**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.

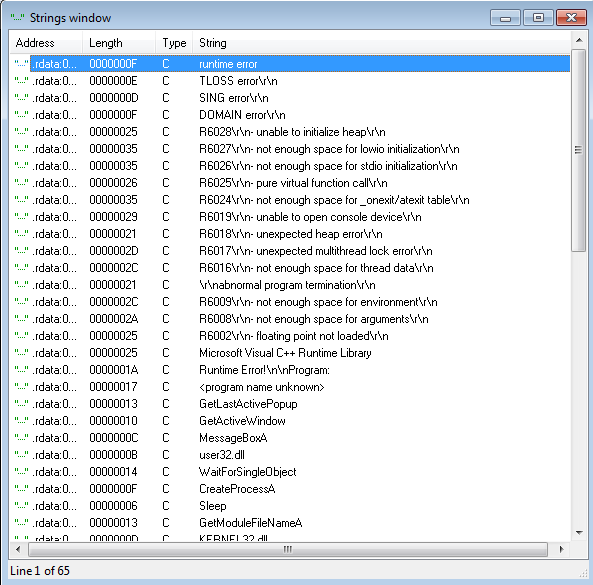
**\*\*\* Remember to snapshot your VM and double-check that it is on an isolated network (e.g. an “internal network” in VirtualBox) before loading any sample in a debugger! \*\*\***

**Sample 2:**

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

------------------------------------------------------------------------------

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

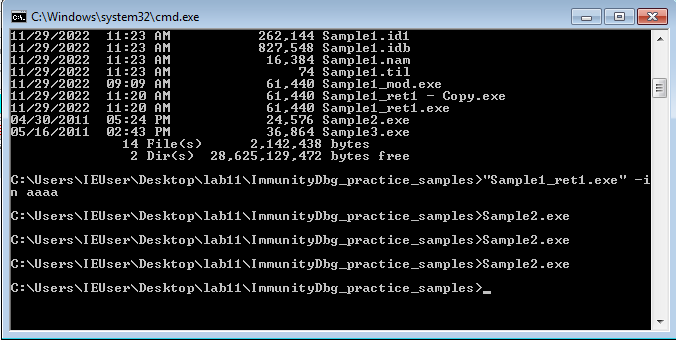


------------------------------------------------------------------------------

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

------------------------------------------------------------------------------

It just terminated.



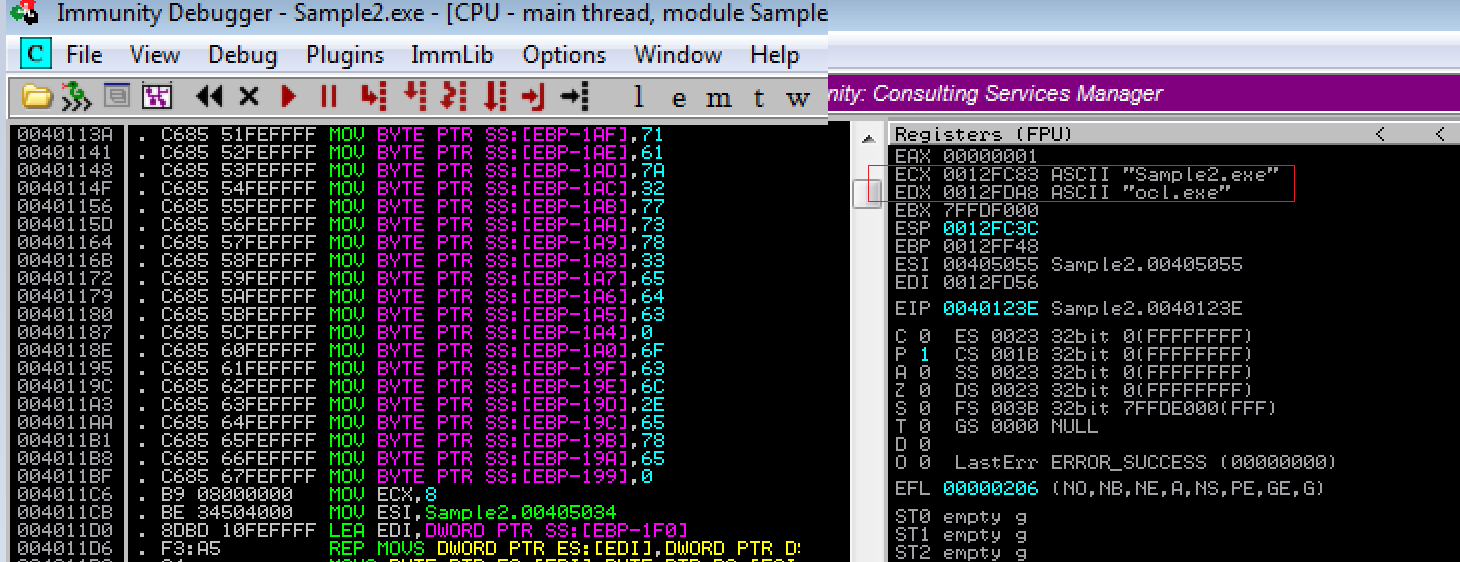
------------------------------------------------------------------------------

