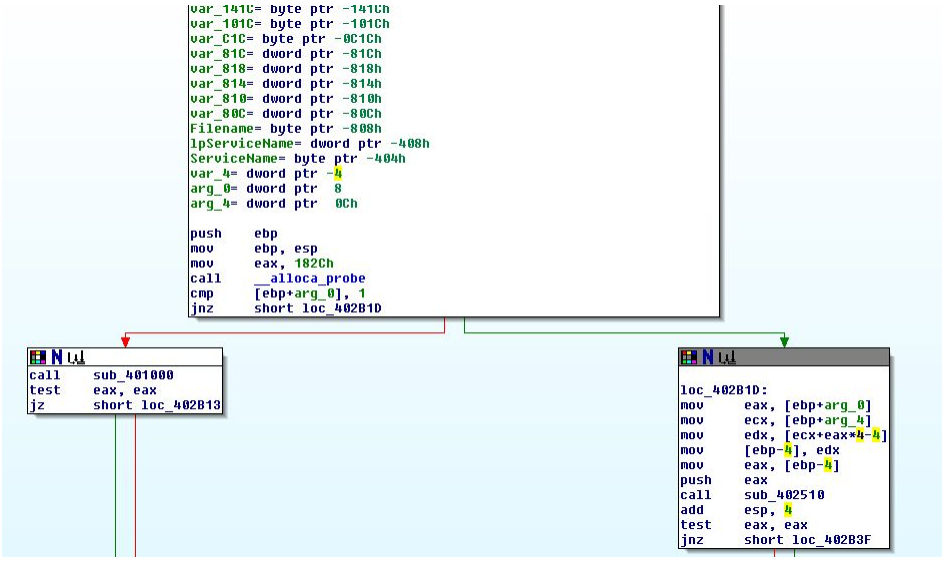
Song Yang (sy540)
Xin Yang (xy213)
Zhuohang Li (zl299)

Report of Homework 8

Lab09-01:

1. How can you get this malware to install itself?

In the main function, by reading the text in edx, it pushes the text to sub_402510.



In sub_402510 it compares the first character with 97 which is the ASCII code of ‘a’. Then add the code up to 100 so we get the password is abcd.

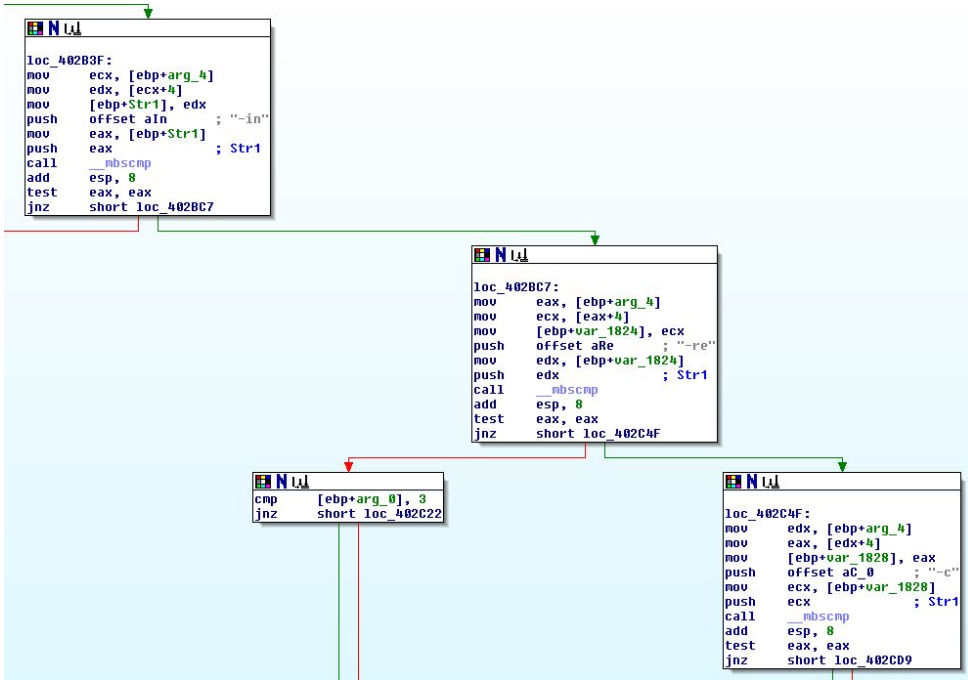
```
; Attributes: bp-based frame

sub_402510 proc near

var_4= byte ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    ecx
push    edi
mov     edi, [ebp+arg_0]
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
cmp     ecx, 4
jz      short loc_40252D
```

```
loc_40252D:
mov     eax, [ebp+8]
mov     cl, [eax]
mov     [ebp-4], cl
movsx   edx, byte ptr [ebp-4]
cmp     edx, 97
jz      short loc_402542
```

```
loc_402542:
mov     eax, [ebp+8]
mov     cl, [eax+1]
mov     [ebp-4], cl
mov     edx, [ebp+8]
mov     al, [ebp-4]
sub     al, [edx]
mov     [ebp-4], al
movsx   ecx, byte ptr [ebp-4]
cmp     ecx, 1
jz      short loc_402563
```

```
loc_402563:
mov     al, [ebp-4]
mov     dl, 63h
imul    dl
mov     [ebp-4], al
movsx   eax, byte ptr [ebp-4]
mov     ecx, [ebp+8]
movsx   edx, byte ptr [ecx+2]
cmp     eax, edx
jz      short loc_402580
```

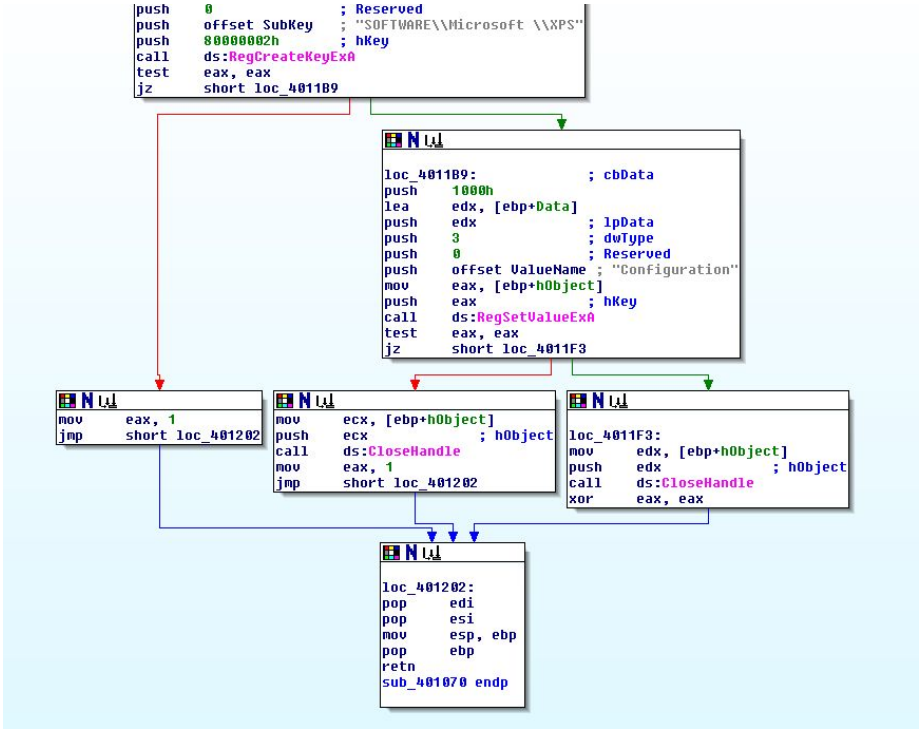
```
loc_402580:
mov     al, [ebp-4]
add     al, 1
mov     [ebp-4], al
movsx   ecx, byte ptr [ebp-4]
mov     edx, [ebp+8]
movsx   eax, byte ptr [edx+3]
cmp     ecx, eax
jz      short loc_40259B
```



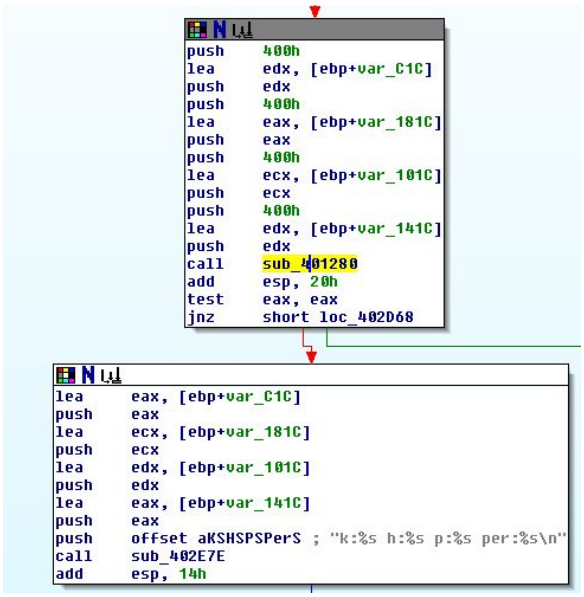
We get from the main function that the program requires some parameters to run. '-in' might be the command of 'install'. So as the order of the reading stack, the install command should firstly -in and then the password.

