LAB 6

OllyDbg Malware Analysis

Under the direction of
Dr. Samuel Liles

Table of Contents

Abstract

This lab is focused on Malware Analysis. The lab is going to use tools and application to do Static/Dynamic analysis of the malware while being isolated from the internet. The Practical Lab 7.1 to Lab 7.3 will be carried out to answer the questions provided.

The Computer Anti-virus was disabled as part of the instructions to enable the download and extract of the files being used. This lab is intended to lay grounds for further labs in the course.

Keywords: Digital Investigation, Forensic Evidence, Malware Analysis.

Lab 5 Malware Analysis

Steps of the process

Preparing the LAB

The Computer was rebooted, anti-virus was disabled, and the appropriate files were downloaded. Different Images of VM were installed. Installation of different windows environment such as XP, 7 and 8.1. Programs needed have been downloaded and snapshots of the process have been taken.

LAB 9-1, 9-3

Applications & Tools

The following applications are used to forensically examine the files. The following descriptions have been captured from the developer's website and manuals.

PEiD," is an intuitive application that relies on its user-friendly interface to detect packers, cryptors and compilers found in PE executable files – its detection rate is higher than that of other similar tools since the app packs more than 600 different signatures in PE files" (Gröbert, 2010).

Resource Hacker,"is a freeware utility to view, modify, rename, add, delete and extract resources in 32bit & 64bit Windows executables and resource files (*.res). It incorporates an internal resource script compiler and decompiler and works on all (Win95 - Win7) Windows operating systems" (Johnson, 2011).

PE Explorer"provides powerful tools for disassembly and inspection of unknown binaries, editing the properties of 32-bit executable files and customizing and translating their resources. Use this product to do reverse engineering, analyze the procedures and libraries an executable uses." (Heaventools Software, 2009).

Process Monitor is an advanced monitoring tool for Windows that shows real-time file system, Registry and process/thread activity. It combines the features of two legacy Sysinternals utilities, Filemon and Regmon, and adds an extensive list of enhancements including rich and non-destructive filtering, comprehensive event properties such session IDs and user names, reliable process information, full thread stacks with integrated symbol support for each operation, simultaneous logging to a file, and much more. Its uniquely powerful features will make Process Monitor a core utility in your system troubleshooting and malware hunting toolkit (Russinovich & Cogswell, 2014).

ApateDNS, is a tool for controlling DNS responses though an easy to use GUI. As a phony DNS server, ApateDNS spoofs DNS responses to a user-specified IP address by listening on UDP port 53 on the local machine. It responds to DNS requests with the response set to any IP address you specify. The tool logs and timestamps any DNS request it receives. You may specify a number of non-existent domain (NXDOMAIN) responses to send before returning a valid response. ApateDNS also automatically sets the local DNS to localhost. By default, it will use either the set DNS or default gateway settings as an IP address to use for DNS responses. Upon exiting the tool, it sets back the original local DNS settings (Davis, 2011).

Regshot, is a small, free and open-source registry compare utility that allows you to quickly take a snapshot of your registry and then compare it with a second one - done after doing system changes or installing a new software product. The changes report can be produced in text or HTML format and contains a list of all modifications that have taken place between the two snapshots. In addition, you can also specify folders (with subfolders) to be scanned for changes as well (Regshot Team, 2013).

IDA is the Interactive DisAssembler: the world's smartest and most feature-full disassembler, which many software security specialists are familiar with (Hex-Rays SA, 2014).

IDA is the Interactive DisAssembler: the world's smartest and most feature-full disassembler, which many software security specialists are familiar with (Hex-Rays SA, 2014).

Issues or problems

For quastion 9-2 part 5, 7, 8 I had problem reading the registery, I tried reinstalling the program and using a fresh copy of the malware yet it was still unreadable, I used the hardware break and toggle the other types of breaks, I also tried reading the value of the registery storing the value of the string and still could not read it. I changed the Encoding from 64 to 32 to unicaode to ASCII and nothing has changed.

