

CSE 434S

Lab: Debugging

Overview

The goal of this lab is to practice debugging. Debugging assembly is a skill that requires continuous practice; the more you practice, the more you become familiar with assembly blocks, and with different debugging techniques.

You are free to choose whatever debugging tool you like. You can try a couple of different tools to see which one works best for you.

Before you start looking at the details, I want to remind you some of the rules of reverse engineering by revisiting some of the things we discussed in the first lecture:

- Don't get caught in details!
- You don't need to understand 100% of the code
- Focus on key features

Please analyze the first two samples in the lab's archive, which you can download from Canvas, and answer the following questions regarding the samples. You may want to use other tools in addition to a debugger, such as our basic static and dynamic analysis tools and IDA, to analyze the samples, and that is encouraged.

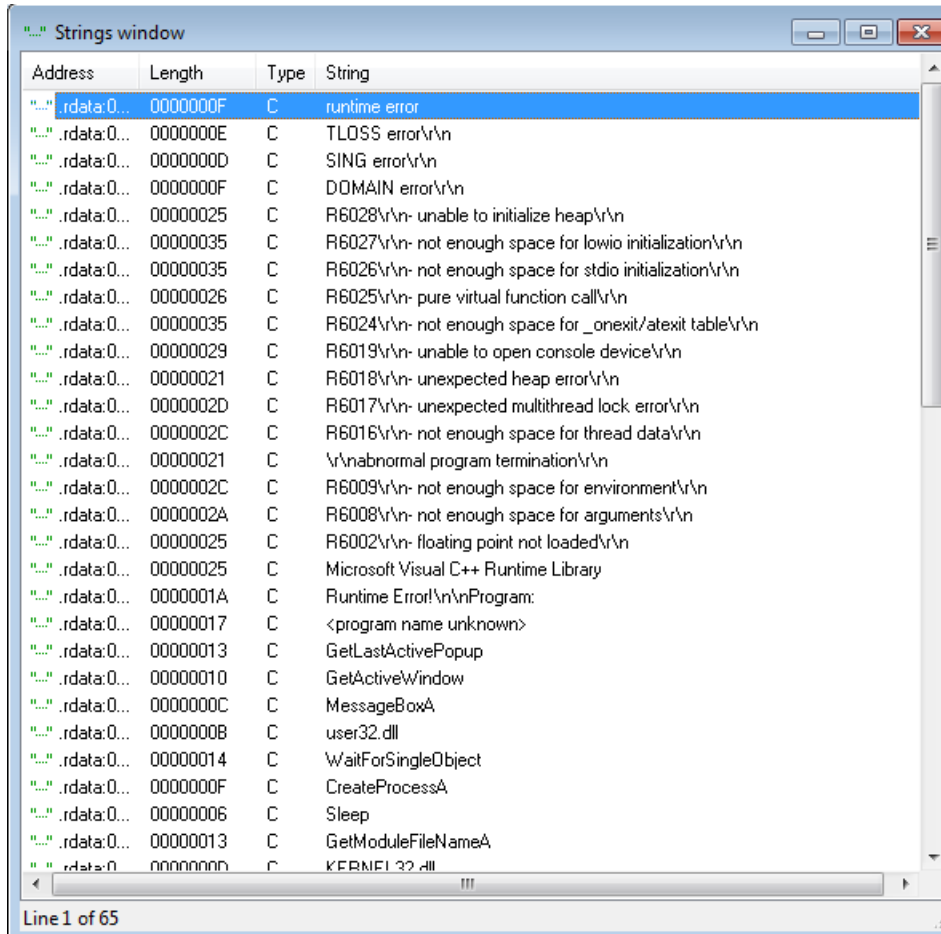
***** Remember to snapshot your VM and double-check that it is on an isolated network (e.g. an "internal network" in VirtualBox) before loading any sample in a debugger! *****

CSE 434S

Sample 2:

Q2-1. What strings do you see statically in the binary?

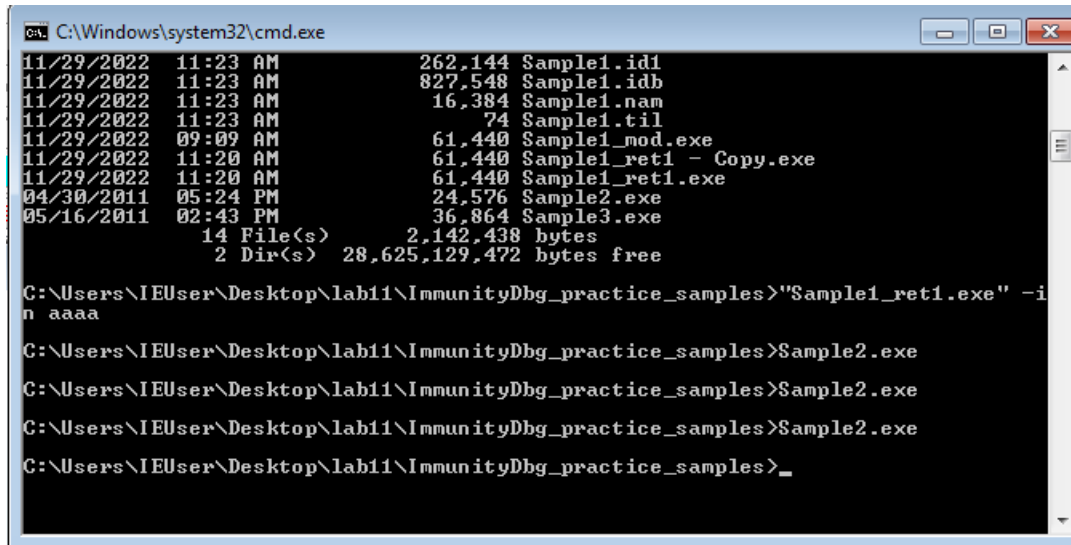
In IDA 'Strings window', I can see dozens of strings.



CSE 434S

Q2-2. What happens when you run this binary?