2. What are the command-line options for this program? What is the password requirement?

What are the command-line options for this program? What is the password requirement?
We got following strings in the main function, '-in', '-re', '-c', '-cc'
'-in' will go the left path to detect if the name of the service that is called to install exists or not. It may call sub_402410. The function includes GetModuleFileNameA, GetShortPathNameA and ShellExecuteA, to execute the function to 'cmd.exe'. The last path is to build a service whose name should contain in the command line.
'-re' would call sub_402900. The function can open and delete the service since it called OpenServiceA and DeleteService. We can regard is as 'Remove the Service'.
'-c' would either call sub_402410 and execute on cmd.exe or finally call sub_401070 to set the register configuration column since it calls RegSetValueExA.
Sub_401070:



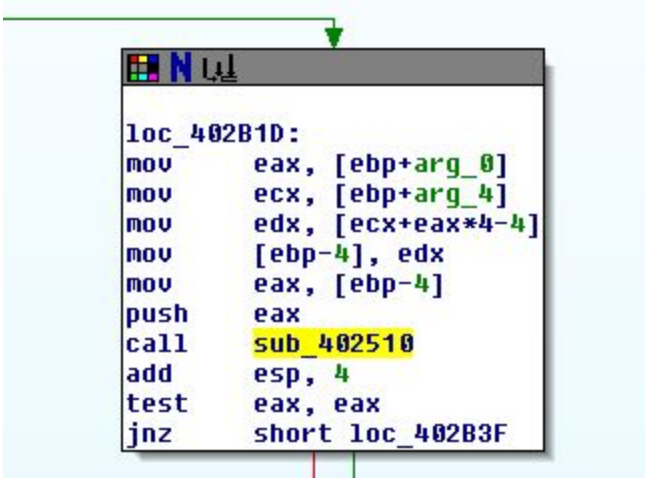
'-cc'. In '-cc' command, sub_401280 will be called to query the configuration of the register key and return the result for printing.

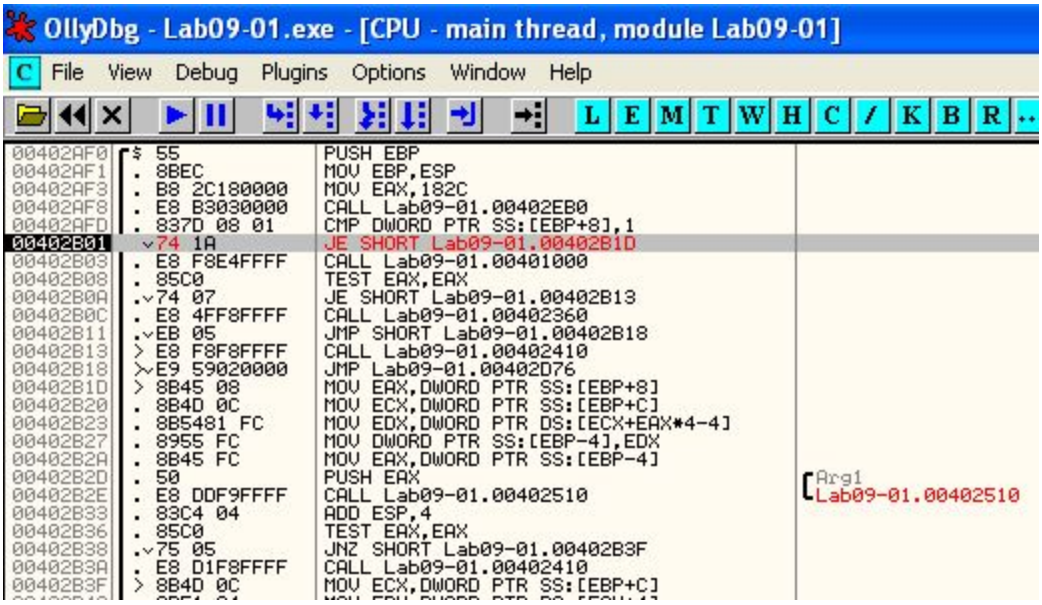


The password requirement is in sub_402510 which has been talked about in the last question, the password is 'abcd'.

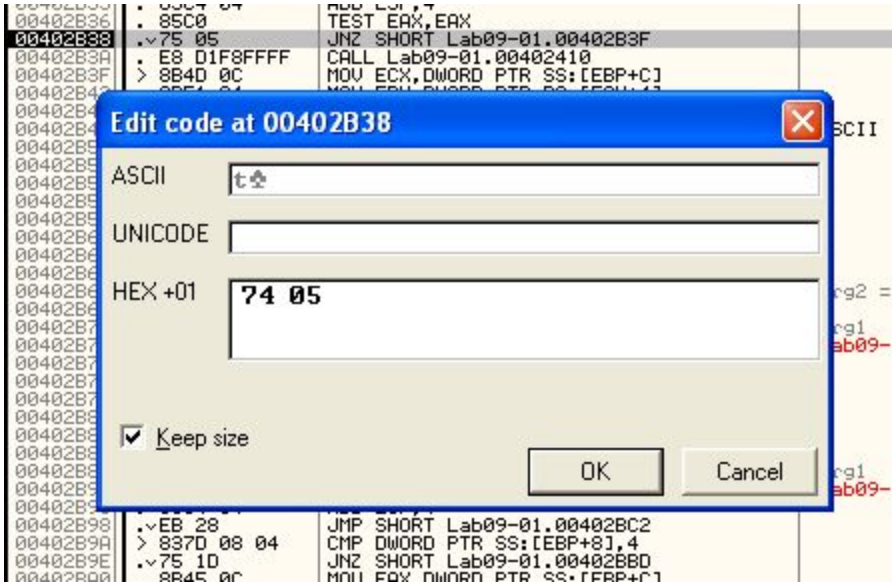
3. How can you use OllyDbg or IDA Pro to permanently patch this malware, so that it doesn't require the special command-line password?

The password input judgment located in the following function. Use OllyDbg on the line.



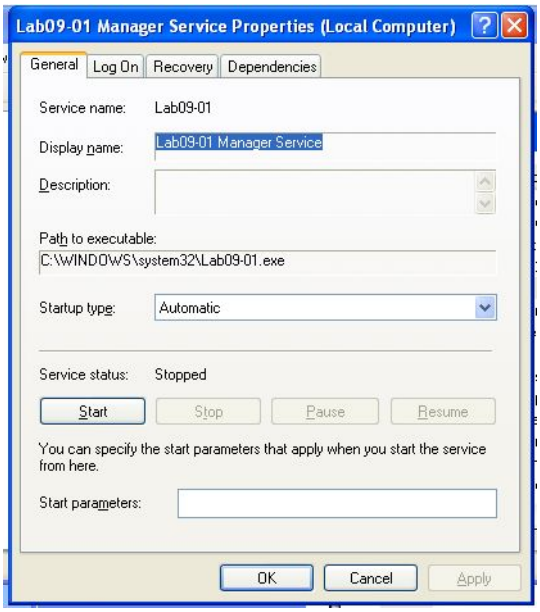


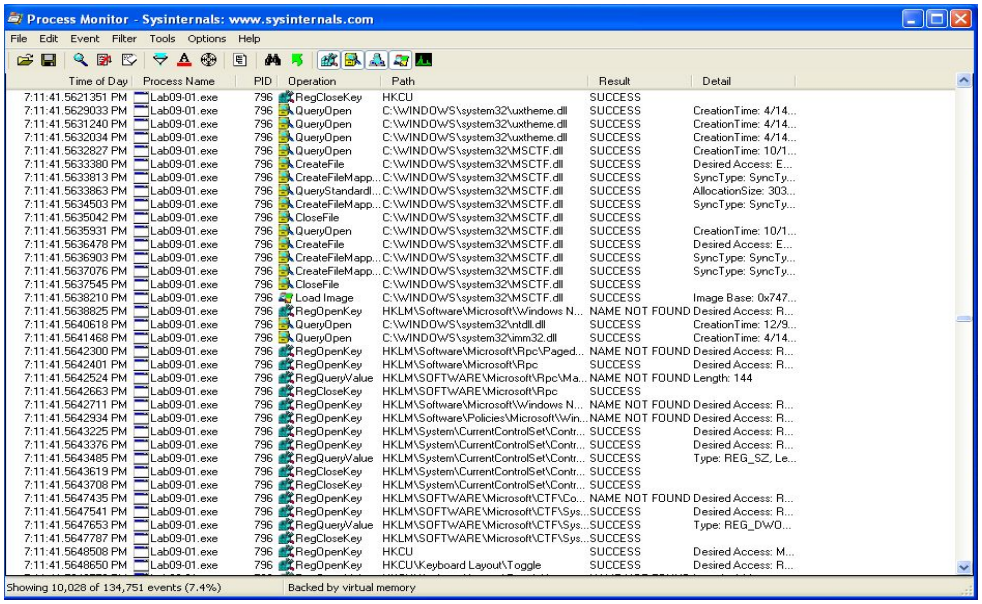
First, we need to pass the first detection located in 00402B01. Change the judgment word from JNZ to JZ. Then, the program goes to 00402B38, and do the same for this code. Finally, save the program, it now requires no command line to install.



4. What are the host-based indicators of this malware?

When analysis with IDA pro in the previous, we saw that the program would call register creating and modifying functions.
In order to verify, we open the regshot, process monitor, and got following results. The malware adds 6 changes to the register and installed a service named Lab09-01 Manager Service after running install command.