Conclusions

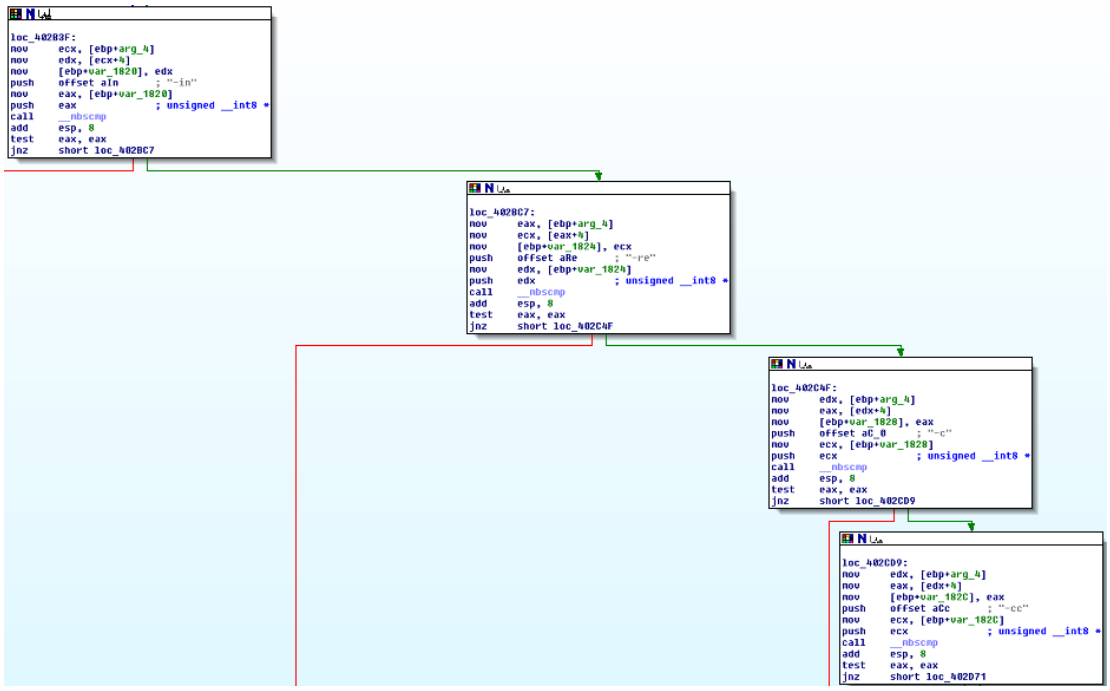
The Lab identified several programs that helps explore the malwares. The tools showed if the files being used are infected or packed. The tools used also showed the resources on the system that is being utilized such as privilege, CPU usage, Network communication.

