data" section i…
```

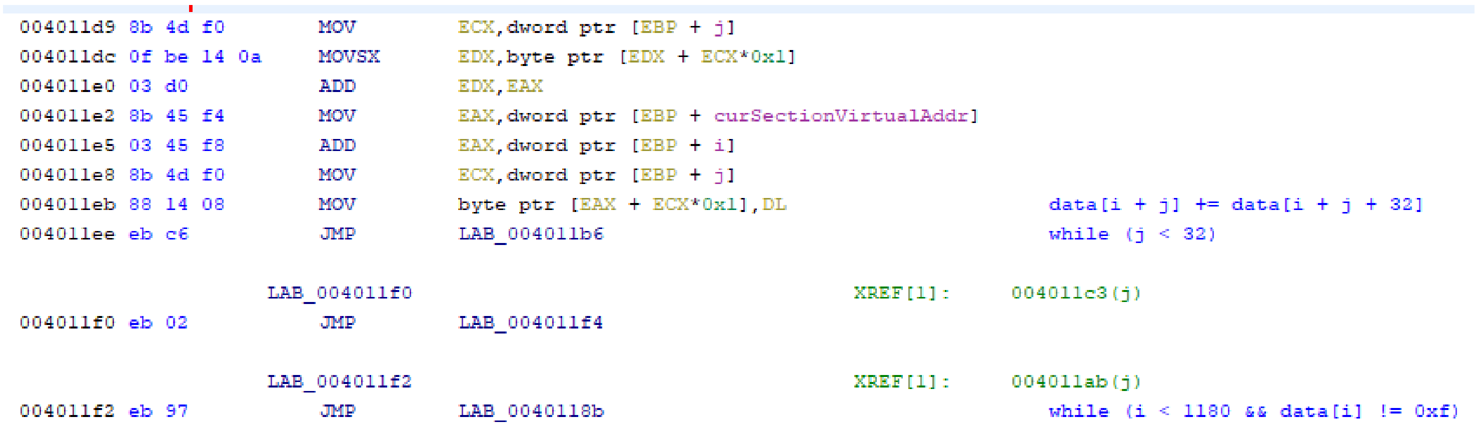
there is loop which checks the name of each section header
loop until ".data" is found
eqivalent C code:

```
targetName = ".data"
loop_flag = 1;
do {
    curSectionName = curSectionHeader;
    i = 0;
    while (1) {
        if (curSectionName[i] != targetName[i]) {
            loop_flag = 1;
            break;
        }
        else if (curSectionName[i] == '\0') {
            loop_flag = 0;
            break;
        }
        else if (curSectionName[i+1] != targetName[i+1]) {
            loop_flag = 1;
            break;
        }
        i += 2;
    }
    curSectionHeader += 0x28; // next section header
} while (loop_flag);
```

part 3: manipulating the secret



```
                     LAB_00401173                                  XREF[1]:     00401166(j)
00401173 8b 45 ec       MOV        EAX,dword ptr [EBP + curSectionHeader]
00401176 8b 0d 8c       MOV        ECX,dword ptr [imageBase]                          = ??
         44 40 00
0040117c 03 48 0c       ADD        ECX,dword ptr [EAX + 0xc]
0040117f 89 4d f4       MOV        dword ptr [EBP + curSectionVirtualAddr],ECX
00401182 c7 45 f8       MOV        dword ptr [EBP + i],0x0
         00 00 00 00
00401189 eb 09          JMP        LAB_00401194

                     LAB_0040118b                                  XREF[1]:     004011f2(j)
0040118b 8b 55 f8       MOV        EDX,dword ptr [EBP + i]
0040118e 83 c2 01       ADD        EDX,0x1
00401191 89 55 f8       MOV        dword ptr [EBP + i],EDX