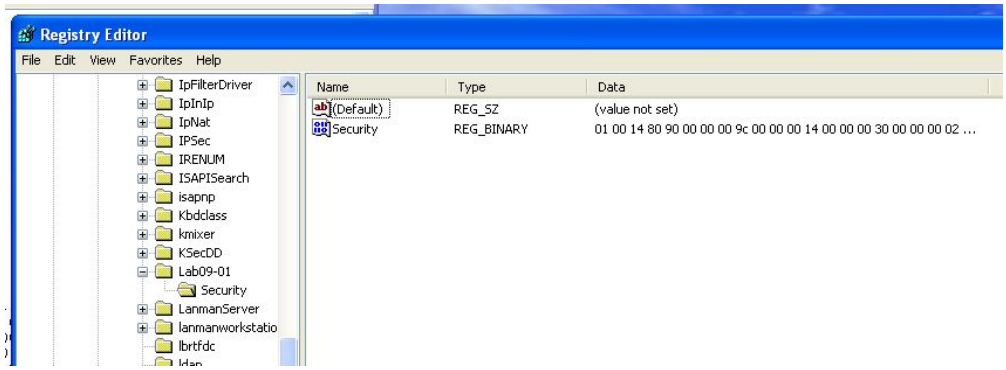
Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of Day	Process Name	PID	Operation	Path	Result	Detail
7:11:41.5621351 PM	Lab09-01.exe	796	RegCloseKey	HKCU	SUCCESS	
7:11:41.5629033 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\uxtheme.dll	SUCCESS	CreationTime: 4/14...
7:11:41.5631240 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\uxtheme.dll	SUCCESS	CreationTime: 4/14...
7:11:41.5632034 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\uxtheme.dll	SUCCESS	CreationTime: 4/14...
7:11:41.5632827 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	CreationTime: 10/1...
7:11:41.5633380 PM	Lab09-01.exe	796	CreateFile	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	Desired Access: E...
7:11:41.5633813 PM	Lab09-01.exe	796	CreateFileMapp...	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5633863 PM	Lab09-01.exe	796	QueryStandardI...	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	AllocationSize: 303...
7:11:41.5634503 PM	Lab09-01.exe	796	CreateFileMapp...	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5635042 PM	Lab09-01.exe	796	CloseFile	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	
7:11:41.5635931 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	CreationTime: 10/1...
7:11:41.5636478 PM	Lab09-01.exe	796	CreateFile	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	Desired Access: E...
7:11:41.5636903 PM	Lab09-01.exe	796	CreateFileMapp...	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5637076 PM	Lab09-01.exe	796	CreateFileMapp...	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5637545 PM	Lab09-01.exe	796	CloseFile	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	
7:11:41.5638210 PM	Lab09-01.exe	796	Load Image	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	Image Base: 0x747...
7:11:41.5638825 PM	Lab09-01.exe	796	RegOpenKey	HKLM\Software\Microsoft\Windows N...	NAME NOT FOUND	Desired Access: R...
7:11:41.5640618 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\ntdll.dll	SUCCESS	CreationTime: 12/9...
7:11:41.5641468 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\imm32.dll	SUCCESS	CreationTime: 4/14...
7:11:41.5642300 PM	Lab09-01.exe	796	RegOpenKey	HKLM\Software\Microsoft\Rpc\Paged...	NAME NOT FOUND	Desired Access: R...
7:11:41.5642401 PM	Lab09-01.exe	796	RegOpenKey	HKLM\Software\Microsoft\Rpc	SUCCESS	Desired Access: R...
7:11:41.5642524 PM	Lab09-01.exe	796	RegQueryValue	HKLM\SOFTWARE\Microsoft\Rpc\Ma...	NAME NOT FOUND	Length: 144
7:11:41.5642663 PM	Lab09-01.exe	796	RegCloseKey	HKLM\SOFTWARE\Microsoft\Rpc	SUCCESS	
7:11:41.5642711 PM	Lab09-01.exe	796	RegOpenKey	HKLM\Software\Microsoft\Windows N...	NAME NOT FOUND	Desired Access: R...
7:11:41.5642934 PM	Lab09-01.exe	796	RegOpenKey	HKLM\Software\Policies\Microsoft\Win...	NAME NOT FOUND	Desired Access: R...
7:11:41.5643225 PM	Lab09-01.exe	796	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
7:11:41.5643376 PM	Lab09-01.exe	796	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
7:11:41.5643485 PM	Lab09-01.exe	796	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ, Le...
7:11:41.5643619 PM	Lab09-01.exe	796	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
7:11:41.5643708 PM	Lab09-01.exe	796	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
7:11:41.5647435 PM	Lab09-01.exe	796	RegOpenKey	HKLM\SOFTWARE\Microsoft\CTF\Co...	NAME NOT FOUND	Desired Access: R...
7:11:41.5647541 PM	Lab09-01.exe	796	RegOpenKey	HKLM\SOFTWARE\Microsoft\CTF\Sys...	SUCCESS	Desired Access: R...
7:11:41.5647653 PM	Lab09-01.exe	796	RegQueryValue	HKLM\SOFTWARE\Microsoft\CTF\Sys...	SUCCESS	Type: REG_DWO...
7:11:41.5647787 PM	Lab09-01.exe	796	RegCloseKey	HKLM\SOFTWARE\Microsoft\CTF\Sys...	SUCCESS	
7:11:41.5648508 PM	Lab09-01.exe	796	RegOpenKey	HKCU	SUCCESS	Desired Access: M...
7:11:41.5648650 PM	Lab09-01.exe	796	RegOpenKey	HKCU\Keyboard Layout\Toggle	SUCCESS	Desired Access: R...

Showing 10,028 of 134,751 events (7.4%) Backed by virtual memory

```
-----------------------------------
Keys added: 6
-----------------------------------
HKLM\SOFTWARE\Microsoft
HKLM\SOFTWARE\Microsoft \XPS
HKLM\SYSTEM\ControlSet001\Services\Lab09-01
HKLM\SYSTEM\ControlSet001\Services\Lab09-01\Security
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01
HKLM\SYSTEM\CurrentControlSet\Services\Lab09-01\Security
```