It just terminated.



```
C:\Windows\system32\cmd.exe
11/29/2022 11:23 AM      262,144 Sample1.id1
11/29/2022 11:23 AM      827,548 Sample1.idb
11/29/2022 11:23 AM       16,384 Sample1.nam
11/29/2022 11:23 AM         74 Sample1.til
11/29/2022 09:09 AM       61,440 Sample1_mod.exe
11/29/2022 11:20 AM       61,440 Sample1_ret1 - Copy.exe
11/29/2022 11:20 AM       61,440 Sample1_ret1.exe
04/30/2011 05:24 PM       24,576 Sample2.exe
05/16/2011 02:43 PM       36,864 Sample3.exe
14 File(s)      2,142,438 bytes
2 Dir(s)  28,625,129,472 bytes free

C:\Users\IEUser\Desktop\lab11\ImmunityDbg_practice_samples>"Sample1_ret1.exe" -i
n aaaa

C:\Users\IEUser\Desktop\lab11\ImmunityDbg_practice_samples>Sample2.exe

C:\Users\IEUser\Desktop\lab11\ImmunityDbg_practice_samples>Sample2.exe

C:\Users\IEUser\Desktop\lab11\ImmunityDbg_practice_samples>Sample2.exe

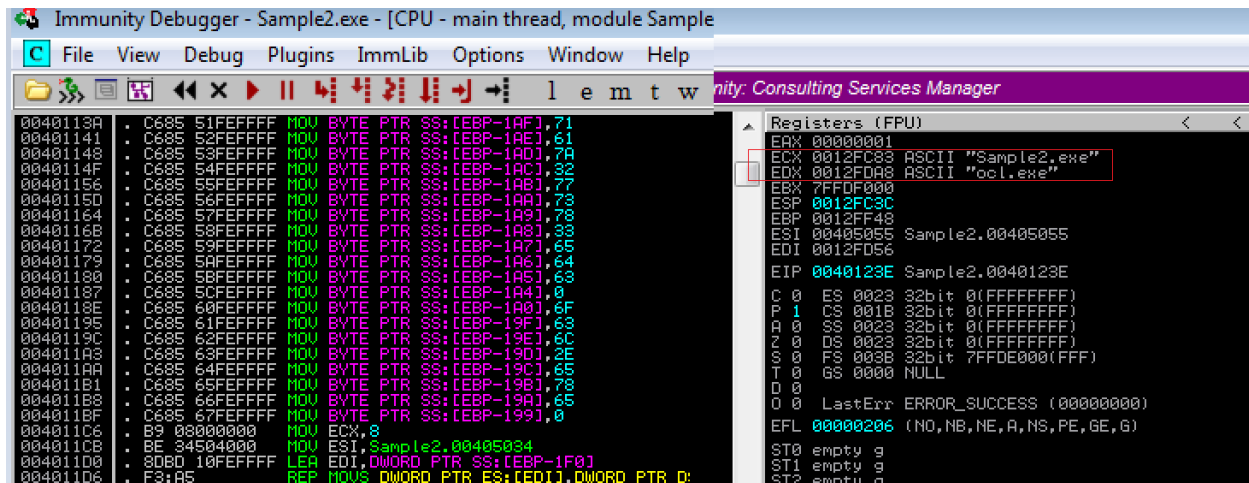
C:\Users\IEUser\Desktop\lab11\ImmunityDbg_practice_samples>_
```

CSE 434S

Q2-3. How can you get this sample to run its malicious payload?

You need to rename the file to 'ocl.exe' for it to run properly.

We can see it is preparing executing program name(Sample2.exe) and the comparing name(ocl.exe)

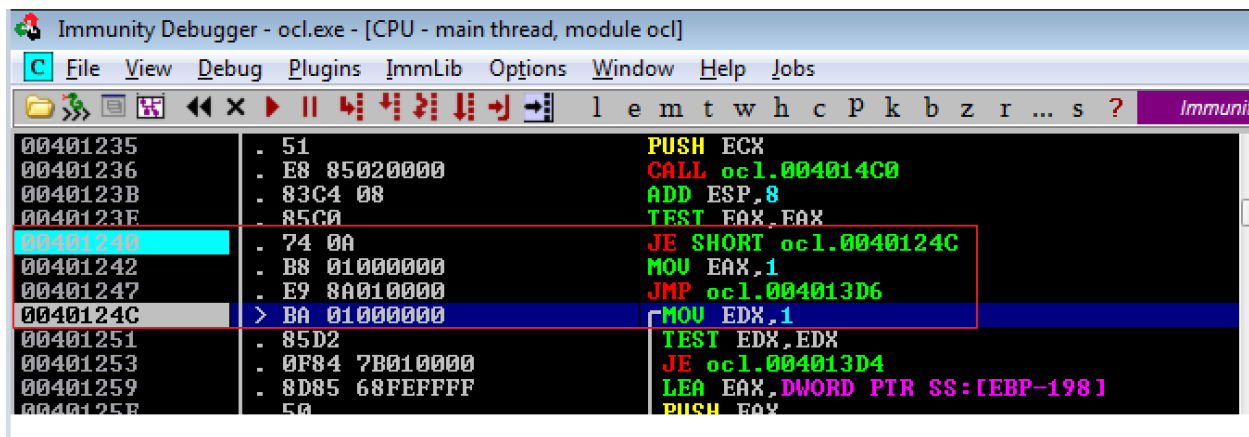


The screenshot shows the Immunity Debugger interface for 'Sample2.exe'. The assembly window on the left displays a loop of instructions from address 0040113A to 004011D6, all of which are 'MOV BYTE PTR SS:[EBP-1XX], 71' instructions. The registers window on the right shows the current state of the CPU registers. A red box highlights the ECX register, which contains the ASCII string 'Sample2.exe', and the EDI register, which contains the ASCII string 'ocl.exe'. The EIP register is at 0040123E.

```
Immunity Debugger - Sample2.exe - [CPU - main thread, module Sample2.exe]
File View Debug Plugins ImmLib Options Window Help
l e m t w h c P k b z r ... s ?
0040113A: C685 51FFFFFF MOV BYTE PTR SS:[EBP-1AF], 71
00401141: C685 52FFFFFF MOV BYTE PTR SS:[EBP-1AE], 61
00401148: C685 53FFFFFF MOV BYTE PTR SS:[EBP-1AD], 7A
0040114F: C685 54FFFFFF MOV BYTE PTR SS:[EBP-1AC], 32
00401156: C685 55FFFFFF MOV BYTE PTR SS:[EBP-1AB], 77
0040115D: C685 56FFFFFF MOV BYTE PTR SS:[EBP-1AA], 73
00401164: C685 57FFFFFF MOV BYTE PTR SS:[EBP-1A9], 78
0040116B: C685 58FFFFFF MOV BYTE PTR SS:[EBP-1A8], 33
00401172: C685 59FFFFFF MOV BYTE PTR SS:[EBP-1A7], 65
00401179: C685 5AFFFFFF MOV BYTE PTR SS:[EBP-1A6], 64
00401180: C685 5BFFFFFF MOV BYTE PTR SS:[EBP-1A5], 63
00401187: C685 5CFFFFFF MOV BYTE PTR SS:[EBP-1A4], 0
0040118E: C685 5DFFFFFF MOV BYTE PTR SS:[EBP-1A3], 6F
00401195: C685 5EFFFFFF MOV BYTE PTR SS:[EBP-1A2], 63
0040119C: C685 5FFFFFFF MOV BYTE PTR SS:[EBP-1A1], 6C
004011A3: C685 60FFFFFF MOV BYTE PTR SS:[EBP-1A0], 2E
004011AA: C685 61FFFFFF MOV BYTE PTR SS:[EBP-19F], 65
004011B1: C685 62FFFFFF MOV BYTE PTR SS:[EBP-19E], 6C
004011B8: C685 63FFFFFF MOV BYTE PTR SS:[EBP-19D], 2E
004011BF: C685 64FFFFFF MOV BYTE PTR SS:[EBP-19C], 65
004011C6: C685 65FFFFFF MOV BYTE PTR SS:[EBP-19B], 78
004011CD: C685 66FFFFFF MOV BYTE PTR SS:[EBP-19A], 65
004011D6: C685 67FFFFFF MOV BYTE PTR SS:[EBP-199], 0
B9 08000000 MOV ECX, 8
BE 34504000 MOV ESI, Sample2.00405034
8DB0 10FFFFFF LEA EDI, DWORD PTR SS:[EBP-1F0]
F3: A5 REP MOVSD, DWORD PTR ES:[EDI], DWORD PTR DS:[EAX]

Registers (FPU)
EAX 00000001
ECX 0012FC83 ASCII "Sample2.exe"
EDI 0012FD98 ASCII "ocl.exe"
EBX 7FFDF000
ESP 0012FC3C
EBP 0012FF48
ESI 00405055 Sample2.00405055
EDI 0012FD56
EIP 0040123E Sample2.0040123E
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000206 (NO, NB, NE, A, NS, PE, GE, G)
ST0 empty g
ST1 empty g
ST2 empty g
```