                     LAB_00401194                                  XREF[1]:     00401189(j)
00401194 8b 45 ec       MOV        EAX,dword ptr [EBP + curSectionHeader]
00401197 8b 4d f8       MOV        ECX,dword ptr [EBP + i]
0040119a 3b 48 08       CMP        ECX,dword ptr [EAX + 0x8]                        curSectionHeader.Misc.VirtualSiz
0040119d 7d 55          JGE        LAB_004011f4                                     if (i >= curSectionSize)
                                                                                    break
0040119f 8b 55 f4       MOV        EDX,dword ptr [EBP + curSectionVirtualAddr]
004011a2 03 55 f8       ADD        EDX,dword ptr [EBP + i]
004011a5 0f be 02       MOVSX      EAX,byte ptr [EDX]
004011a8 83 f8 0f       CMP        EAX,0xf                                          if (dataSection[i] == 0xf)
                                                                                    break
004011ab 75 45          JNZ        LAB_004011f2
004011ad c7 45 f0       MOV        dword ptr [EBP + j],0x0                          j = 0
         00 00 00 00
004011b4 eb 09          JMP        LAB_004011bf

                     LAB_004011b6                                  XREF[1]:     004011ee(j)
004011b6 8b 4d f0       MOV        ECX,dword ptr [EBP + j]
004011b9 83 c1 01       ADD        ECX,0x1
004011bc 89 4d f0       MOV        dword ptr [EBP + j],ECX

                     LAB_004011bf                                  XREF[1]:     004011b4(j)
004011bf 83 7d f0 20    CMP        dword ptr [EBP + j],0x20
004011c3 7d 2b          JGE        LAB_004011f0                                     if (j >= 32)
                                                                                    break
004011c5 8b 55 f4       MOV        EDX,dword ptr [EBP + curSectionVirtualAddr]
004011c8 03 55 f8       ADD        EDX,dword ptr [EBP + i]
004011cb 8b 45 f4       MOV        EAX,dword ptr [EBP + curSectionVirtualAddr]
004011ce 03 45 f8       ADD        EAX,dword ptr [EBP + i]
004011d1 8b 4d f0       MOV        ECX,dword ptr [EBP + j]
004011d4 0f be 44       MOVSX      EAX,byte ptr [EAX + ECX*0x1 + 0x20]
         08 20
```

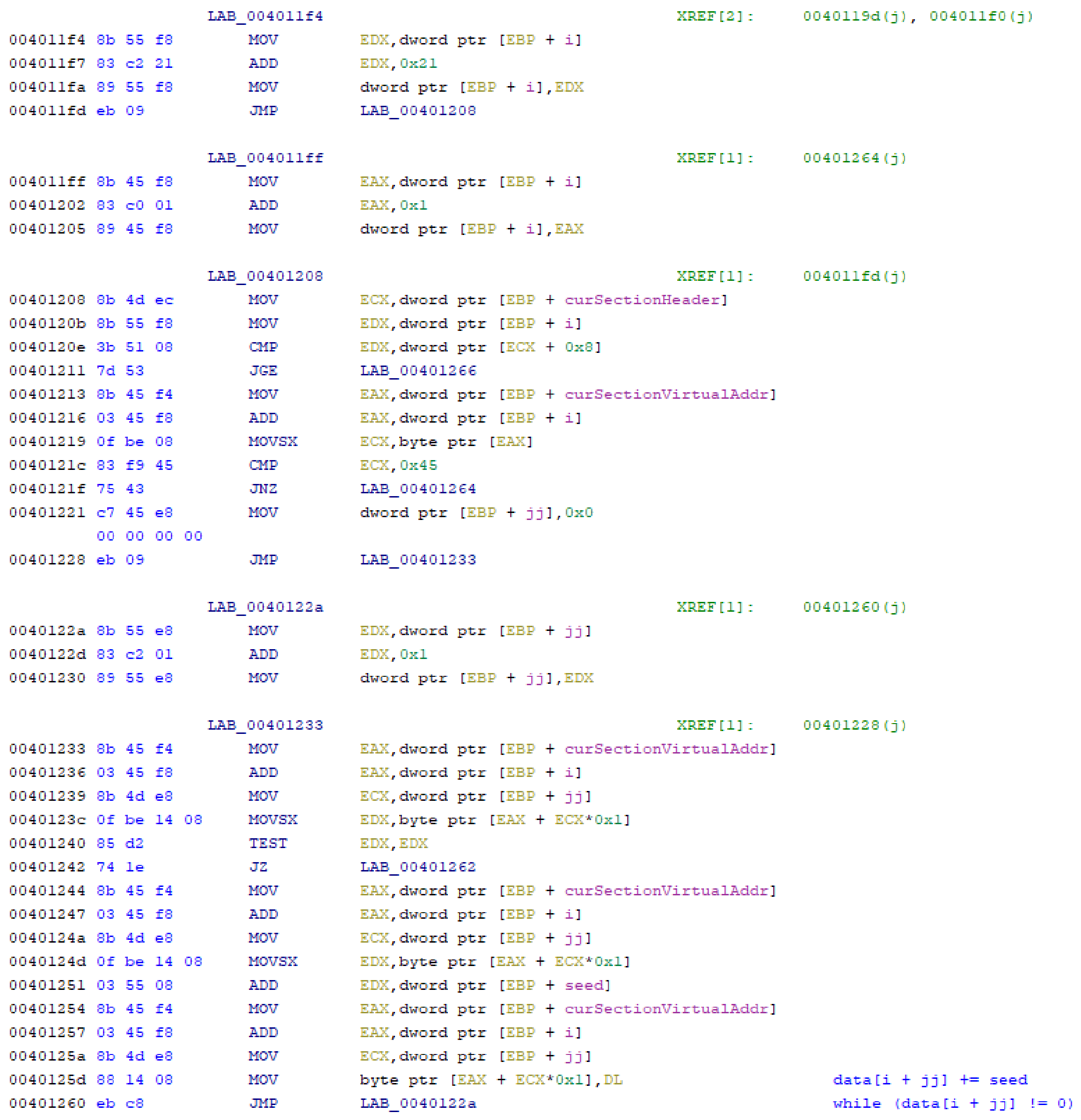