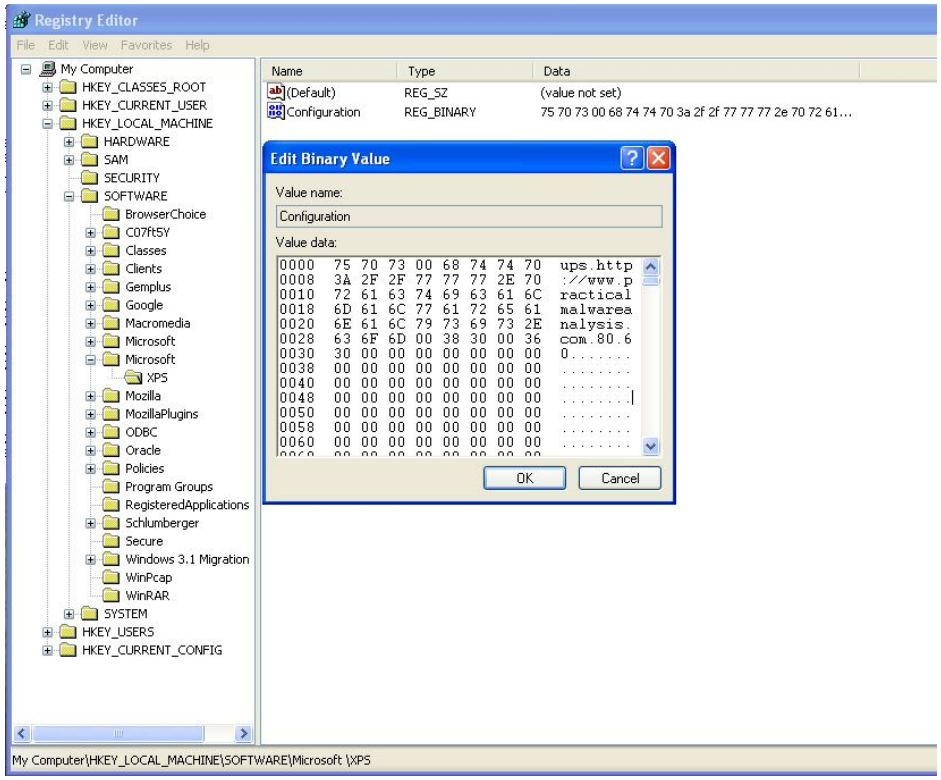


Registry Editor

File Edit View Favorites Help

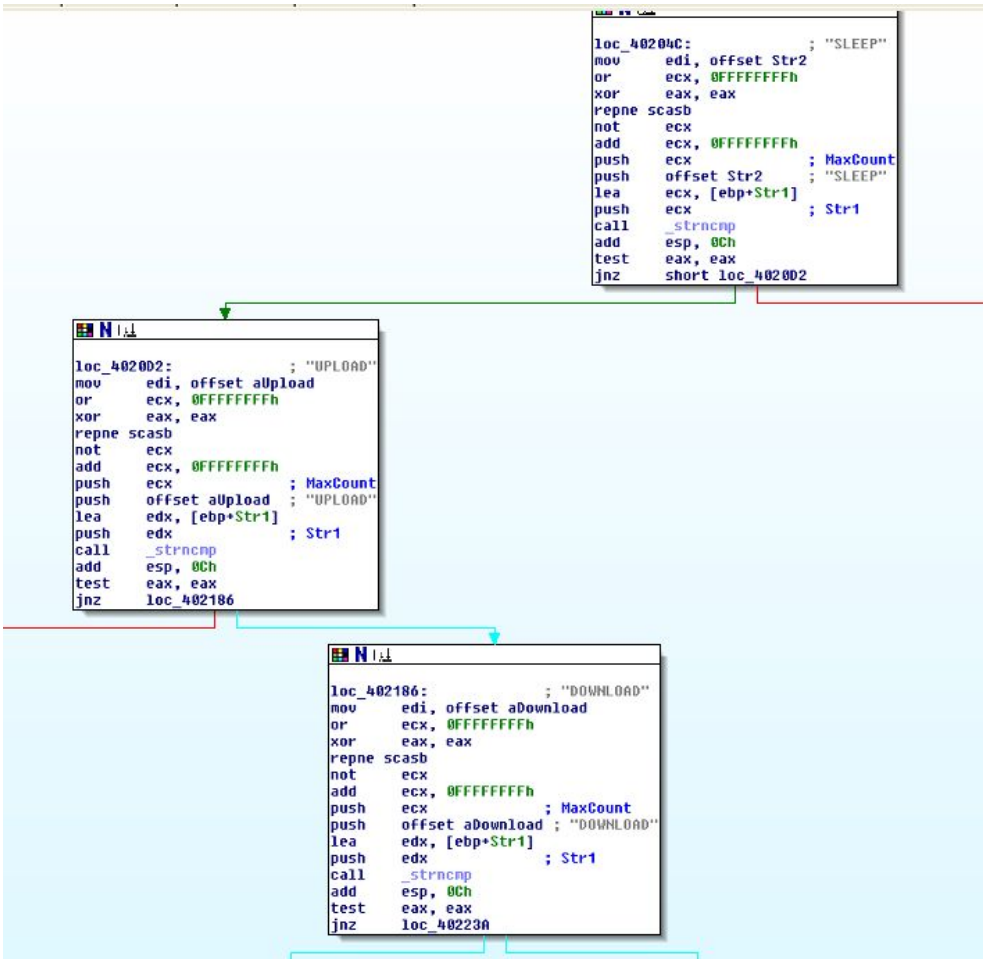
IpFilterDriver
IpInIp
IpNat
IPSec
IRENUM
ISAPISearch
isapnp
Kbdclass
kmixer
KSecDD
Lab09-01
Security
LanmanServer
lanmanworkstatio
lbrtfdc

Name	Type	Data
(Default)	REG_SZ	(value not set)
Security	REG_BINARY	01 00 14 80 90 00 00 00 9c 00 00 00 14 00 00 00 30 00 00 00 02 ...



5. What are the different actions this malware can be instructed to take via the network?

With the command line, the malware can install, add configuration to services and register. Or, if there is no command parameter, the program will also query the register with function sub_401000 and finally goes to sub_402360. The function can call sleep of the system and call sub_402020 which provide sleep, upload, download, open cmd operations.



6. Are there any useful network-based signatures for this malware?

We get from Wireshark that the malware is trying to build a connection to www.practicalmalwareanalysis.com.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	CadmusCo_3e:cf:05	Broadcast	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
2	0.00032500	RealtekU_12:35:02	CadmusCo_3e:cf:05	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
3	0.00032900	10.0.2.15	192.168.1.1	DNS	99	Standard query 0xc1aa A http://www.practicalmalwareanalysis.com
4	0.09001400	192.168.1.1	10.0.2.15	DNS	145	Standard query response 0xc1aa CNAME practicalmalwareanalysis.com A 192.0.78.24 A 192.0.78.25
5	0.09328600	10.0.2.15	192.0.78.24	TCP	62	nsstp > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
6	0.10278300	192.0.78.24	10.0.2.15	TCP	60	http > nsstp [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7	0.10281600	10.0.2.15	192.0.78.24	TCP	54	nsstp > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
8	0.10297800	10.0.2.15	192.0.78.24	HTTP	84	GET 7S4g/3Wh4.7yS HTTP/1.0
9	0.10323300	192.0.78.24	10.0.2.15	TCP	60	http > nsstp [ACK] Seq=1 Ack=31 Win=65535 Len=0
10	0.11116900	192.0.78.24	10.0.2.15	HTTP	1473	HTTP/1.1 400 Bad Request (text/html)
11	0.11119300	192.0.78.24	10.0.2.15	TCP	60	http > nsstp [FIN, ACK] Seq=1420 Ack=31 Win=65535 Len=0
12	0.11121400	10.0.2.15	192.0.78.24	TCP	54	nsstp > http [ACK] Seq=31 Ack=1421 Win=62821 Len=0
13	0.11176400	10.0.2.15	192.0.78.24	TCP	54	nsstp > http [FIN, ACK] Seq=31 Ack=1421 Win=62821 Len=0
14	0.11202300	192.0.78.24	10.0.2.15	TCP	60	http > nsstp [ACK] Seq=1421 Ack=32 Win=65535 Len=0

Lab09-02:

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

We got some alert messages and text notifications. The GetLastActivePopup, GetActiveWindow, and MessageBoxA might be the name of the functions.

IDA View-A | Strings window | Hex View-A | Structures | Enums | Imports

Address	Length	T...	String
.rdata:0040...	0000000F	C	runtime error
.rdata:0040...	0000000E	C	TLOSS error\r\n
.rdata:0040...	0000000D	C	SING error\r\n
.rdata:0040...	0000000F	C	DOMAIN error\r\n
.rdata:0040...	00000025	C	R6028\r\n- unable to initialize heap\r\n
.rdata:0040...	00000035	C	R6027\r\n- not enough space for lowio initialization\r\n
.rdata:0040...	00000035	C	R6026\r\n- not enough space for stdio initialization\r\n
.rdata:0040...	00000026	C	R6025\r\n- pure virtual function call\r\n
.rdata:0040...	00000035	C	R6024\r\n- not enough space for _onexit/atexit table\r\n
.rdata:0040...	00000029	C	R6019\r\n- unable to open console device\r\n
.rdata:0040...	00000021	C	R6018\r\n- unexpected heap error\r\n
.rdata:0040...	0000002D	C	R6017\r\n- unexpected multithread lock error\r\n
.rdata:0040...	0000002C	C	R6016\r\n- not enough space for thread data\r\n
.rdata:0040...	00000021	C	\r\nabnormal program termination\r\n
.rdata:0040...	0000002C	C	R6009\r\n- not enough space for environment\r\n
.rdata:0040...	0000002A	C	R6008\r\n- not enough space for arguments\r\n
.rdata:0040...	00000025	C	R6002\r\n- floating point not loaded\r\n
.rdata:0040...	00000025	C	Microsoft Visual C++ Runtime Library
.rdata:0040...	0000001A	C	Runtime Error!\n\nProgram:
.rdata:0040...	00000017	C	<program name unknown>
.rdata:0040...	00000013	C	GetLastActivePopup
.rdata:0040...	00000010	C	GetActiveWindow
.rdata:0040...	0000000C	C	MessageBoxA
.rdata:0040...	0000000B	C	user32.dll
.rdata:0040...	0000000D	C	KERNEL32.dll
.rdata:0040...	0000000B	C	WS2_32.dll
.data:0040...	00000006	C	`éyé!

2. What happens when you run this binary?

It runs and finished. Nothing is displayed.

```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\lynn>cd Desktop

C:\Documents and Settings\lynn\Desktop>cd Chapter_5L

C:\Documents and Settings\lynn\Desktop\Chapter_5L>cd ..

C:\Documents and Settings\lynn\Desktop>cd Chapter_9L

C:\Documents and Settings\lynn\Desktop\Chapter_9L>Lab09-02.exe

C:\Documents and Settings\lynn\Desktop\Chapter_9L>_
```

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

```
lea     edi, [ebp+var_1F0]
rep movsd
movsb
mov     [ebp+var_1B8], 0
mov     [ebp+Str], 0
mov     ecx, 43h
xor     eax, eax
lea     edi, [ebp+var_2FF]
rep stosd
stosb
push    10Eh            ; nSize
lea     eax, [ebp+Str]
push    eax             ; lpFilename
push    0               ; hModule
call    ds:GetModuleFileNameA
push    5Ch             ; Ch
lea     ecx, [ebp+Str]
push    ecx             ; Str
call    _strrchr
add     esp, 8
mov     [ebp+Str2], eax
mov     edx, [ebp+Str2]
add     edx, 1
mov     [ebp+Str2], edx
mov     eax, [ebp+Str2]
push    eax             ; Str2
lea     ecx, [ebp+Str1]
push    ecx             ; Str1
call    _strcmp
add     esp, 8
test    eax, eax
jz      short loc_40124C
```

The malware gets its module file name and compare it with some letters of its alphabet. [Str1+epb] indicate ‘o’

```
push    edi
mov     [ebp+var_1B0], '1'
mov     [ebp+var_1AF], 'q'
mov     [ebp+var_1AE], 'a'
mov     [ebp+var_1AD], 'z'
mov     [ebp+var_1AC], '2'
mov     [ebp+var_1AB], 'w'
mov     [ebp+var_1AA], 's'
mov     [ebp+var_1A9], 'x'
mov     [ebp+var_1A8], '3'
mov     [ebp+var_1A7], 'e'
mov     [ebp+var_1A6], 'd'
mov     [ebp+var_1A5], 'c'
mov     [ebp+var_1A4], 0
mov     [ebp+Str1], 'o'
mov     [ebp+var_19F], 'c'
mov     [ebp+var_19E], 'l'
mov     [ebp+var_19D], '.'
mov     [ebp+var_19C], 'e'
mov     [ebp+var_19B], 'x'
mov     [ebp+var_19A], 'e'
mov     [ebp+var_199], 0
```