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

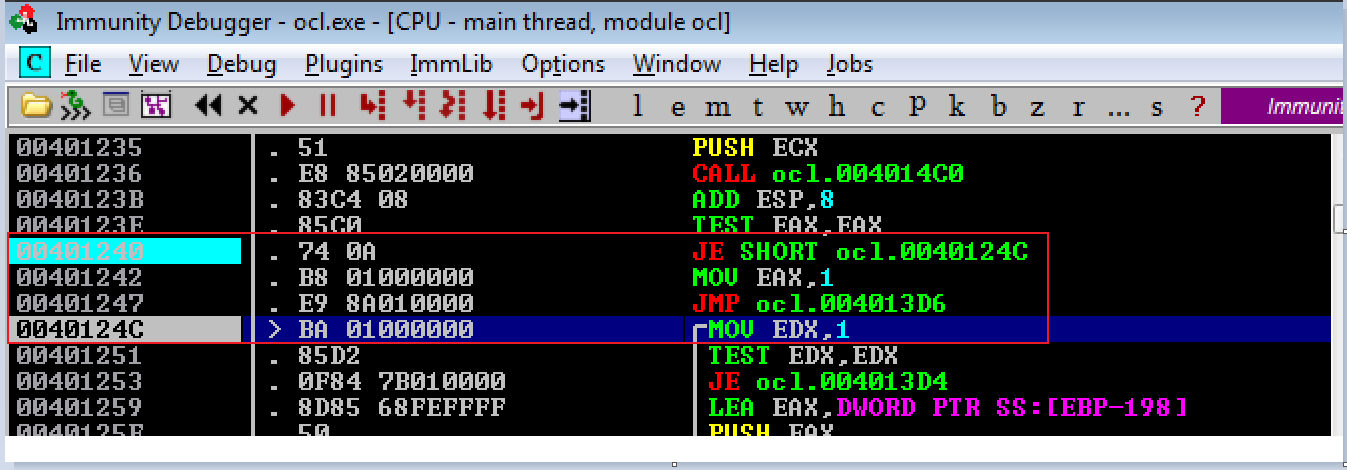
------------------------------------------------------------------------------

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)



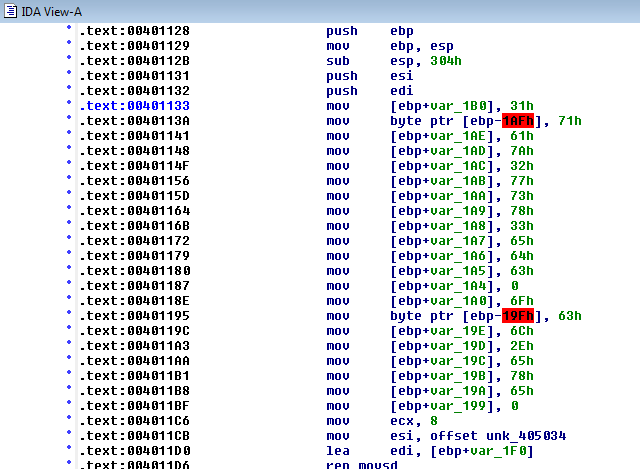
After changing the filename into ocl.exe, I can move to the address of 0040124C, not falling into 004013D6 which is termination of the program.(You can notice the white rectangle has moved to the 0040124C)