```
004011d9 8b 4d f0       MOV        ECX,dword ptr [EBP + j]
004011dc 0f be 14 0a    MOVSX      EDX,byte ptr [EDX + ECX*0x1]
004011e0 03 d0          ADD        EDX,EAX
004011e2 8b 45 f4       MOV        EAX,dword ptr [EBP + curSectionVirtualAddr]
004011e5 03 45 f8       ADD        EAX,dword ptr [EBP + i]
004011e8 8b 4d f0       MOV        ECX,dword ptr [EBP + j]
004011eb 88 14 08       MOV        byte ptr [EAX + ECX*0x1],DL                      data[i + j] += data[i + j + 32]
004011ee eb c6          JMP        LAB_004011b6                                     while (j < 32)

                     LAB_004011f0                                  XREF[1]:     004011c3(j)
004011f0 eb 02          JMP        LAB_004011f4

                     LAB_004011f2                                  XREF[1]:     004011ab(j)
004011f2 eb 97          JMP        LAB_0040118b                                     while (i < 1180 && data[i] != 0xf)
```

1. get virtual address of ".data" section from `*(curSectionHeader + 0xc)`
2. modify secret array `A` (starts with 0xf) in ".data" section:

```
// curSectionHeader is ".data" section
data = curSectionHeader.VirtualAddress
size = curSectionHeader.Misc.VirtualSize