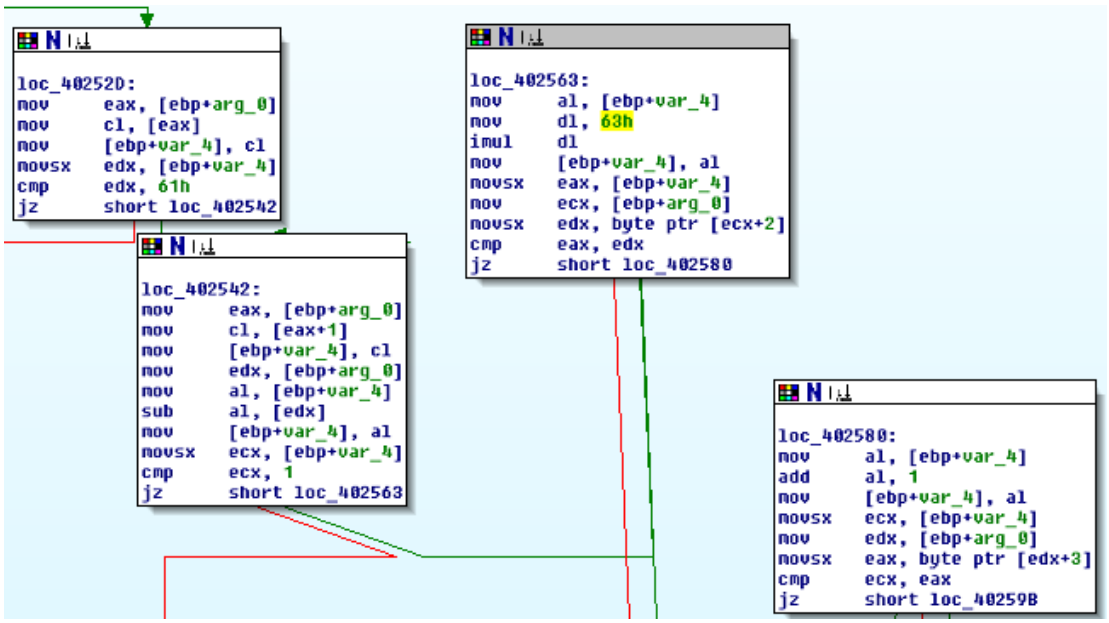
Case studies

No Case studies was given with this lab.

Review questions

Lab 9-1

Answers	Lab09-01. exe
1	From viwing the graphic design of the program in IDA Pro we can see that the program terminates if two arguments are not given. if the first one is given it checks for another if not it fails. The arguments are the commands that needs to be excuted from the graph we see them as -in, -re, -c,-cc. The second part is like a constant that has 4 charechaters  From checking the Function at 0x402512 we can find that its comparing each value to four other ones however we can clearly identify two of them which is 61h, 63h by using the refrence to look into the code we find that 61h is an 'a' and 63h is a 'c' and since we are missing a value between 'a' and 'c' it was resnable to assume the 2nd value is a 'b' and the last one is a 'd' with trying that on the command line after '-in' we can see that the malware response and install itself, uninstall, sets something, and finally prints. Going around that can be done by modifying the code however its important to note that

	the -in,-re,-c,-cc are important to specify the path needed to run the program therefore it should not be touched however the constant which is the password could be nullified to accept any input regardless of what it is. 
2	Command line options are as follows • Lab09-01.exe -in abcd <===== used to (in)stall • Lab09-01.exe -re abcd <===== used to (re)move • Lab09-01.exe -c abcd <===== used to (c)onfigure • Lab09-01.exe -cc abcd <===== used to (c)opy (c)onfigure
3	I can simply add a print out with the above commands when the program is called with no arguments to inform the user of his option this is better since no code modificaion is done. Or simply have the password hardcoded to be always correct, or nullify the password function so that it accepts anything or returns succesfully regardless of the input.
4	Checking the strings tabs in IDA Pro we can see that that system direcotry is being used as well as a function called copy. Which is also reflected in the graphsbellow.