After changing the filename into ocl.exe, I can move to the address of 0040124C, not falling into 004013D6 which is termination of the program.(You can notice the white rectangle has moved to the 0040124C)



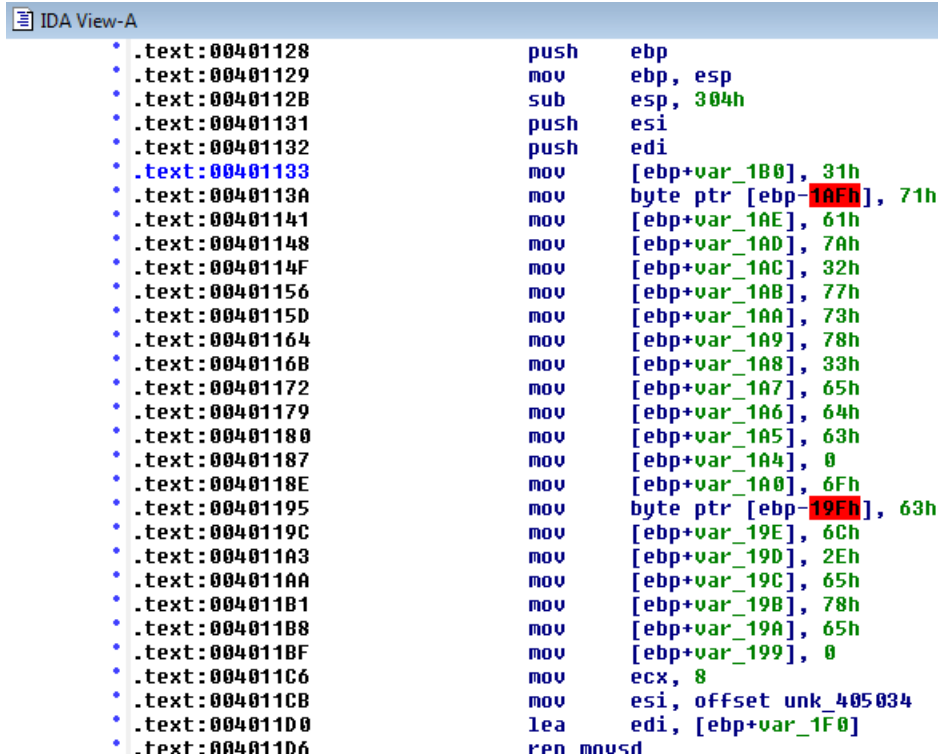
The screenshot shows the Immunity Debugger interface for 'ocl.exe'. The assembly window displays instructions from address 00401235 to 0040125F. A red box highlights the instruction at address 0040124C: 'MOV EDI, 1'. The instruction at 00401247 is 'JMP ocl.004013D6', which is the termination point. The instruction at 00401240 is 'JE SHORT ocl.0040124C', which branches to the highlighted instruction.

```
Immunity Debugger - ocl.exe - [CPU - main thread, module ocl]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
00401235: 51 PUSH ECX
00401236: E8 85020000 CALL ocl.004014C0
0040123B: 83C4 08 ADD ESP, 8
0040123F: 85C0 TEST EAX, EAX
00401240: 74 0A JE SHORT ocl.0040124C
00401242: B8 01000000 MOV EAX, 1
00401247: E9 8A010000 JMP ocl.004013D6
0040124C: BA 01000000 MOV EDI, 1
00401251: 85D2 TEST EDX, EDX
00401253: 0F84 7B010000 JE ocl.004013D4
00401259: 8DB5 68FFFFFF LEA EAX, DWORD PTR SS:[EBP-198]
0040125F: 50 PUSH EAX
```

CSE 434S

Q2-4. What is happening at 0x00401133?

It is moving the ASCII code into memory. Maybe the malware didn't want to show the password or anything per se.



```
.text:00401128      push     ebp
.text:00401129      mov      ebp, esp
.text:0040112B      sub      esp, 304h
.text:00401131      push     esi
.text:00401132      push     edi
.text:00401133      mov      [ebp+var_1B0], 31h
.text:0040113A      mov      byte ptr [ebp-1AFh], 71h
.text:00401141      mov      [ebp+var_1AE], 61h
.text:00401148      mov      [ebp+var_1AD], 7Ah
.text:0040114F      mov      [ebp+var_1AC], 32h
.text:00401156      mov      [ebp+var_1AB], 77h
.text:0040115D      mov      [ebp+var_1AA], 73h
.text:00401164      mov      [ebp+var_1A9], 78h
.text:0040116B      mov      [ebp+var_1A8], 33h
.text:00401172      mov      [ebp+var_1A7], 65h
.text:00401179      mov      [ebp+var_1A6], 64h
.text:00401180      mov      [ebp+var_1A5], 63h
.text:00401187      mov      [ebp+var_1A4], 0
.text:0040118E      mov      [ebp+var_1A0], 6Fh
.text:00401195      mov      byte ptr [ebp-19Fh], 63h
.text:0040119C      mov      [ebp+var_19E], 6Ch
.text:004011A3      mov      [ebp+var_19D], 2Eh
.text:004011AA      mov      [ebp+var_19C], 65h
.text:004011B1      mov      [ebp+var_19B], 78h
.text:004011B8      mov      [ebp+var_19A], 65h
.text:004011BF      mov      [ebp+var_199], 0
.text:004011C6      mov      ecx, 8
.text:004011CB      mov      esi, offset unk_405034
.text:004011D0      lea      edi, [ebp+var_1F0]
.text:004011D6      rep movsd
```

We can also check the ASCII string in Immunity Debugger by check the content of the EBP – 1B0 address! Click right mouse button above the 'EBP' on the 'Registers' window, and choose 'Follow in Dump'. And then, the content of the address will be shown in the 'Hex dump' windows on the left bottom of the Immunity Debugger.