for (i = 0; i < size; i++) {
    if (data[i] == 0xf) {
        for (j = 0; j < 32; j++)
            data[i + j] += data[i + j + 32];
        break;
    }
}
```

3. modify secret array B (starts with 0x45) in ".data" section:

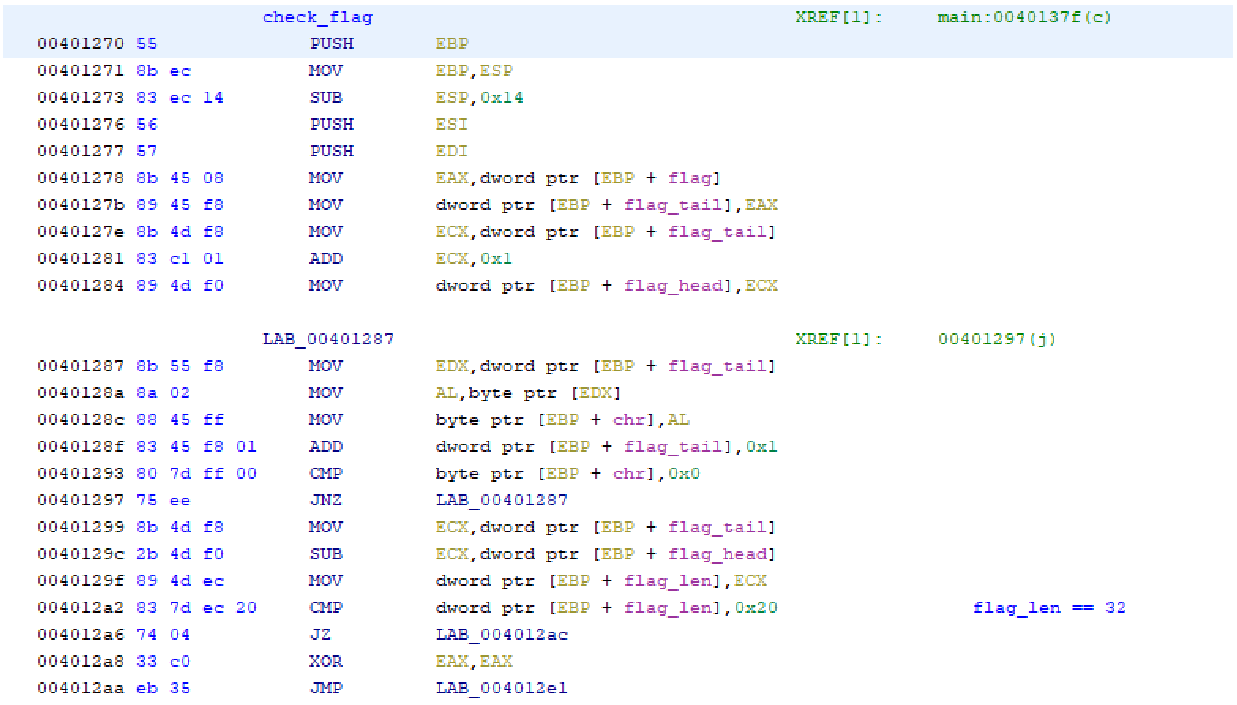


```
                     LAB_004011f4                          XREF[2]:     0040119d(j), 004011f0(j)
004011f4 8b 55 f8        MOV        EDX,dword ptr [EBP + i]
004011f7 83 c2 21        ADD        EDX,0x21
004011fa 89 55 f8        MOV        dword ptr [EBP + i],EDX
004011fd eb 09           JMP        LAB_00401208

                     LAB_004011ff                          XREF[1]:     00401264(j)
004011ff 8b 45 f8        MOV        EAX,dword ptr [EBP + i]
00401202 83 c0 01        ADD        EAX,0x1
00401205 89 45 f8        MOV        dword ptr [EBP + i],EAX

                     LAB_00401208                          XREF[1]:     004011fd(j)
00401208 8b 4d ec        MOV        ECX,dword ptr [EBP + curSectionHeader]
0040120b 8b 55 f8        MOV        EDX,dword ptr [EBP + i]
0040120e 3b 51 08        CMP        EDX,dword ptr [ECX + 0x8]
00401211 7d 53           JGE        LAB_00401266
00401213 8b 45 f4        MOV        EAX,dword ptr [EBP + curSectionVirtualAddr]
00401216 03 45 f8        ADD        EAX,dword ptr [EBP + i]
00401219 0f be 08        MOVSX      ECX,byte ptr [EAX]
0040121c 83 f9 45        CMP        ECX,0x45
0040121f 75 43           JNZ        LAB_00401264
00401221 c7 45 e8        MOV        dword ptr [EBP + jj],0x0
         00 00 00 00
00401228 eb 09           JMP        LAB_00401233

                     LAB_0040122a                          XREF[1]:     00401260(j)
0040122a 8b 55 e8        MOV        EDX,dword ptr [EBP + jj]
0040122d 83 c2 01        ADD        EDX,0x1
00401230 89 55 e8        MOV        dword ptr [EBP + jj],EDX