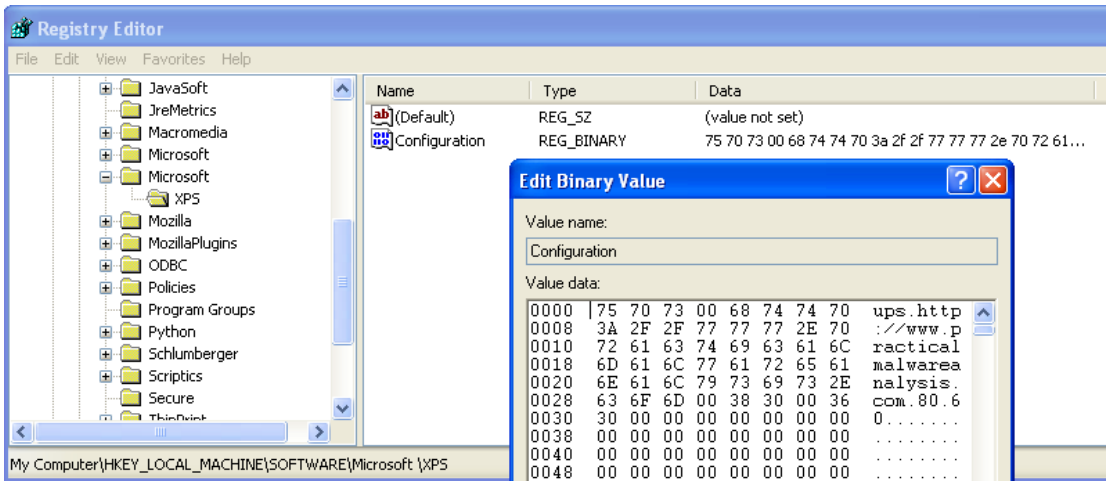
```
push    20h             ; dwServiceType
push    0F01FFh         ; dwDesiredAccess
lea     ecx, [ebp+DisplayName]
push    ecx             ; lpDisplayName
mov     edx, [ebp+lpServiceName]
push    edx             ; lpServiceName
mov     eax, [ebp+hSCManager]
push    eax             ; hSCManager
call    ds:CreateServiceA
mov     [ebp+hSCObject], eax
cmp     [ebp+hSCObject], 0
jnz     short loc_402831
```

```
loc_402891:             ; bFailIfExists
push    0
lea     ecx, [ebp+NewFileName]
push    ecx             ; lpNewFileName
lea     edx, [ebp+ExistingFileName]
push    edx             ; lpExistingFileName
call    ds:CopyFileA
test    eax, eax
jnz     short loc_4028B2
```

we can also see that a service is being created which means a record will be available in the registry using the strings we can see that the location will mostly likely be SOFTWARE\\Microsoft \\XPS.

```
".".data:00...  00000018  C  SOFTWARE\\Microsoft \\XPS
```

Using Regedit we can find the key with a website link in it under the location specified above.

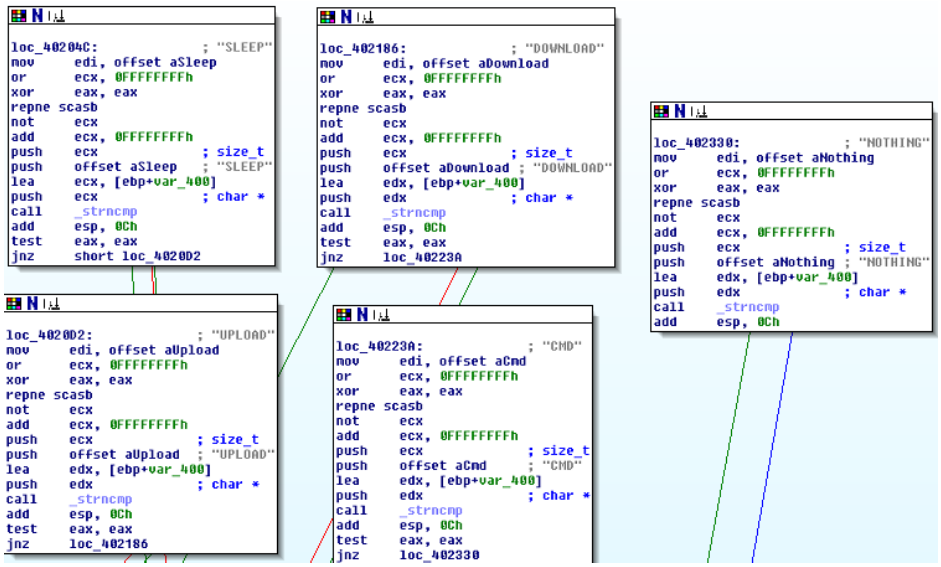


5

Going over the Strings once more we find the following interesting strings

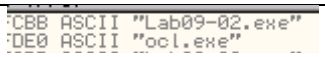
```
".".data:00...  00000008  C  NOTHING
".".data:00...  00000009  C  DOWNLOAD
".".data:00...  00000007  C  UPLOAD
".".data:00...  00000006  C  SLEEP
".".data:00...  00000008  C  cmd.exe
```

Using them as a refrence we go over the code to find that it reflects the options that can be taken by the malware. Nothing is do nothing, Download is using to download files from the host computer, upload is to upload a file to the host PC from a location given. cmd.exe is the most interesting one since it provided a shell command to the malware

	owner to run and use any system commands. 
6	Using the website link found in the rigestry key created and available in the strings in IDA Pro. we can locate the networking function and see in OllyDbg that the Practicemalwareanalysis.com website is being used to communicate with the malware owner. 