------------------------------------------------------------------------------

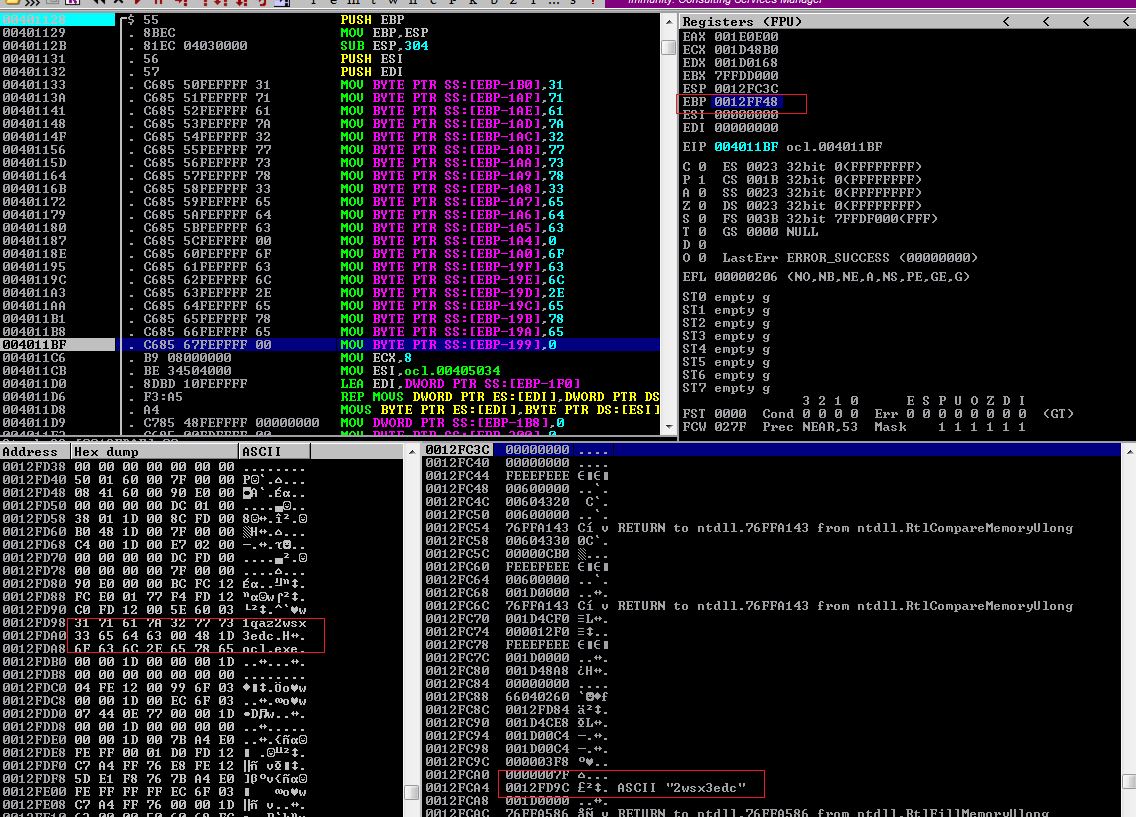
Q2-4. What is happening at 0x00401133?

------------------------------------------------------------------------------

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


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’.