                     LAB_00401233                          XREF[1]:     00401228(j)
00401233 8b 45 f4        MOV        EAX,dword ptr [EBP + curSectionVirtualAddr]
00401236 03 45 f8        ADD        EAX,dword ptr [EBP + i]
00401239 8b 4d e8        MOV        ECX,dword ptr [EBP + jj]
0040123c 0f be 14 08     MOVSX      EDX,byte ptr [EAX + ECX*0x1]
00401240 85 d2           TEST       EDX,EDX
00401242 74 1e           JZ         LAB_00401262
00401244 8b 45 f4        MOV        EAX,dword ptr [EBP + curSectionVirtualAddr]
00401247 03 45 f8        ADD        EAX,dword ptr [EBP + i]
0040124a 8b 4d e8        MOV        ECX,dword ptr [EBP + jj]
0040124d 0f be 14 08     MOVSX      EDX,byte ptr [EAX + ECX*0x1]
00401251 03 55 08        ADD        EDX,dword ptr [EBP + seed]
00401254 8b 45 f4        MOV        EAX,dword ptr [EBP + curSectionVirtualAddr]
00401257 03 45 f8        ADD        EAX,dword ptr [EBP + i]
0040125a 8b 4d e8        MOV        ECX,dword ptr [EBP + jj]
0040125d 88 14 08        MOV        byte ptr [EAX + ECX*0x1],DL          data[i + jj] += seed
00401260 eb c8           JMP        LAB_0040122a                         while (data[i + jj] != 0)
```

```
for (i = 0; i < size; i++) {
    if (data[i] == 0x45) {
        for (jj = 0; data[i + jj] != '\0'; jj++)
            data[i + jj] += seed;
        break;
    }
}
```

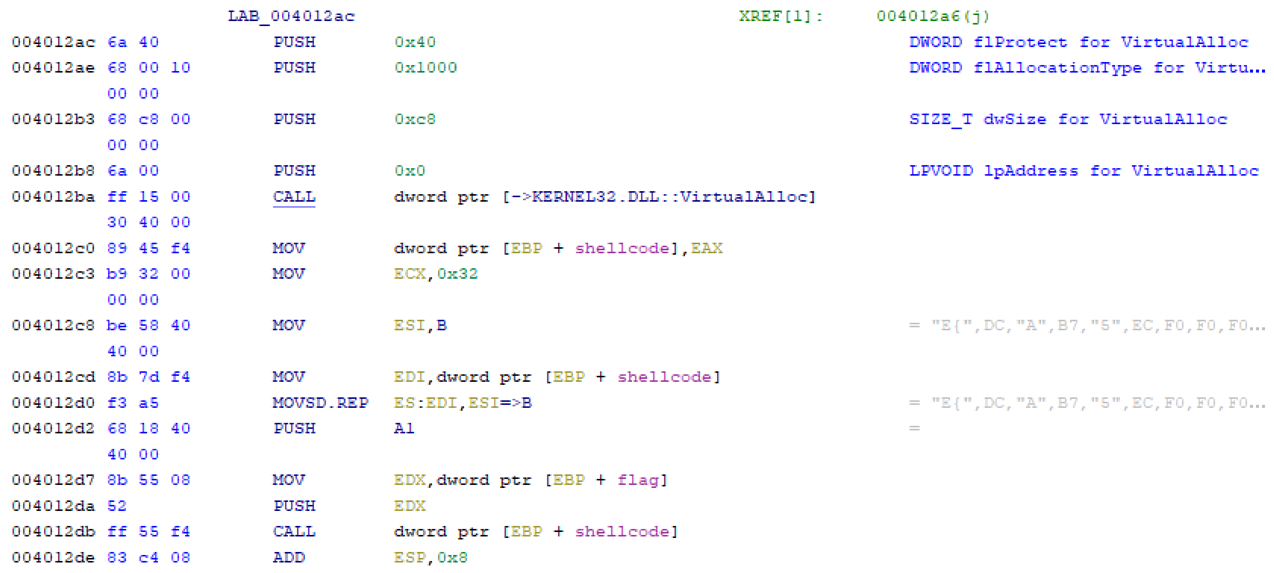
check_flag(flag)

1. calculate the length of the flag, return 0 if the length is not 32