Lab 9-2

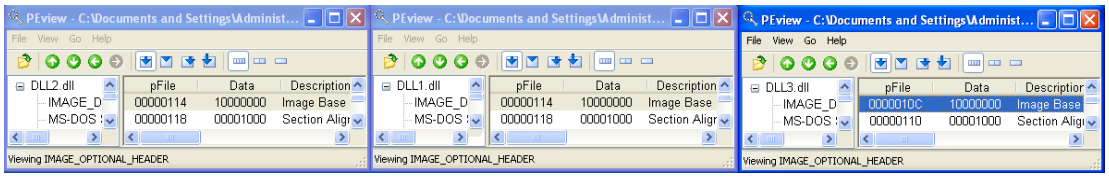
Answers	Lab09-02. exe
1	Single steping the program we find the following string being created when selecting different paths, ocl.exe, hi. After running the program and looking into the memory we find imports. Also when Finding all strings refrenced we find CMD.
2	The process finishes in no time. with no output as shown when using the following command Lab09-02.exe > temp.txt
3	We can see that the file is exiting after comparing the string to ocl.exe therefore we need to rename it to ocl.exe in order for the comparision to pass.

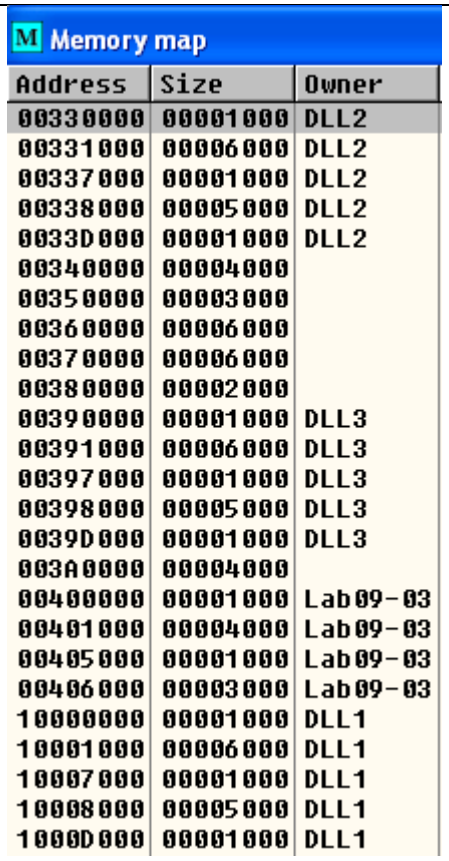
	
4	Lots of charicaters are being relocated and pushed to the stack. It looks like the attacker is trying to build a string out of letters. a nice way of hiding information or making it hard to read when scanning for strings available in the program.
5	the string was not readable.
6	This is the only domain name available in the malware practicalmalwareanalysis.com
7	The string was not readable.

8	All info needed to create the process has been pushed to the stack and one of the values is cmd therefore the process will create a socket that is linked to a command shell. This is interesting since cmd could be hidden and values can be streamed to any location.
	`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`