In dynamic analysis, we got that the program is comparing “ocl.exe” with the name that it gets. So just simply rename the file as ocl.exe.

```
00401210  . 8D8D 00FDFFFF  LEA ECX,DWORD PTR SS:[EBP-300]
00401216  . 51             PUSH ECX
0040121C  . E8 34030000    CALL Lab09-02.00401550
0040121C    83C4 08        ADD ESP,8
0040121F  . 8945 FC        MOV DWORD PTR SS:[EBP-4],EAX
00401222  . 8B55 FC        MOV EDX,DWORD PTR SS:[EBP-4]
00401225  . 83C2 01        ADD EDX,1
00401228  . 8955 FC        MOV DWORD PTR SS:[EBP-4],EDX
0040122B  . 8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
0040122E  . 50             PUSH EAX
0040122F  . 8D8D 60FEFFFF  LEA ECX,DWORD PTR SS:[EBP-1A0]
00401235  . 51             PUSH ECX
00401236  . E8 85020000    CALL Lab09-02.004014C0
0040123B  . 83C4 08        ADD ESP,8
0040123E  . 85C0           TEST EAX,EAX
00401240  .v74 0A          JE SHORT Lab09-02.0040124C
00401242  . B8 01000000    MOV EAX,1
00401247  .vE9 8A010000    JMP Lab09-02.004013D6
0040124C  > BA 01000000    MOV EDX,1
00401251  . 85D2           TEST EDX,EDX
ESP=0012FC6C

Address   0012FC5C  00000000  ..
          0012FC60  0012FD8E  ‡
          0012FC64  0012FF80  ‡
          0012FC68  0040121C  @  Lab09-02.0040121C
          0012FC6C  0012FC80  ‡  ASCII "C:\Documents and Settings\lynn\Desktop\Chapter_9L\Lab09-02.exe"
          0012FC70  0000005C  \
          0012FC74  7C910228  ''  ntdll.7C910228
          0012FC78  FFFFFFFF  ..
```

```
0040122F  . 8D8D 60FEFFFF  LEA ECX,DWORD PTR SS:[EBP-1A0]
00401235  . 51             PUSH ECX
00401236  . E8 85020000    CALL Lab09-02.004014C0
0040123B  . 83C4 08        ADD ESP,8
0040123E  . 85C0           TEST EAX,EAX
00401240  .v74 0A          JE SHORT Lab09-02.0040124C
00401242  . B8 01000000    MOV EAX,1
00401247  .vE9 8A010000    JMP Lab09-02.004013D6
0040124C  > BA 01000000    MOV EDX,1
00401251  . 85D2           TEST EDX,EDX
004014C0=Lab09-02.004014C0

Address   0012FC6C  0012FDE0  ‡  ASCII "ocl.exe"
          0012FC70  0012FCB2  ‡  ASCII "Lab09-02.exe"
          0012FC74  7C910228  ''  ntdll.7C910228
          0012FC78  FFFFFFFF  ..
```

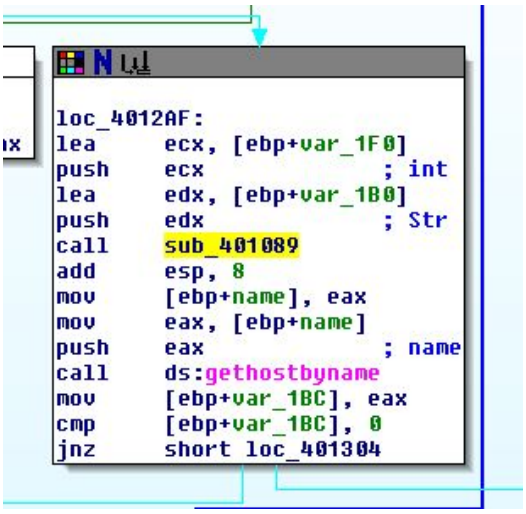
4. What is happening at 0x00401133?

That’s the alphabet the programmer create that for preventing IDA pro detect the string ‘ocl.exe’ in the strings tool, so he uses characters to combine them together.

```
push    edi
mov     [ebp+var_1B0], '1'
mov     [ebp+var_1AF], 'q'
mov     [ebp+var_1AE], 'a'
mov     [ebp+var_1AD], 'z'
mov     [ebp+var_1AC], '2'
mov     [ebp+var_1AB], 'w'
mov     [ebp+var_1AA], 's'
mov     [ebp+var_1A9], 'x'
mov     [ebp+var_1A8], '3'
mov     [ebp+var_1A7], 'e'
mov     [ebp+var_1A6], 'd'
mov     [ebp+var_1A5], 'c'
mov     [ebp+var_1A4], l
mov     [ebp+Str1], 'o'
mov     [ebp+var_19F], 'c'
mov     [ebp+var_19E], 'l'
mov     [ebp+var_19D], '.'
mov     [ebp+var_19C], 'e'
mov     [ebp+var_19B], 'x'
mov     [ebp+var_19A], 'e'
mov     [ebp+var_199], 0
```

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

0x00401089 is called here, using the Xref tool. It passed a string and a pointer as it showed.



It seems that the function sub_401089 contains a loop that can read a handle the string to other forms probably XOR operation.

```
; int __cdecl sub_401089(char *Str, int)
sub_401089 proc near

var_108= dword ptr -108h
var_104= dword ptr -104h
var_100= byte ptr -100h
var_FF= byte ptr -0FFh
Str= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 108h
push    edi
mov     [ebp+var_108], 0
mov     [ebp+var_100], 0
mov     ecx, 3Fh
xor     eax, eax
lea     edi, [ebp+var_FF]
rep stosd
stosw
stosb
mov     eax, [ebp+Str]
push    eax             ; Str
call    _strlen
add     esp, 4
mov     [ebp+var_104], eax
mov     [ebp+var_108], 0
jmp     short loc_4010E3
```

```
loc_4010E3:
cmp     [ebp+var_108], 20h
jge     short loc_40111D
```

```
mov     edx, [ebp+arg_4]
add     edx, [ebp+var_108]
movsx   ecx, byte ptr [edx]
mov     eax, [ebp+var_108]
cdq
idiv    [ebp+var_104]
mov     eax, [ebp+Str]
movsx   edx, byte ptr [eax+edx]
xor     ecx, edx
mov     eax, [ebp+var_108]
mov     [ebp+eax+var_100], cl
jmp     short loc_4010D4
```

```
loc_4010D4:
mov     ecx, [ebp+var_108]
add     ecx, 1
mov     [ebp+var_108], ecx
```

```
loc_40111D:
lea     eax, [ebp+var_100]
pop     edi
mov     esp, ebp
pop     ebp
retn
sub_401089 endp
```