The strings are '1qaz2wsx3edc' and 'ocl.exe'.

CSE 434S

00401128 55 PUSH EBP
 00401129 8BEC MOV EBP,ESP
 0040112B 81EC SUB ESP,304
 00401131 56 PUSHESI
 00401132 57 PUSHEDI
 00401133 C685 50FFFFFF 31 MOV BYTE PTR SS:[EBP-1801.31]
 0040113A C685 51FFFFFF 71 MOV BYTE PTR SS:[EBP-1AF1.71]
 00401141 C685 52FFFFFF 61 MOV BYTE PTR SS:[EBP-1AE1.61]
 00401148 C685 53FFFFFF 7A MOV BYTE PTR SS:[EBP-1AD1.7A]
 0040114F C685 54FFFFFF 32 MOV BYTE PTR SS:[EBP-1AC1.32]
 00401156 C685 55FFFFFF 77 MOV BYTE PTR SS:[EBP-1AB1.77]
 0040115D C685 56FFFFFF 73 MOV BYTE PTR SS:[EBP-1AA1.73]
 00401164 C685 57FFFFFF 78 MOV BYTE PTR SS:[EBP-1A91.78]
 0040116B C685 58FFFFFF 33 MOV BYTE PTR SS:[EBP-1A81.33]
 00401172 C685 59FFFFFF 65 MOV BYTE PTR SS:[EBP-1A71.65]
 00401179 C685 5AFFFFFFF 64 MOV BYTE PTR SS:[EBP-1A61.64]
 00401180 C685 5BFFFFFF 63 MOV BYTE PTR SS:[EBP-1A51.63]
 00401187 C685 5CFFFFFF 00 MOV BYTE PTR SS:[EBP-1A41.0]
 0040118E C685 60FFFFFF 6F MOV BYTE PTR SS:[EBP-1A01.6F]
 00401195 C685 61FFFFFF 63 MOV BYTE PTR SS:[EBP-19F1.63]
 0040119C C685 62FFFFFF 6C MOV BYTE PTR SS:[EBP-19E1.6C]
 004011A3 C685 63FFFFFF 2E MOV BYTE PTR SS:[EBP-19D1.2E]
 004011AA C685 64FFFFFF 65 MOV BYTE PTR SS:[EBP-19C1.65]
 004011B1 C685 65FFFFFF 78 MOV BYTE PTR SS:[EBP-19B1.78]
 004011B8 C685 66FFFFFF 65 MOV BYTE PTR SS:[EBP-19A1.65]
 004011BF C685 67FFFFFF 00 MOV BYTE PTR SS:[EBP-1991.0]
 004011C6 B9 00000000 MOV ECK,8
 004011CB BE 34504000 MOVESI,oc1.00405034
 004011D0 8DBD 10FFFFFF LEA EDI,DWORD PTR SS:[EBP-1F0]
 004011D6 F3A5 REP MOVSDWORD PTR ES:[EDI],DWORD PTR DS:
 004011DB A4 MOVSDWORD PTR ES:[EDI],BYTE PTR DS:[ESI]
 004011D9 C785 48FFFFFF 00000000 MOV DWORD PTR SS:[EBP-1B81.0]
 004011E2 C685 68FFFFFF 00 MOV BYTE PTR SS:[EBP-1B71.0]

Registers (FPU)
 EAX 001E0E00
 ECX 001D48B0
 EDI 001D0168
 EBX 7FDD0000
 ESP 0012FC3C
 EBP 0012FF40
 ESI 00000000
 EDI 00000000
 EIP 004011BF ocl.004011BF
 C 0 ES 0023 32bit 0<FFFFFFFF>
 P 1 CS 001B 32bit 0<FFFFFFFF>
 A 0 SS 0023 32bit 0<FFFFFFFF>
 Z 0 DS 0023 32bit 0<FFFFFFFF>
 S 0 FS 003B 32bit 7FDF0000<FFF>
 T 0 GS 0000 NULL
 D 0
 O 0 LastErr ERROR_SUCCESS <00000000>
 EFL 00000206 <NO,NB,NE,A,NS,PE,GE,G>
 ST0 empty g
 ST1 empty g
 ST2 empty g
 ST3 empty g
 ST4 empty g
 ST5 empty g
 ST6 empty g
 ST7 empty g
 FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 <GT>
 FCW 027F Prec NEAR.53 Mask 1 1 1 1 1