```
                        check_flag                                    XREF[1]:     main:0040137f(c)
  00401270 55              PUSH       EBP
  00401271 8b ec           MOV        EBP,ESP
  00401273 83 ec 14        SUB        ESP,0x14
  00401276 56              PUSH       ESI
  00401277 57              PUSH       EDI
  00401278 8b 45 08        MOV        EAX,dword ptr [EBP + flag]
  0040127b 89 45 f8        MOV        dword ptr [EBP + flag_tail],EAX
  0040127e 8b 4d f8        MOV        ECX,dword ptr [EBP + flag_tail]
  00401281 83 c1 01        ADD        ECX,0x1
  00401284 89 4d f0        MOV        dword ptr [EBP + flag_head],ECX

                        LAB_00401287                                  XREF[1]:     00401297(j)
  00401287 8b 55 f8        MOV        EDX,dword ptr [EBP + flag_tail]
  0040128a 8a 02           MOV        AL,byte ptr [EDX]
  0040128c 88 45 ff        MOV        byte ptr [EBP + chr],AL
  0040128f 83 45 f8 01     ADD        dword ptr [EBP + flag_tail],0x1
  00401293 80 7d ff 00     CMP        byte ptr [EBP + chr],0x0
  00401297 75 ee           JNZ        LAB_00401287
  00401299 8b 4d f8        MOV        ECX,dword ptr [EBP + flag_tail]
  0040129c 2b 4d f0        SUB        ECX,dword ptr [EBP + flag_head]
  0040129f 89 4d ec        MOV        dword ptr [EBP + flag_len],ECX
  004012a2 83 7d ec 20     CMP        dword ptr [EBP + flag_len],0x20          flag_len == 32
  004012a6 74 04           JZ         LAB_004012ac
  004012a8 33 c0           XOR        EAX,EAX
  004012aa eb 35           JMP        LAB_004012e1
```

2. allocate a memory in heap to store `B` (secret array mentioned before), let’s say `shellcode`



```
                        LAB_004012ac                                XREF[1]:     004012a6(j)
  004012ac 6a 40           PUSH       0x40                                     DWORD flProtect for VirtualAlloc
  004012ae 68 00 10        PUSH       0x1000                                   DWORD flAllocationType for Virtu…
           00 00
  004012b3 68 c8 00        PUSH       0xc8                                     SIZE_T dwSize for VirtualAlloc
           00 00
  004012b8 6a 00           PUSH       0x0                                      LPVOID lpAddress for VirtualAlloc
  004012ba ff 15 00        CALL       dword ptr [->KERNEL32.DLL::VirtualAlloc]
           30 40 00
  004012c0 89 45 f4        MOV        dword ptr [EBP + shellcode],EAX
  004012c3 b9 32 00        MOV        ECX,0x32
           00 00
  004012c8 be 58 40        MOV        ESI,B                                    = "E{",DC,"A",B7,"5",EC,F0,F0,F0…
           40 00
  004012cd 8b 7d f4        MOV        EDI,dword ptr [EBP + shellcode]
  004012d0 f3 a5           MOVSD.REP  ES:EDI,ESI=>B                            = "E{",DC,"A",B7,"5",EC,F0,F0,F0…
  004012d2 68 18 40        PUSH       A1                                       =
           40 00
  004012d7 8b 55 08        MOV        EDX,dword ptr [EBP + flag]
  004012da 52              PUSH       EDX
  004012db ff 55 f4        CALL       dword ptr [EBP + shellcode]
  004012de 83 c4 08        ADD        ESP,0x8
```

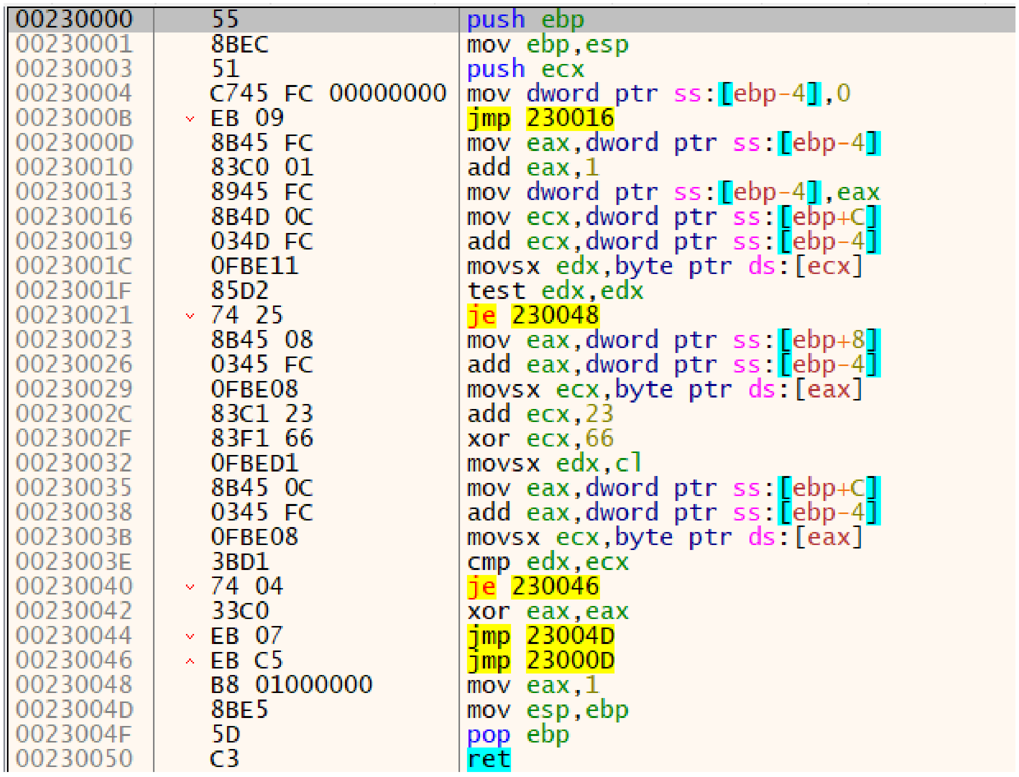
3. call `shellcode` as a function, `flag` and `A` (secret array) as parameter
   - `shellcode(flag, A)`

figure out what seed is

- as a function, shellcode should start with `push ebp` (machine code: 0x55)
- `B` starts with 0x45
- so we guess the seed is: 16

disassembled shellcode

- disassemble shellcode obtained by decrypting `B` with `seed`