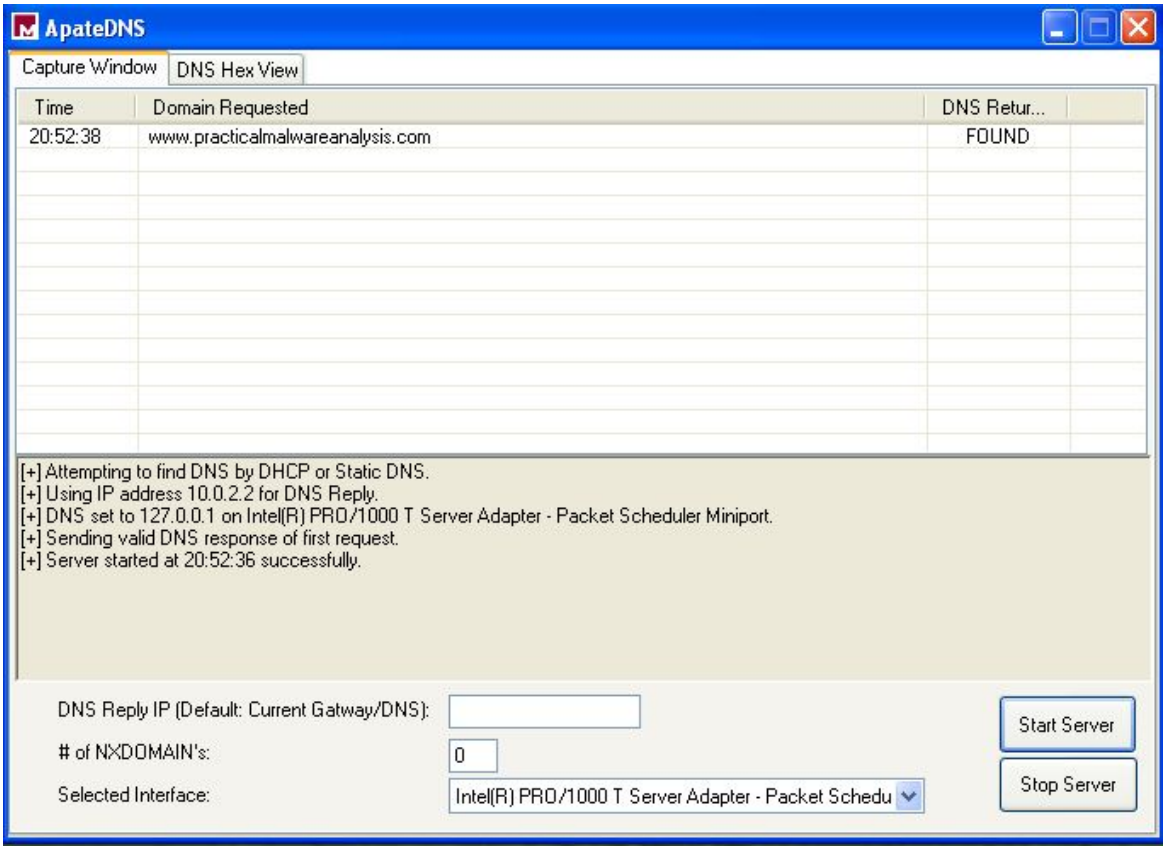
As it is shown in Ollydbg the string can probably be “1qaz2wsx3edc”, and it’s trying to decode to get the host domain!

```
004010E3  > 83BD F8FEFFFF  CMP DWORD PTR SS:[EBP-108],20
004010EA  .v7D 31          JGE SHORT ocl.0040111D
004010EC  . 8B55 0C        MOV EDX,DWORD PTR SS:[EBP+C]
004010EF  . 0395 F8FEFFFF  ADD EDX,DWORD PTR SS:[EBP-108]
004010F5  . 0FBE0A         MOVSX ECX,BYTE PTR DS:[EDX]
004010F8  . 8B85 F8FEFFFF  MOV EAX,DWORD PTR SS:[EBP-108]
004010FE  . 99             CDQ
004010FF  . F7BD FCFEFFFF  IDIV DWORD PTR SS:[EBP-104]
00401105  . 8B45 08        MOV EAX,DWORD PTR SS:[EBP+8]
00401108  . 0FBE1410       MOVSX EDX,BYTE PTR DS:[EAX+EDX]
0040110C  . 33CA           XOR ECX,EDX
0040110E  . 8B85 F8FEFFFF  MOV EAX,DWORD PTR SS:[EBP-108]
00401114  . 888C05 00FFFF  MOV BYTE PTR SS:[EBP+EAX-100],CL
0040111B  .^EB B7          JMP SHORT ocl.004010D4
0040111D  > 8D85 00FFFFFF  LEA EAX,DWORD PTR SS:[EBP-100]
00401123  . 5F             POP EDI
00401124  . 8BE5           MOV ESP,EBP
00401126  . 5D             POP EBP
00401127  . C3             RETN
00401128  $ 55             PUSH EBP
00401129  . 8BEC           MOV EBP,ESP
0040112B  . 81EC 04030000  SUB ESP,304
00401131  . 56             PUSH ESI
00401132  . 57             PUSH EDI
Stack SS:[0012FC6C]=0012FDD0, (ASCII "1qaz2wsx3edc")
EAX=00000000
```

```
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 MOV DWORD PTR SS:[EBP-108],0
004010D2  .v 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 CMP DWORD PTR SS:[EBP-108],20
004010EA  .v 7D 31        JGE SHORT ocl.0040111D
004010EC  . 8B55 0C       MOV EDX,DWORD PTR SS:[EBP+C]
004010EF  . 0395 F8FEFFFF ADD EDX,DWORD PTR SS:[EBP-108]
004010F5  . 0FBE0A        MOVSX ECX,BYTE PTR DS:[EDX]
004010F8  . 8B85 F8FEFFFF MOV EAX,DWORD PTR SS:[EBP-108]
004010FE  . 99            CDQ
004010FF  . F7BD FCFEFFFF IDIV DWORD PTR SS:[EBP-104]
00401105  . 8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
00401108  . 0FBE1410      MOVSX EDX,BYTE PTR DS:[EAX+EDX]
0040110C  . 33CA          XOR ECX,EDX
0040110E  . 8B85 F8FEFFFF MOV EAX,DWORD PTR SS:[EBP-108]
00401114  . 888C05 00FFFF MOV BYTE PTR SS:[EBP+EAX-100],CL
0040111B  .^EB B7         JMP SHORT ocl.004010D4
0040111D  > 8D85 00FFFFFF LEA EAX,DWORD PTR SS:[EBP-100]
00401123  . 5F            POP EDI
00401124  . 8BE5          MOV ESP,EBP
00401126  . 5D            POP EBP
00401127  . C3            RETN
00401128  $ 55            PUSH EBP
00401129  . 8BEC          MOV EBP,ESP
0040112B  . 81EC 04030000 SUB ESP,304
00401131  . 56            PUSH ESI
00401132  . 57            PUSH EDI

Stack SS:[0012FB5C]=0000000A
ECX=00000063
Jump from 0040111B

Address   ASCII dump
0012FA7C  ...www.practic...
0012FA9C  ...www.practic...
0012FABC  ...www.practic...
0012FADC  ...www.practic...
0012FAFC  ...www.practic...
0012FB1C  ...www.practic...
0012FB3C  ...www.practic...
0012FB5C  ...www.practic...
0012FB7C  ...www.practic...
0012FB9C  ...www.practic...
0012FBBC  ...www.practic...
0012FBDC  ...www.practic...
0012FBFC  ...www.practic...
0012FC1C  ...www.practic...
```

6. Whatdomain name does this malware use?

Find in ApateDNS, it's www.practicalmalwareanalysis.com.

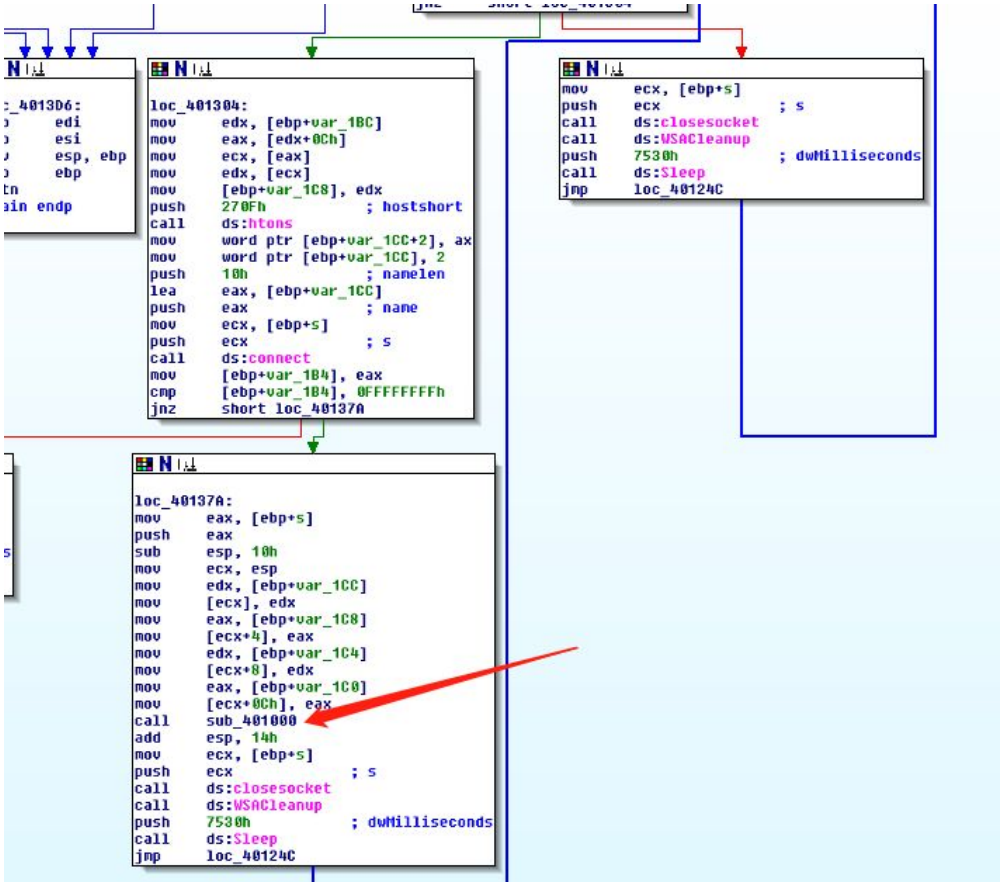


7. What encoding routine is being used to obfuscate the domain name?

As analyzed in question 5, it should be XOR.

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

CreateProcessA is called by sub_401000 in the main function. Generally, it builds a sock connection to a domain, and run CreateProcessA before the socket is closed.