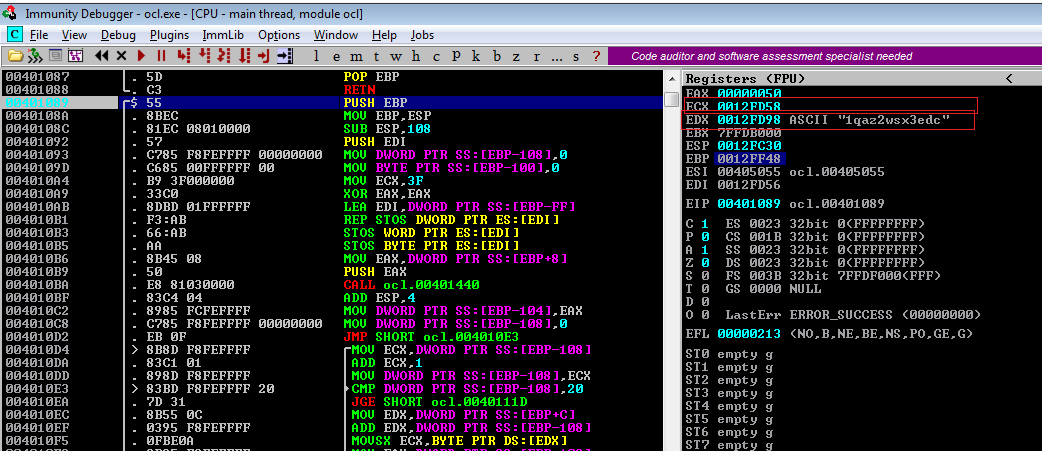
------------------------------------------------------------------------------

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 the see parameter in the register window.



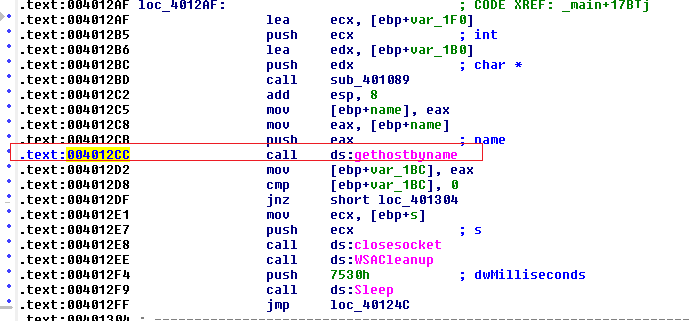
------------------------------------------------------------------------------

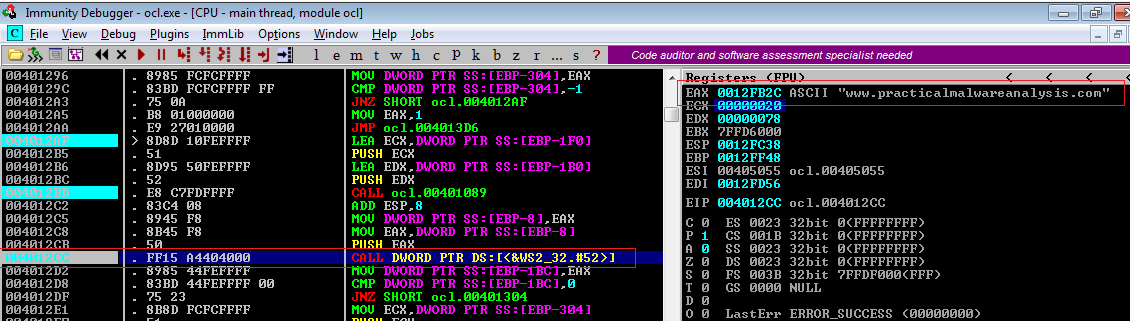
Q2-6. What domain name does this malware use?

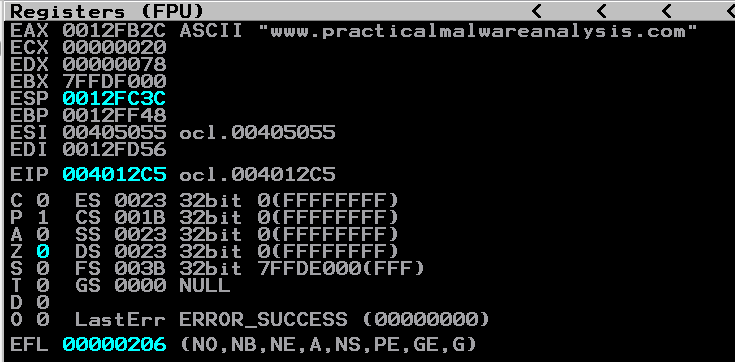
------------------------------------------------------------------------------

[www.practicalmalwareanalysis.com](http://www.practicalmalwareanalysis.com)

I tried to find host name something on the code and I found it using IDA. So, I though 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.