```
00230000   55               push ebp
00230001   8BEC             mov ebp,esp
00230003   51               push ecx
00230004   C745 FC 00000000 mov dword ptr ss:[ebp-4],0
0023000B   EB 09            jmp 230016
0023000D   8B45 FC          mov eax,dword ptr ss:[ebp-4]
00230010   83C0 01          add eax,1
00230013   8945 FC          mov dword ptr ss:[ebp-4],eax
00230016   8B4D 0C          mov ecx,dword ptr ss:[ebp+C]
00230019   034D FC          add ecx,dword ptr ss:[ebp-4]
0023001C   0FBE11           movsx edx,byte ptr ds:[ecx]
0023001F   85D2             test edx,edx
00230021   74 25            je 230048
00230023   8B45 08          mov eax,dword ptr ss:[ebp+8]
00230026   0345 FC          add eax,dword ptr ss:[ebp-4]
00230029   0FBE08           movsx ecx,byte ptr ds:[eax]
0023002C   83C1 23          add ecx,23
0023002F   83F1 66          xor ecx,66
00230032   0FBED1           movsx edx,cl
00230035   8B45 0C          mov eax,dword ptr ss:[ebp+C]
00230038   0345 FC          add eax,dword ptr ss:[ebp-4]
0023003B   0FBE08           movsx ecx,byte ptr ds:[eax]
0023003E   3BD1             cmp edx,ecx
00230040   74 04            je 230046
00230042   33C0             xor eax,eax
00230044   EB 07            jmp 23004D
00230046   EB C5            jmp 23000D
00230048   B8 01000000      mov eax,1
0023004D   8BE5             mov esp,ebp
0023004F   5D               pop ebp
00230050   C3               ret
```

- stack / parameter:
  - [ebp+8]: flag
  - [ebp+C]: A
  - [ebp-4]: i
- equivalent C code

```
for (int i = 0; A[i] != '\0'; i++)
    if (A[i] != (flag[i] + 0x23) ^ 0x66)
        return 0;
return 1;
```

recover flag from secret A

- see `sol.py`