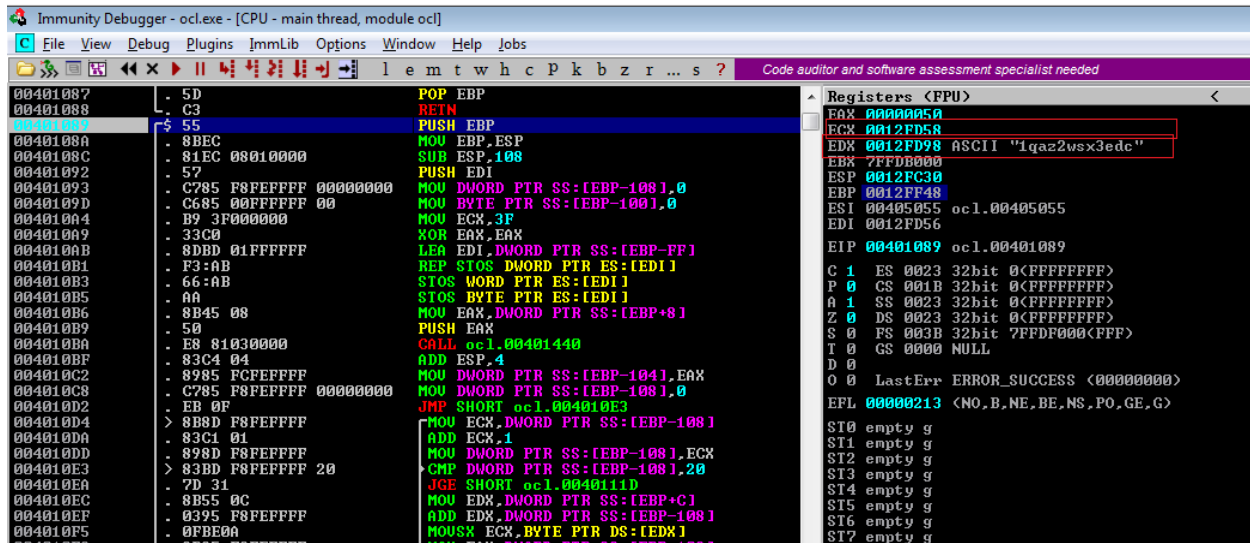
Address Hex dump ASCII
 0012FD38 00 00 00 00 00 00 00 00
 0012FD40 50 01 60 00 7F 00 00 00 PG.Δ...
 0012FD48 08 41 60 00 90 E0 00 00 0A'.Eα...
 0012FD50 00 00 00 00 DC 01 00 00@...
 0012FD58 38 01 1D 00 8C FD 00 80 00 8@+.i².0
 0012FD60 B0 48 1D 00 7F 00 00 00 H.Δ...
 0012FD68 C4 00 1D 00 E7 02 00 00 -.+.τ0...
 0012FD70 00 00 00 00 DC FD 00 00 00 ..².0...
 0012FD78 00 00 00 00 7F 00 00 00 ...Δ...
 0012FD80 70 E0 00 00 EC FC 12 6A 00 6A.Δ...
 0012FD88 FC E0 01 77 F4 FD 12 "αΔv²±...
 0012FD90 C0 FD 12 00 5E 60 03 12±.ΛVw...
 0012FD98 31 71 61 7A 32 77 73 1qaz2wsx...
 0012FDA0 33 65 64 63 00 48 1D 3edc.H+...
 0012FDA8 6F 63 6C 2F 65 78 65 oc1.exp...
 0012FDB0 00 00 1D 00 00 00 1D 00 ..+.+.+...
 0012FDB8 00 00 00 00 00 00 00 00
 0012FDC0 04 FE 12 00 99 6F 03 4it.0oWu...
 0012FDC8 00 00 1D 00 EC 6F 03 ..+.0oWu...
 0012FDD0 07 44 0E 77 00 00 1D 00 Dkw.+.+...
 0012FDD8 00 00 1D 00 00 00 00 00 ..+.+.+.+...
 0012FDE0 00 00 1D 00 7B A4 E0 ..+.<ñα0...
 0012FDE8 FE FF 00 01 D0 FD 12 00 00 00±...
 0012FDF0 C7 A4 FF 76 E8 FE 12 00 00 00±...
 0012FDF8 5D E1 F8 76 7B A4 E0 00 00 00±...
 0012FE00 FE FF FF FF EC 6F 03 00 00 00...
 0012FE08 C7 A4 FF 76 00 00 1D 00 00 00...
 0012FE10 00 00 00 00 00 00 00 00
 0012FE18 00 00 00 00 00 00 00 00
 0012FE20 00 00 00 00 00 00 00 00
 0012FE28 00 00 00 00 00 00 00 00
 0012FE30 00 00 00 00 00 00 00 00
 0012FE38 00 00 00 00 00 00 00 00
 0012FE40 00 00 00 00 00 00 00 00
 0012FE48 00 00 00 00 00 00 00 00
 0012FE50 00 00 00 00 00 00 00 00
 0012FE58 00 00 00 00 00 00 00 00
 0012FE60 00 00 00 00 00 00 00 00
 0012FE68 00 00 00 00 00 00 00 00
 0012FE70 00 00 00 00 00 00 00 00
 0012FE78 00 00 00 00 00 00 00 00
 0012FE80 00 00 00 00 00 00 00 00
 0012FE88 00 00 00 00 00 00 00 00
 0012FE90 00 00 00 00 00 00 00 00
 0012FE98 00 00 00 00 00 00 00 00
 0012FEA0 00 00 00 00 00 00 00 00
 0012FEA8 00 00 00 00 00 00 00 00
 0012FEB0 00 00 00 00 00 00 00 00
 0012FEB8 00 00 00 00 00 00 00 00
 0012FEC0 00 00 00 00 00 00 00 00
 0012FEC8 00 00 00 00 00 00 00 00
 0012FED0 00 00 00 00 00 00 00 00
 0012FED8 00 00 00 00 00 00 00 00
 0012FEE0 00 00 00 00 00 00 00 00
 0012FEE8 00 00 00 00 00 00 00 00
 0012FEF0 00 00 00 00 00 00 00 00
 0012FEF8 00 00 00 00 00 00 00 00
 0012FF00 00 00 00 00 00 00 00 00
 0012FF08 00 00 00 00 00 00 00 00
 0012FF10 00 00 00 00 00 00 00 00
 0012FF18 00 00 00 00 00 00 00 00
 0012FF20 00 00 00 00 00 00 00 00
 0012FF28 00 00 00 00 00 00 00 00
 0012FF30 00 00 00 00 00 00 00 00
 0012FF38 00 00 00 00 00 00 00 00
 0012FF40 00 00 00 00 00 00 00 00
 0012FF48 00 00 00 00 00 00 00 00
 0012FF50 00 00 00 00 00 00 00 00
 0012FF58 00 00 00 00 00 00 00 00
 0012FF60 00 00 00 00 00 00 00 00
 0012FF68 00 00 00 00 00 00 00 00
 0012FF70 00 00 00 00 00 00 00 00
 0012FF78 00 00 00 00 00 00 00 00
 0012FF80 00 00 00 00 00 00 00 00
 0012FF88 00 00 00 00 00 00 00 00
 0012FF90 00 00 00 00 00 00 00 00
 0012FF98 00 00 00 00 00 00 00 00
 0012FFA0 00 00 00 00 00 00 00 00
 0012FFA8 00 00 00 00 00 00 00 00
 0012FFB0 00 00 00 00 00 00 00 00
 0012FFB8 00 00 00 00 00 00 00 00
 0012FFC0 00 00 00 00 00 00 00 00
 0012FFC8 00 00 00 00 00 00 00 00
 0012FFD0 00 00 00 00 00 00 00 00
 0012FFD8 00 00 00 00 00 00 00 00
 0012FFE0 00 00 00 00 00 00 00 00
 0012FFE8 00 00 00 00 00 00 00 00
 0012FFF0 00 00 00 00 00 00 00 00
 0012FFF8 00 00 00 00 00 00 00 00
 00130000 00 00 00 00 00 00 00 00
 00130008 00 00 00 00 00 00 00 00
 00130010 00 00 00 00 00 00 00 00
 00130018 00 00 00 00 00 00 00 00
 00130020 00 00 00 00 00 00 00 00
 00130028 00 00 00 00 00 00 00 00
 00130030 00 00 00 00 00 00 00 00
 00130038 00 00 00 00 00 00 00 00
 00130040 00 00 00 00 00 00 00 00
 00130048 00 00 00 00 00 00 00 00
 00130050 00 00 00 00 00 00 00 00
 00130058 00 00 00 00 00 00 00 00
 00130060 00 00 00 00 00 00 00 00
 00130068 00 00 00 00 00 00 00 00
 00130070 00 00 00 00 00 00 00 00
 00130078 00 00 00 00 00 00 00 00
 00130080 00 00 00 00 00 00 00 00
 00130088 00 00 00 00 00 00 00 00
 00130090 00 00 00 00 00 00 00 00
 00130098 00 00 00 00 00 00 00 00
 001300A0 00 00 00 00 00 00 00 00
 001300A8 00 00 00 00 00 00 00 00
 001300B0 00 00 00 00 00 00 00 00
 001300B8 00 00 00 00 00 00 00 00
 001300C0 00 00 00 00 00 00 00 00
 001300C8 00 00 00 00 00 00 00 00
 001300D0 00 00 00 00 00 00 00 00
 001300D8 00 00 00 00 00 00 00 00
 001300E0 00 00 00 00 00 00 00 00
 001300E8 00 00 00 00 00 00 00 00
 001300F0 00 00 00 00 00 00 00 00
 001300F8 00 00 00 00 00 00 00 00
 00130100 00 00 00 00 00 00 00 00
 00130108 00 00 00 00 00 00 00 00
 00130110 00 00 00 00 00 00 00 00
 00130118 00 00 00 00 00 00 00 00
 00130120 00 00 00 00 00 00 00 00
 00130128 00 00 00 00 00 00 00 00
 00130130 00 00 00 00 00 00 00 00
 00130138 00 00 00 00 00 00 00 00
 00130140 00 00 00 00 00 00 00 00
 00130148 00 00 00 00 00 00 00 00
 00130150 00 00 00 00 00 00 00 00
 00130158 00 00 00 00 00 00 00 00