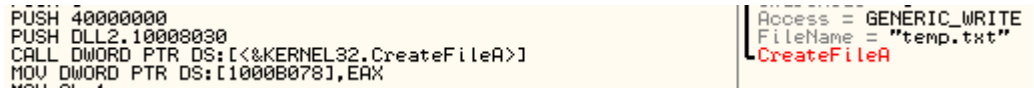
Lab 9-3

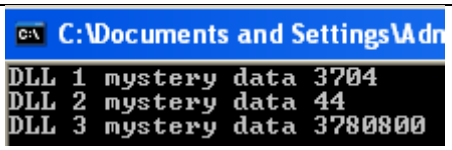
Answers	Lab09-03.exe; DLL1.dll, DLL2.dll, DLL3.dll
1	Using PEiD we can see the following DLLs being used; KERNEL32.dll, NETAPI32.dll, DLL1.dll,DLL2.dll. However, Running the string; using the following command strings -n 5 Lab09-03.exe > temp.txt; or viewing it in IDA Pro we can find that two more DLL names are mentioned which means its probebly being used at some point. user32.dll, DLL3.dll.

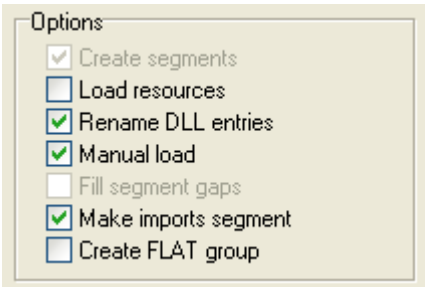
	Imports Viewer DllName / OriginalFirstThunk / TimeDateStamp / ForwarderChain / Name / FirstThunk KERNEL32.dll 000054CC 00000000 00000000 000055BC 00005014 NETAPI32.dll 00005570 00000000 00000000 000055DE 000050B8 DLL1.dll 000054B8 00000000 00000000 000055F8 00005000 DLL2.dll 000054C0 00000000 00000000 0000561C 00005008 **Occurences of: .dll** Address / Instruction .text:0040103C push offset LibFileName ; "DLL3.dll" .idata:00405000 ; Imports from DLL1.dll .idata:00405008 ; Imports from DLL2.dll .idata:00405014 ; Imports from KERNEL32.dll .idata:004050B8 ; Imports from NETAPI32.dll .rdata:0040541C aUser32_dll db 'user32.dll',0 .data:00406054 LibFileName db 'DLL3.dll',0
2	We can see that all of them start at the same point 0x10000000 
3	We can see from the picture that each DLL had a different starting address which makes sence since at run time you cannot have different instructions at the same address space. DLL2 is at 0x00330000,DLL3 is at 0x00390000, DLL1 is at 0x1000000.

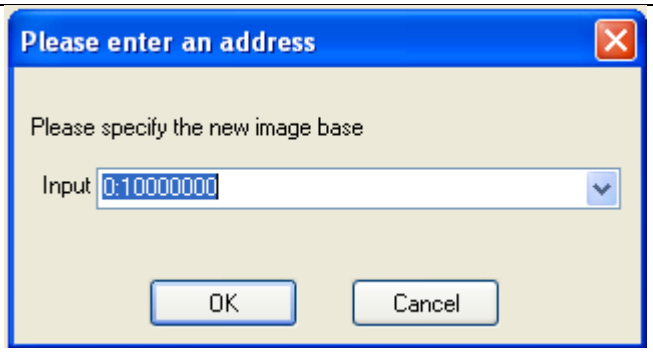
<table>
<tr><td></td><td>

Memory map

Address	Size	Owner
00330000	00001000	DLL2
00331000	00006000	DLL2
00337000	00001000	DLL2
00338000	00005000	DLL2
0033D000	00001000	DLL2
00340000	00004000	
00350000	00003000	
00360000	00006000	
00370000	00006000	
00380000	00002000	
00390000	00001000	DLL3
00391000	00006000	DLL3
00397000	00001000	DLL3
00398000	00005000	DLL3
0039D000	00001000	DLL3
003A0000	00004000	
00400000	00001000	Lab09-03
00401000	00004000	Lab09-03
00405000	00001000	Lab09-03
00406000	00003000	Lab09-03
10000000	00001000	DLL1
10001000	00006000	DLL1
10007000	00001000	DLL1
10008000	00005000	DLL1
1000D000	00001000	DLL1
</td></tr>
<tr><td>4</td><td>Using IDA Pro we can find that CALL command is used when calling DLL1Print. There is no other function that is being called from DLL1 other than this. So after knowing the function we start loading the DLL1 function into IDA Pro to analyze the function. After loading the file we find that GetCurrentProcessId is the only function that is being called and the variable storing the information returned is then used in the printing string.
In the following three lines we see the value returned by the function moved to eax which is then pushed to the stack, and then when the print command is given the %d is replaced by eax value.
.text:10001023 mov eax, dword_10008030
.text:10001028 push eax
.text:10001029 push offset aDll1MysteryDat ; "DLL 1 mystery data %d\n"</td></tr>
</table>