The startupinfo, hstdinput looks like the inputs/outputs from a command window. Guessing that the socket is made to passing data to the domain and return the data the domain inputted which looks like an ssh connection from a remote client.

```
call    _memset
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    0FFFFFFFFh      ; dwMilliseconds
mov     ecx, [ebp+ProcessInformation.hProcess]
push    ecx             ; hHandle
call    ds:WaitForSingleObject
xor     eax, eax
```

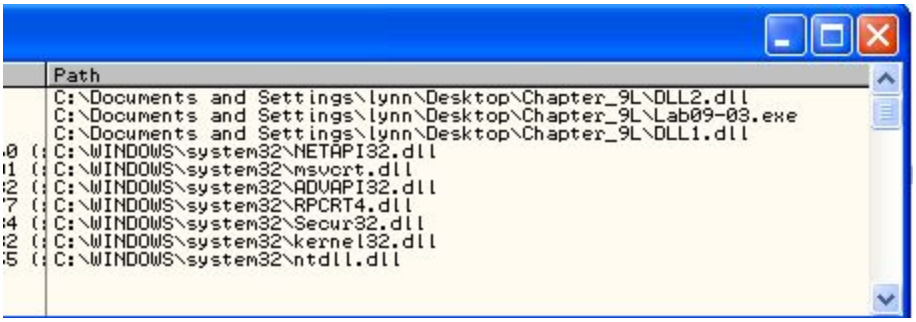
Lab09-03:

1. What DLLs are imported by Lab09-03.exe?

000000...	GetCurrentProcess	KERNEL32
000000...	GetCommandLineA	KERNEL32
000000...	GetCPInfo	KERNEL32
000000...	GetACP	KERNEL32
000000...	FreeEnvironmentStringsW	KERNEL32
000000...	FreeEnvironmentStringsA	KERNEL32
000000...	ExitProcess	KERNEL32
000000...	CloseHandle	KERNEL32
000000...	WriteFile	KERNEL32
000000...	WideCharToMultiByte	KERNEL32
000000...	VirtualFree	KERNEL32
000000...	VirtualAlloc	KERNEL32
000000...	UnhandledExceptionFilter	KERNEL32
000000...	TerminateProcess	KERNEL32
000000...	Sleep	KERNEL32
000000...	SetHandleCount	KERNEL32
000000...	RtlUnwind	KERNEL32
000000...	MultiByteToWideChar	KERNEL32
000000...	LoadLibraryA	KERNEL32
000000...	LCMapStringW	KERNEL32
000000...	LCMapStringA	KERNEL32
000000...	HeapReAlloc	KERNEL32
000000...	HeapFree	KERNEL32
000000...	HeapDestroy	KERNEL32
000000...	HeapCreate	KERNEL32
000000...	HeapAlloc	KERNEL32
000000...	GetVersionExA	KERNEL32
000000...	GetVersion	KERNEL32
000000...	DLL2ReturnJ	DLL2
000000...	DLL2Print	DLL2
000000...	DLL1Print	DLL1

00000000004050B8	NetScheduleJobAdd	NETAPI32
0000000000405060	SetHandleCount	KERNEL32
0000000000405064	GetStdHandle	KERNEL32
0000000000405068	GetFileType	KERNEL32
000000000040506C	GetStartupInfoA	KERNEL32
0000000000405070	GetModuleHandleA	KERNEL32
0000000000405074	GetEnvironmentVariableA	KERNEL32
0000000000405078	GetVersionExA	KERNEL32
000000000040507C	HeapDestroy	KERNEL32
0000000000405080	HeapCreate	KERNEL32
0000000000405084	VirtualFree	KERNEL32
0000000000405088	HeapFree	KERNEL32
000000000040508C	RtlUnwind	KERNEL32
0000000000405090	HeapAlloc	KERNEL32
0000000000405094	GetCPInfo	KERNEL32
0000000000405098	GetACP	KERNEL32
000000000040509C	GetOEMCP	KERNEL32
00000000004050A0	VirtualAlloc	KERNEL32
00000000004050A4	HeapReAlloc	KERNEL32
00000000004050A8	MultiByteToWideChar	KERNEL32
00000000004050AC	LCMapStringA	KERNEL32
00000000004050B0	GetStringTypeW	KERNEL32
0000000000405014	WriteFile	KERNEL32
0000000000405018	LCMapStringW	KERNEL32
000000000040501C	CloseHandle	KERNEL32
0000000000405020	LoadLibraryA	KERNEL32
0000000000405024	GetProcAddress	KERNEL32
0000000000405028	GetStringTypeA	KERNEL32
000000000040502C	Sleep	KERNEL32
0000000000405030	GetCommandLineA	KERNEL32
0000000000405034	GetVersion	KERNEL32

Got it from IDA pro, DLLs like kernel32, dll2, dll1. Netapi32 are imported.



In OllyDbg msvcrt.dll, RPCRT4.dll, Secur32.dll, ntdll.dll are also imported after the programming runs.

2. What is the base address requested by DLL1.dll, DLL2.dll, and DLL3.dll?

```
;
; Input MD5    : 1F9775ED5D105B4D86B67DEED9C5CF62

; File Name    : C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL1.dll
; Format       : Portable executable for 80386 (PE)
; Imagebase    : 10000000
; Section 1. (virtual address 00001000)
; Virtual size                 : 000054FA (   21754.)
; Section size in file         : 00006000 (   24576.)
; Offset to raw data for section: 00001000
; Flags 60000020: Text Executable Readable
```

```
; File Name    : C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL2.dll
; Format       : Portable executable for 80386 (PE)
; Imagebase    : 10000000
; Section 1. (virtual address 00001000)
; Virtual size                 : 0000551A (   21786.)
; Section size in file         : 00006000 (   24576.)
; Offset to raw data for section: 00001000
; Flags 60000020: Text Executable Readable
```

```
; File Name    : C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL3.dll
; Format       : Portable executable for 80386 (PE)
; Imagebase    : 10000000
; Section 1. (virtual address 00001000)
; Virtual size                 : 0000554A (   21834.)
; Section size in file         : 00006000 (   24576.)
; Offset to raw data for section: 00001000
; Flags 60000020: Text Executable Readable
; Alignment      : default
; OS type          :  MS Windows
; Application type:  DLL 32bit
```

They all show that in Imagebase :10000000.

3. When you use OllyDbg to debug Lab09-03.exe, what is the assigned based address for: DLL1.dll,DLL2.dll, and DLL3.dll?

Executable modules

Base	Size	Entry	Name	File version	Path
00330000	0000E000	00331174	DLL2		C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL2.dll
00400000	00009000	004010A2	Lab09-03		C:\Documents and Settings\lynn\Desktop\Chapter_9L\Lab09-03.exe
10000000	0000E000	10001152	DLL1		C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL1.dll
5B860000	00056000	5B868B48	NETAPI32	5.1.2600.6260 (	C:\WINDOWS\system32\NETAPI32.dll
77C10000	00058000	77C1F2A1	msvcrt	7.0.2600.5701 (	C:\WINDOWS\system32\msvcrt.dll
77DD0000	0009B000	77DD710B	ADVAPI32	5.1.2600.6382 (	C:\WINDOWS\system32\ADVAPI32.dll
77E70000	00093000	77E7628F	RPCRT4	5.1.2600.6477 (	C:\WINDOWS\system32\RPCRT4.dll
77FE0000	00011000	77FE2146	Secur32	5.1.2600.5834 (	C:\WINDOWS\system32\Secur32.dll
7C800000	000F6000	7C80B64E	kernel32	5.1.2600.6532 (	C:\WINDOWS\system32\kernel32.dll
7C900000	000B2000	7C912AFC	ntdll	5.1.2600.6055 (	C:\WINDOWS\system32\ntdll.dll

After open .exe with OllyDbg, the executable window shows the address, entry, and base address of dll libraries.

4. When Lab09-03.exe calls an import function from DLL1.dll, what does this import function do?

```
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 1Ch
call    ds:DLL1Print
call    ds:DLL2Print
call    ds:DLL2ReturnJ
mov     [ebp+hObject], eax
push    0               ; lpOverlapped
lea     eax, [ebp+NumberOfBytesWritten]
push    eax             ; lpNumberOfBytesWritten
push    17h             ; nNumberOfBytesToWrite
push    offset aMalwareanalysi ; "malwareanalysisbook.com"
```

In the main function, it called DLL1Print which is shown in the picture below.

```
; Exported entry   1. DLL1Print


; Attributes: bp-based frame

public DLL1Print
DLL1Print proc near
push    ebp
mov     ebp, esp
mov     eax, dword_10008030
push    eax
push    offset aDll1MysteryDat ; "DLL 1 mystery data %d\n"
call    sub_10001038
add     esp, 8
pop     ebp
retn
DLL1Print endp
```

It pushes the offset which is “DLL 1 mystery data %d\n” and eax, which is dword_10008030 to sub_10001038. Check the reference of dword, we got that it seems to be the current process id.

```
; Attributes: bp-based frame

; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
_DllMain@12 proc near

hinstDLL= dword ptr  8
fdwReason= dword ptr  0Ch
lpvReserved= dword ptr  10h

push    ebp
mov     ebp, esp
call    ds:GetCurrentProcessId
mov     dword_10008030, eax
mov     al, 1
pop     ebp
retn    0Ch
_DllMain@12 endp
```

5. When Lab09-03.exe calls WriteFile, what is the filename it writes to?

```
push    ebp
mov     ebp, esp
sub     esp, 1Ch
call    ds:DLL1Print
call    ds:DLL2Print
call    ds:DLL2ReturnJ
mov     [ebp+hObject], eax
push    0               ; lpOverlapped
lea     eax, [ebp+NumberOfBytesWritten]
push    eax             ; lpNumberOfBytesWritten
push    17h             ; nNumberOfBytesToWrite
push    offset aMalwareanalysi ; "malwareanalysisbook.com"
mov     ecx, [ebp+hObject]
push    ecx             ; hFile
call    ds:WriteFile
mov     edx, [ebp+hObject]
push    edx             ; hObject
call    ds:CloseHandle
push    offset LibFileName ; "DLL3.dll"
call    ds:LoadLibraryA
mov     [ebp+hModule], eax
push    offset ProcName ; "DLL3Print"
mov     eax, [ebp+hModule]
push    eax             ; hModule
call    ds:GetProcAddress
```

```
BOOL WINAPI WriteFile(
  _In_        HANDLE       hFile,
  _In_        LPCVOID      lpBuffer,
  _In_        DWORD        nNumberOfBytesToWrite,
  _Out_opt_   LPDWORD      lpNumberOfBytesWritten,
  _Inout_opt_ LPOVERLAPPED lpOverlapped
);
```

The WriteFile function requires a Handle with contains the name of the file to be written. The handle, in this case, is [ebp + h0bject] which is from DLL2ReturnJ.

```
; Exported entry   2. DLL2ReturnJ


; Attributes: bp-based frame

public DLL2ReturnJ
DLL2ReturnJ proc near
push    ebp
mov     ebp, esp
mov     eax, dword_1000B078
pop     ebp
retn
DLL2ReturnJ endp
```

```
push    ebp
mov     ebp, esp
push    0                ; hTemplateFile
push    80h              ; dwFlagsAndAttributes
push    2                ; dwCreationDisposition
push    0                ; lpSecurityAttributes
push    0                ; dwShareMode
push    40000000h        ; dwDesiredAccess
push    offset FileName  ; "temp.txt"
call    ds:CreateFileA
mov     dword_1000B078, eax
mov     al, 1
pop     ebp
retn    0Ch
_DllMain@12 endp
```

Finally we get dowrd_1000B078 in xref. The name of the file is “temp.txt”.