 00130160 00 00 00 00 00 00 00 00
 00130168 00 00 00 00 00 00 00 00
 00130170 00 00 00 00 00 00 00 00
 00130178 00 00 00 00 00 00 00 00
 00130180 00 00 00 00 00 00 00 00
 00130188 00 00 00 00 00 00 00 00
 00130190 00 00 00 00 00 00 00 00
 00130198 00 00 00 00 00 00 00 00
 001301A0 00 00 00 00 00 00 00 00
 001301A8 00 00 00 00 00 00 00 00
 001301B0 00 00 00 00 00 00 00 00
 001301B8 00 00 00 00 00 00 00 00
 001301C0 00 00 00 00 00 00 00 00
 001301C8 00 00 00 00 00 00 00 00
 001301D0 00 00 00 00 00 00 00 00
 001301D8 00 00 00 00 00 00 00 00
 001301E0 00 00 00 00 00 00 00 00
 001301E8 00 00 00 00 00 00 00 00
 001301F0 00 00 00 00 00 00 00 00
 001301F8 00 00 00 00 00 00 00 00
 00130200 00 00 00 00 00 00 00 00
 00130208 00 00 00 00 00 00 00 00
 00130210 00 00 00 00 00 00 00 00
 00130218 00 00 00 00 00 00 00 00
 00130220 00 00 00 00 00 00 00 00
 00130228 00 00 00 00 00 00 00 00
 00130230 00 00 00 00 00 00 00 00
 00130238 00 00 00 00 00 00 00 00
 00130240 00 00 00 00 00 00 00 00
 00130248 00 00 00 00 00 00 00 00
 00130250 00 00 00 00 00 00 00 00
 00130258 00 00 00 00 00 00 00 00
 00130260 00 00 00 00 00 00 00 00
 00130268 00 00 00 00 00 00 00 00
 00130270 00 00 00 00 00 00 00 00
 00130278 00 00 00 00 00 00 00 00
 00130280 00 00 00 00 00 00 00 00
 00130288 00 00 00 00 00 00 00 00
 00130290 00 00 00 00 00 00 00 00
 00130298 00 00 00 00 00 00 00 00
 001302A0 00 00 00 00 00 00 00 00
 001302A8 00 00 00 00 00 00 00 00
 001302B0 00 00 00 00 00 00 00 00
 001302B8 00 00 00 00 00 00 00 00
 001302C0 00 00 00 00 00 00 00 00
 001302C8 00 00 00 00 00 00 00 00
 001302D0 00 00 00 00 00 00 00 00
 001302D8 00 00 00 00 00 00 00 00
 001302E0 00 00 00 00 00 00 00 00
 001302E8 00 00 00 00 00 00 00 00
 001302F0 00 00 00 00 00 00 00 00
 001302F8 00 00 00 00 00 00 00 00
 00130300 00 00 00 00 00 00 00 00
 00130308 00 00 00 00 00 00 00 00
 00130310 00 00 00 00 00 00 00 00
 00130318 00 00 00 00 00 00 00 00
 00130320 00 00 00 00 00 00 00 00
 00130328 00 00 00 00 00 00 00 00
 00130330 00 00 00 00 00 00 00 00
 00130338 00 00 00 00 00 00 00 00
 00130340 00 00 00 00 00 00 00 00
 00130348 00 00 00 00 00 00 00 00
 00130350 00 00 00 00 00 00 00 00
 00130358 00 00 00 00 00 00 00 00
 00130360 00 00 00 00 00 00 00 00
 00130368 00 00 00 00 00 00 00 00
 00130370 00 00 00 00 00 00 00 00
 00130378 00 00 00 00 00 00 00 00
 00130380 00 00 00 00 00 00 00 00
 00130388 00 00 00 00 00 00 00 00
 00130390 00 00 00 00 00 00 00 00
 00130398 00 00 00 00 00 00 00 00
 001303A0 00 00 00 00 00 00 00 00
 001303A8 00 00 00 00 00 00 00 00
 001303B0 00 00 00 00 00 00 00 00
 001303B8 00 00 00 00 00 00 00 00
 001303C0 00 00 00 00 00 00 00 00
 001303C8 00 00 00 00 00 00 00 00
 001303D0 00 00 00 00 00 00 00 00
 001303D8 00 00 00 00 00 00 00 00
 001303E0 00 00 00 00 00 00 00 00
 001303E8 00 00 00 00 00 00 00 00
 001303F0 00 00 00 00 00 00 00 00
 001303F8 00 00 00 00 00 00 00 00
 00130400 00 00 00 00 00 00 00 00
 00130408 00 00 00 00 00 00 00 00
 00130410 00 00 00 00 00 00 00 00
 00130418 00 00 00 00 00 00 00 00
 00130420 00 00 00 00 00 00 00 00
 00130428 00 00 00 00 00 00 00 00
 00130430 00 00 00 00 00 00 00 00
 00130438 00 00 00 00 00 00 00 00
 00130440 00 00 00 00 00 00 00 00
 00130448 00 00 00 00 00 00 00 00
 00130450 00 00 00 00 00 00 00 00
 00130458 00 00 00 00 00 00 00 00
 00130460 00 00 00 00 00 00 00 00
 00130468 00 00 00 00 00 00 00 00
 00130470 00 00 00 00 00 00 00 00
 00130478 00 00 00 00 00 00 00 00
 00130480 00 00 00 00 00 00 00 00
 00130488 00 00 00 00 00 00 00 00
 00130490 00 00 00 00 00 00 00 00
 00130498 00 00 00 00 00 00 00 00
 001304A0 00 00 00 00 00 00 00 00
 001304A8 00 00 00 00 00 00 00 00
 001304B0 00 00 00 00 00 00 00 00
 001304B8 00 00 00 00 00 00 00 00
 001304C0 00 00 00 00 00 00 00 00
 001304C8 00 00 00 00 00 00 00 00
 001304D0 00 00 00 00 00 00 00 00
 001304D8 00 00 00 00 00 00 00 00
 001304E0 00 00 00 00 00 00 00 00
 001304E8 00 00 00 00 00 00 00 00
 001304F0 00 00 00 00 00 00 00 00
 001304F8 00 00 00 00 00 00 00 00
 00130500 00 00 00 00 00 00 00 00
 00130508 00 00 00 00 00 00 00 00
 00130510 00 00 00 00 00 00 00 00
 00130518 00 00 00 00 00 00 00 00
 00130520 00 00 00 00 00 00 00 00
 00130528 00 00 00 00 00 00 00 00
 00130530 00 00 00 00 00 00 00 00
 00130538 00 00 00 00 00 00 00 00
 00130540 00 00 00 00 00 00 00 00
 00130548 00 00 00 00 00 00 00 00
 00130550 00 00 00 00 00 00 00 00
 00130558 00 00 00 00 00 00 00 00
 00130560 00 00 00 00 00 00 00 00
 00130568 00 00 00 00 00 00 00 00
 00130570 00 00 00 00 00 00 00 00
 00130578 00 00 00 00 00 00 00 00
 00130580 00 00 00 00 00 00 00 00
 00130588 00 00 00 00 00 00 00 00
 00130590 00 00 00 00 00 00 00 00
 00130598 00 00 00 00 00 00 00 00
 001305A0 00 00 00 00 00 00 00 00
 001305A8 00 00 00 00 00 00 00 00
 001305B0 00 00 00 00 00 00 00 00
 001305B8 00 00 00 00 00 00 00 00
 001305C0 00 00 00 00 00 00 00