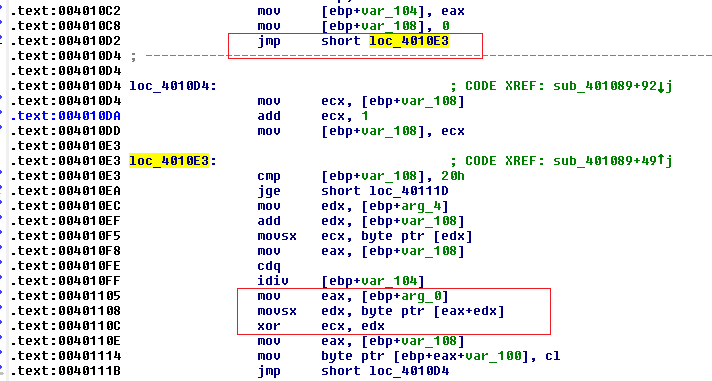
------------------------------------------------------------------------------

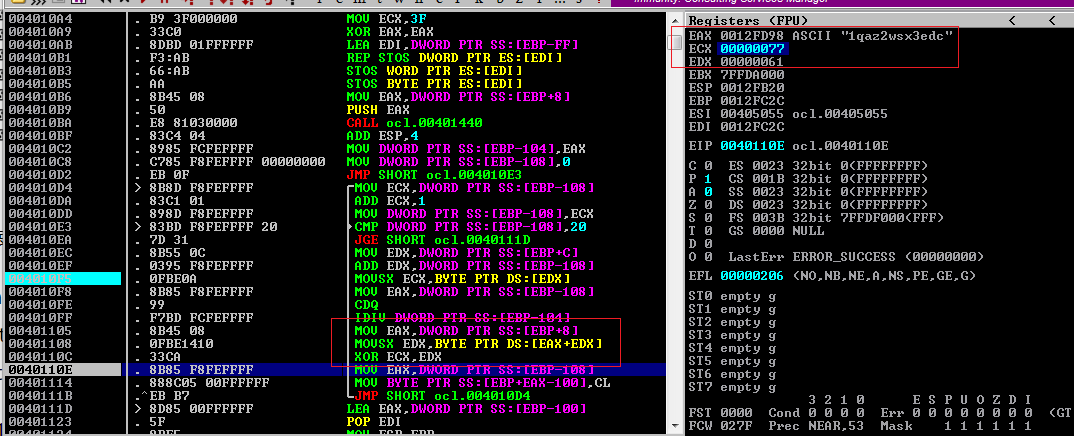
Q2-7. What encoding routine is being used to obfuscate the domain name?

------------------------------------------------------------------------------

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’.



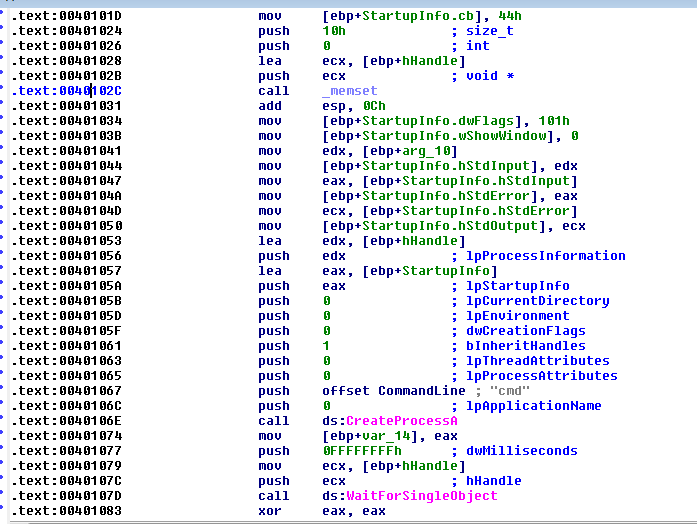


------------------------------------------------------------------------------

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.



------------------------------------------------------------------------------