6. When Lab09-03.exe creates a job using NetScheduleJobAdd, where does it get the data for the second parameter?

```
push    offset LibFileName ; "DLL3.dll"
call    ds:LoadLibraryA
mov     [ebp+hModule], eax
push    offset ProcName ; "DLL3Print"
mov     eax, [ebp+hModule]
push    eax             ; hModule
call    ds:GetProcAddress
mov     [ebp+var_8], eax
call    [ebp+var_8]
push    offset aDll3getstructu ; "DLL3GetStructure"
mov     ecx, [ebp+hModule]
push    ecx             ; hModule
call    ds:GetProcAddress
mov     [ebp+var_10], eax
lea     edx, [ebp+Buffer]
push    edx
call    [ebp+var_10]
add     esp, 4
lea     eax, [ebp+JobId]
push    eax             ; JobId
mov     ecx, [ebp+Buffer]
push    ecx             ; Buffer
push    0               ; Servername
call    NetScheduleJobAdd
push    2710h           ; dwMilliseconds
call    ds:Sleep
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
_main endp
```

NetScheduleJodAdd called in main, it is a function which submits a job to run at a specified future time and date according to MSDN. This function requires that the schedule service is started on the computer to which the job is submitted.

```
NET_API_STATUS NetScheduleJobAdd(
  _In_opt_ LPCWSTR Servername,
  _In_     LPBYTE  Buffer,
  _Out_    LPDWORD JobId
);
```

The second parameter is Buffer which is [ebp + Buffer] in this case. Track ebp from LoadLibraryA. It loads DLL3.dll first and then calls GetProcAddress to get the base address of the function in DLL3.dll.

```
; Exported entry   1. DLL3GetStructure


; Attributes: bp-based frame

public DLL3GetStructure
DLL3GetStructure proc near

arg_0= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+arg_0]
mov     dword ptr [eax], offset dword_1000B0A0
pop     ebp
retn
DLL3GetStructure endp
```

```
lpMultiByteStr= dword ptr -4
hinstDLL= dword ptr  8
fdwReason= dword ptr  0Ch
lpvReserved= dword ptr  10h

push    ebp
mov     ebp, esp
push    ecx
mov     [ebp+lpMultiByteStr], offset aPingWww_malwar ; "ping www.malwareanalysisbook.com"
push    32h             ; cchWideChar
push    offset WideCharStr ; lpWideCharStr
push    0FFFFFFFFh      ; cbMultiByte
mov     eax, [ebp+lpMultiByteStr]
push    eax             ; lpMultiByteStr
push    0               ; dwFlags
push    0               ; CodePage
call    ds:MultiByteToWideChar
mov     dword_1000B0AC, offset WideCharStr
mov     dword_1000B0A0, 36EE80h
mov     dword_1000B0A4, 0
mov     byte_1000B0A8, 7Fh
mov     byte_1000B0A9, 11h
mov     al, 1
mov     esp, ebp
pop     ebp
retn    0Ch
_DllMain@12 endp
```

After checking the xref of dword_1000B0A0, we got the data source.

7. While running or debugging the program, you will see that it prints out three pieces of mystery data. What are the following: DLL 1 mystery data 1, DLL 2 mystery data 2, and DLL 3 mystery data 3?

DLL1 we get 1828. It should be the current process id according to previous analysis.

```
Command Prompt - Lab09-03.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\lynn>cd Desktop

C:\Documents and Settings\lynn\Desktop>cd Chapter_9L

C:\Documents and Settings\lynn\Desktop\Chapter_9L>Lab09-03.exe
DLL 1 mystery data 1828
DLL 2 mystery data -1
DLL 3 mystery data 3780800
```

```
; Exported entry   1. DLL2Print


; Attributes: bp-based frame

public DLL2Print
DLL2Print proc near
push    ebp
mov     ebp, esp
mov     eax, dword_1000B078
push    eax
push    offset aDll2MysteryDat ; "DLL 2 mystery data %d\n"
call    sub_1000105A
add     esp, 8
pop     ebp
retn
DLL2Print endp
```

```
push    0                   ; lpSecurityAttributes
push    0                   ; dwShareMode
push    40000000h           ; dwDesiredAccess
push    offset FileName     ; "temp.txt"
call    ds:CreateFileA
mov     dword_1000B078, eax
mov     al, 1
pop     ebp
retn    0Ch
_DllMain@12 endp
```

```
HANDLE WINAPI CreateFile(
  _In_      LPCTSTR                lpFileName,
  _In_      DWORD                  dwDesiredAccess,
  _In_      DWORD                  dwShareMode,
  _In_opt_  LPSECURITY_ATTRIBUTES  lpSecurityAttributes,
  _In_      DWORD                  dwCreationDisposition,
  _In_      DWORD                  dwFlagsAndAttributes,
  _In_opt_  HANDLE                 hTemplateFile
);
```

DLL2 print the data from dword_1000B078 which is the return value of CreateFileA, it returns a handle according to MSDN.

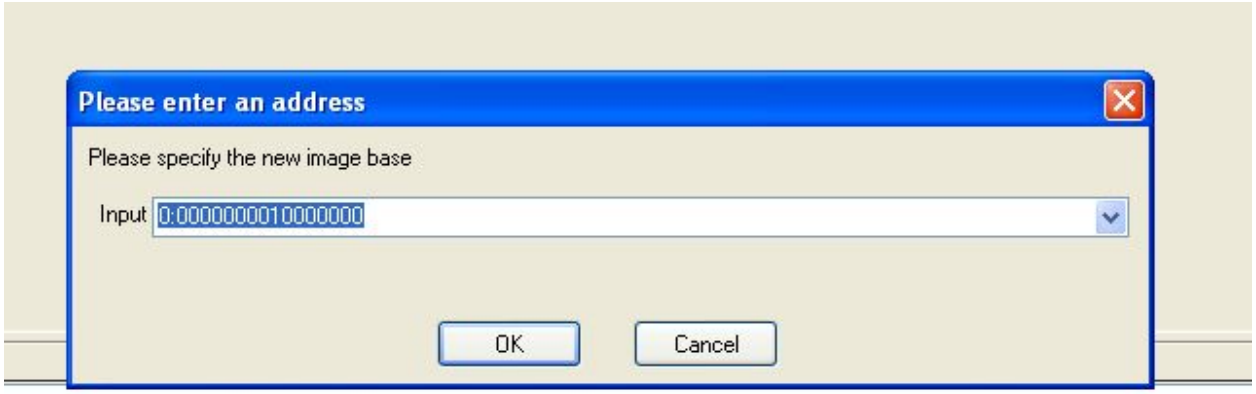
DLL2 print 3780800, which should be the byte of “ping www.malwareanalysisbook.com”.

```
; Exported entry   2. DLL3Print


; Attributes: bp-based frame

public DLL3Print
DLL3Print proc near
push    ebp
mov     ebp, esp
push    offset WideCharStr
push    offset aDll3MysteryDat ; "DLL 3 mystery data %d\n"
call    sub_10001087
add     esp, 8
pop     ebp
retn
DLL3Print endp
```

```
push    ebp
mov     ebp, esp
push    ecx
mov     [ebp+lpMultiByteStr], offset aPingWww_malwar ; "ping www.malwareanalysisbook.com"
push    32h             ; cchWideChar
push    offset WideCharStr ; lpWideCharStr
push    0FFFFFFFFh      ; cbMultiByte
mov     eax, [ebp+lpMultiByteStr]
push    eax             ; lpMultiByteStr
push    0               ; dwFlags
push    0               ; CodePage
call    ds:MultiByteToWideChar
mov     dword_1000B0AC, offset WideCharStr
mov     dword_1000B0A0, 36EE80h
mov     dword_1000B0A4, 0
mov     byte_1000B0A8, 7Fh
mov     byte_1000B0A9, 11h
mov     al, 1
mov     esp, ebp
pop     ebp
retn    0Ch
_DllMain@12 endp
```

8. How can you load DLL2.dll into IDA Pro so that it matches the load address used by OllyDbg?

As we explained the image base of these DLLs are 10000000. So when opening it with IDA pro, choose manual load to input the image base.


Please enter an address
Please specify the new image base
Input
0:0000000010000000
OK
Cancel