CSE 434S

Q2-5. What arguments are being passed to subroutine 0x00401089?

The two arguments being passed to this function. One is '0012FD58' and the other is a string '1qaz2wsx3edc'. First parameter is in ECX and the other is in EDX.

You can set a break point at 401089 and run the debug. It will stop at break point and we can see parameter in the register window.



The screenshot shows the Immunity Debugger interface. The assembly window displays the following code starting at address 00401087:

```
00401087 5D POP EBP
00401088 C3 RETN
00401089 55 PUSH EBP
0040108A 8BEC MOV EBP,ESP
0040108C 81EC 08010000 SUB ESP,108
00401092 57 PUSH EDI
00401093 C785 F8FEFFFF 00000000 MOV DWORD PTR SS:[EBP-108],0
0040109D C685 00FFFFFF 00 MOV BYTE PTR SS:[EBP-100],0
004010A4 B9 3F000000 MOV ECX,3F
004010A9 33C0 XOR EAX,EAX
004010AB 8DBD 01FFFFFF LEA EDI,DWORD PTR SS:[EBP-FF]
004010B1 F3:AB REP STOS DWORD PTR ES:[EDI]
004010B3 66:AB STOS WORD PTR ES:[EDI]
004010B5 AA STOS BYTE PTR ES:[EDI]
004010B6 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
004010B9 50 PUSH EAX
004010BA E8 81030000 CALL ocl.00401440
004010BF 83C4 04 ADD ESP,4
004010C2 8985 FCFEFFFF MOV DWORD PTR SS:[EBP-104],EAX
004010C8 C785 F8FEFFFF 00000000 MOV DWORD PTR SS:[EBP-108],0
004010D2 EB 0F JMP SHORT ocl.004010E3
004010D4 8B8D F8FEFFFF MOV ECX,DWORD PTR SS:[EBP-108]
004010DA 83C1 01 ADD ECX,1
004010DD 898D F8FEFFFF MOV DWORD PTR SS:[EBP-108],ECX
004010E3 83BD F8FEFFFF 20 CMP DWORD PTR SS:[EBP-108],20
004010EA 7D 31 JGE SHORT ocl.0040111D
004010EC 8B55 0C MOV EDX,DWORD PTR SS:[EBP+C]
004010EF 8395 F8FEFFFF ADD EDX,DWORD PTR SS:[EBP-108]
004010F5 0FBE0A MOVSX ECX,BYTE PTR DS:[EDX]
```

The registers window on the right shows the following values:

Register	Value
EAX	00000050
ECX	0012FD58
EDX	0012FD98 ASCII "1qaz2wsx3edc"
EBX	7FFDF000
ESP	0012FC30
EBP	0012FF48
ESI	00405055 ocl.00405055
EDI	0012FD56
EIP	00401089 ocl.00401089

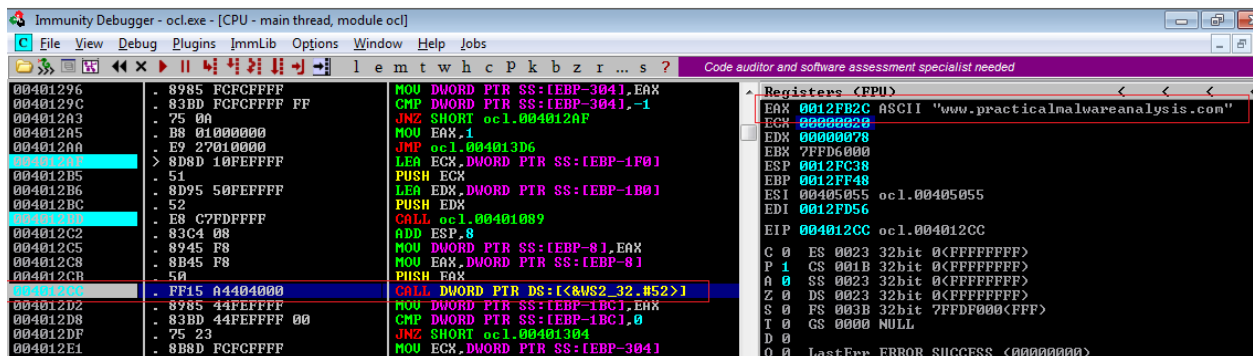
CSE 434S

Q2-6. What domain name does this malware use?

www.practicalmalwareanalysis.com

I tried to find host name something on the code and I found it using IDA. So, I thought that if I set a break point before 'gethostname' and then I will get a host name. So I set the break point at '4012CC' and run Immunity debugger and it shows the host name below.

```
.text:004012AF loc_4012AF: ; CODE XREF: _main+17BTj
.text:004012AF lea ecx, [ebp+var_1F0]
.text:004012B5 push ecx ; int
.text:004012B6 lea edx, [ebp+var_1B0]
.text:004012BC push edx ; char *
.text:004012BD call sub_401089
.text:004012C2 add esp, 8
.text:004012C5 mov [ebp+name], eax
.text:004012C8 mov eax, [ebp+name]
.text:004012CB push eax ; name
.text:004012CC call ds:gethostname
.text:004012D2 mov [ebp+var_1BC], eax
.text:004012D8 cmp [ebp+var_1BC], 0
.text:004012DF jnz short loc_401304
.text:004012E1 mov ecx, [ebp+s]
.text:004012E7 push ecx ; s
.text:004012E8 call ds:closesocket
.text:004012EE call ds:WSACleanup
.text:004012F4 push 7530h ; dwMilliseconds
.text:004012F9 call ds:Sleep
.text:004012FF jmp loc_40124C
```




```
Registers (FPU) < < < <
EAX 0012FB2C ASCII "www.practicalmalwareanalysis.com"
ECX 00000020
EDX 00000078
EBX 7FFDF000
ESP 0012FC3C
EBP 0012FF48
ESI 00405055 ocl.00405055
EDI 0012FD56
EIP 004012C5 ocl.004012C5
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
```

CSE 434S

Q2-7. What encoding routine is being used to obfuscate the domain name?

I think XOR is used to obfuscate the domain name with the string 1qaz2wsx3edc.

You can find this part on the function before 'gethostname'. If you go to 'sub_401089', you can find that it will move to '40110C' which XORing with '1qaz2wsx3edc'.

```
.text:004010C2      mov     [ebp+var_104], eax
.text:004010C8      mov     [ebp+var_108], 0
.text:004010D2      jmp     short loc_4010E3
.text:004010D4      ; -----
.text:004010D4      loc_4010D4:      ; CODE XREF: sub_401089+92↓j
.text:004010D4      mov     ecx, [ebp+var_108]
.text:004010DA      add     ecx, 1
.text:004010DD      mov     [ebp+var_108], ecx
.text:004010E3      loc_4010E3:      ; CODE XREF: sub_401089+49↑j
.text:004010E3      cmp     [ebp+var_108], 20h
.text:004010EA      jge     short loc_40111D
.text:004010EC      mov     edx, [ebp+arg_4]
.text:004010EF      add     edx, [ebp+var_108]
.text:004010F5      movsx   ecx, byte ptr [edx]
.text:004010F8      mov     eax, [ebp+var_108]
.text:004010FE      cdq
.text:004010FF      idiv    [ebp+var_104]
.text:00401105      mov     eax, [ebp+arg_0]
.text:00401108      movsx   edx, byte ptr [eax+edx]
.text:0040110C      xor     ecx, edx
.text:0040110E      mov     eax, [ebp+var_108]
.text:00401114      mov     byte ptr [ebp+eax+var_100], cl
.text:0040111B      jmp     short loc_4010D4
```

The screenshot shows a debugger window with the following components:

- Assembly View:** Displays assembly code for the function 'sub_401089'. The code includes instructions like `MOV EAX, 3F`, `XOR EAX, EAX`, `LEA EDI, DWORD PTR SS:[EBP-FF]`, `REP STOS DWORD PTR ES:[EDI]`, `STOS WORD PTR ES:[EDI]`, `STOS BYTE PTR ES:[EDI]`, `Mov EAX, DWORD PTR SS:[EBP+8]`, `PUSH EAX`, `CALL ocl.00401440`, `ADD ESP, 4`, `MOV DWORD PTR SS:[EBP-104], EAX`, `MOV DWORD PTR SS:[EBP-108], 0`, `JMP SHORT ocl.004010E3`, `MOV ECX, DWORD PTR SS:[EBP-108]`, `ADD ECX, 1`, `MOV DWORD PTR SS:[EBP-108], ECX`, `CMP DWORD PTR SS:[EBP-108], 20`, `JGE SHORT ocl.0040111D`, `MOV EDI, DWORD PTR SS:[EBP+8]`, `ADD EDI, DWORD PTR SS:[EBP-108]`, `Movsx ECX, BYTE PTR DS:[EDI]`, `MOV EAX, DWORD PTR SS:[EBP-108]`, `CDQ`, `IDIV DWORD PTR SS:[EBP-104]`, `MOV EAX, DWORD PTR SS:[EBP+8]`, `Movsx EDI, BYTE PTR DS:[EAX+EDI]`, `XOR ECX, EDI`, `MOV EAX, DWORD PTR SS:[EBP-108]`, `MOV BYTE PTR SS:[EBP+EAX-100], CL`, `JMP SHORT ocl.004010D4`, `LEA EAX, DWORD PTR SS:[EBP-100]`, `POP EDI`, and `MOV EBP, EBP`.
- Registers (FPU):** Shows the state of the registers. The EAX register is highlighted and contains the ASCII string '1qaz2wsx3edc'. Other registers like ECX, EDX, EBX, ESP, EBP, ESI, and EDI are also shown.

CSE 434S

Q2-8. What is the significance of the CreateProcessA call at 0x0040106E?

From the IDA, I assume that it is trying to make a process doing some Standard input and output thing. And I can see a 'cmd' command as well. Therefore, my guess is that the malware is making a process which can shell execution.

```
.text:0040101D      mov     [ebp+StartupInfo.cb], 44h
.text:00401024      push    10h                ; size_t
.text:00401026      push    0                  ; int
.text:00401028      lea     ecx, [ebp+hHandle]
.text:0040102B      push    ecx                ; void *
.text:0040102C      call    _memset
.text:00401031      add     esp, 0Ch
.text:00401034      mov     [ebp+StartupInfo.dwFlags], 101h
.text:0040103B      mov     [ebp+StartupInfo.wShowWindow], 0
.text:00401041      mov     edx, [ebp+arg_10]
.text:00401044      mov     [ebp+StartupInfo.hStdInput], edx
.text:00401047      mov     eax, [ebp+StartupInfo.hStdInput]
.text:0040104A      mov     [ebp+StartupInfo.hStdError], eax
.text:0040104D      mov     ecx, [ebp+StartupInfo.hStdError]
.text:00401050      mov     [ebp+StartupInfo.hStdOutput], ecx
.text:00401053      lea     edx, [ebp+hHandle]
.text:00401056      push    edx                ; lpProcessInformation
.text:00401057      lea     eax, [ebp+StartupInfo]
.text:0040105A      push    eax                ; lpStartupInfo
.text:0040105B      push    0                  ; lpCurrentDirectory
.text:0040105D      push    0                  ; lpEnvironment
.text:0040105F      push    0                  ; dwCreationFlags
.text:00401061      push    1                  ; bInheritHandles
.text:00401063      push    0                  ; lpThreadAttributes
.text:00401065      push    0                  ; lpProcessAttributes
.text:00401067      push    offset CommandLine ; "cmd"
.text:0040106C      push    0                  ; lpApplicationName
.text:0040106E      call    ds:CreateProcessA
.text:00401074      mov     [ebp+var_14], eax
.text:00401077      push    0FFFFFFFFh        ; dwMilliseconds
.text:00401079      mov     ecx, [ebp+hHandle]
.text:0040107C      push    ecx                ; hHandle
.text:0040107D      call    ds:WaitForSingleObject
.text:00401083      xor     eax, eax
```