	Which probebly indecates that the function will print out the Process ID used by the malware. To verify that we can see that the value printed in the Command shell is the same one showen when running as shown in the graph bellow. 
5	Looking into all functions called writefile in IDA Pro we can see it used twice, in the Main section of the malware & in the subroutine sub_401E11. However, nothing in the program showes when the file is created before we can write to it therefore we start examining the other three DLLs. Since we have already checked DLL1 and found only the process ID function we start with DLL2. As soon as its loaded into OllyDbg we can see in the bignning of the code a CreateFileA function with temp.txt used for valuable which can be verified by looking into the folder hosting the malware to find an actual file called temp.txt already created. 
6	Looking into IDA Pro we can see that the function is being called after GetProcAddress which is a function in DLL3.dll. Furthermore, the function header is as followes; NetScheduleJobAdd(LPCWSTR Servername,LPBYTE Buffer,LPDWORD JobId).Which means that the 2nd parameter is LPBYTE Buffer. Going over the code we see that the value of Buffer is being stored in the Registery ecx. Following exc we find that it got its value from GetProcAddress Function which minipulated the value coming from LoadLibraryA which was stored in aex. Now we go into the DLL3.dll to see its original input source. Analyzing the code showes that the value return is a refrence pointer.

7	
	Previously we identified that DLL 1 mystery data is the proccess ID for the Malware. By going over DLL2print we can see that the value stored and printed is a pointer to the resouce file created temp.txt. that value enables the malware to interact with the file. And for the thrid we established that a pointer is being created and refrenced and one of the functions that refrences it is DLL3print therefore it is also the refrence point in memory. However this time the refrence point is the location where the following command is saved.
	ping www.malwareanalysisbook.com
	As shown bellow the WideCharStr value is the one being printed which is the value of the pointer to the above command.
	`push offset WideCharStr`
	`push offset aDll3MysteryDat ; "DLL 3 mystery data %d\n"`
	`.`
	`mov [ebp+lpMultiByteStr], offset aPingWww_malwar ; "ping www.malwareanalysisbook.com"`
	`push 32h ; cchWideChar`
	`push offset WideCharStr ; lpWideCharStr`

8	By checking the Manual Load in the Options when loading the file and then clicking OK.  Then it will show us another window in which we put in the value we want for loading the file and in this case that would be 0x00330000 for DLL2.dll

Please enter an address
Please specify the new image base
Input
0:10000000
OK
Cancel

References

Davis, S. (2011, October). *ApateDNS*. Retrieved from https://www.mandiant.com/blog/research-tool-release-apatedns/

Gröbert, F. (2010, 02 07). *PEiD*. Retrieved 02 18, 2014, from https://code.google.com/p/kerckhoffs/downloads/

Heaventools Software. (2009, 10 14). *Heaventools*. Retrieved from http://heaventools.com/download.htm

Hex-Rays SA. (2014, July). *Freeware Download Page*. Retrieved from https://www.hex-rays.com/index.shtml

Johnson, A. (2011, 09 16). *Resource Hacker*. Retrieved from http://www.angusj.com/resourcehacker/

Regshot Team. (2013, August). *Regshot*. Retrieved from http://sourceforge.net/projects/regshot/

Russinovich, M., & Cogswell, B. (2014, March). *Process Monitor v3.1*. Retrieved from http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx