LAB 8

Malware Behavior

Under the direction of
Dr. Samuel Liles

Table of Contents

Abstract 3
Steps of the process 4
Preparing the LAB 4
LAB 10-1, 10-3 4
Applications & Tools 4
PEiD 4
Resource Hacker 4
PE Explorer 5
Process Monitor 5
ApateDNS 5
Regshot 6
IDA 6
OllyDbg 6
WinDbg, ... 6
Issues or problems 6
Conclusions 7
Case studies 7
Review questions 7
Lab 10-1 7
Lab 10-2 11
Lab 10-3 14
References 18

Abstract

This lab is focused on MalwareAnalysis. 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 11.1 to Lab 11.3 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 8 Malware Behavior

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 11-1, 11-3

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).

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

OllyDbg, is a 32-bit assembler level analyzing debugger for Microsoft® Windows®. Emphasis on **binary code analysis** makes it particularly useful in cases where source is unavailable(Yuschuk, 2014).

WinDbg, provides full source-level debugging for the Windows kernel, kernel-mode drivers, and system services, as well as user-mode applications and drivers (Microsoft, 2014).

Show Drivers is the free command-line tool to list Drivers running on your Windows system(SecurityXploded, 2013).

Autoruns, this utility, which has the most comprehensive knowledge of auto-starting locations of any startup monitor, shows you what programs are configured to run during system bootup or login, and shows you the entries in the order Windows processes them. (Cogswell & Russinovich, 2014)

Issues or problems

Nothing so far.

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 11-1

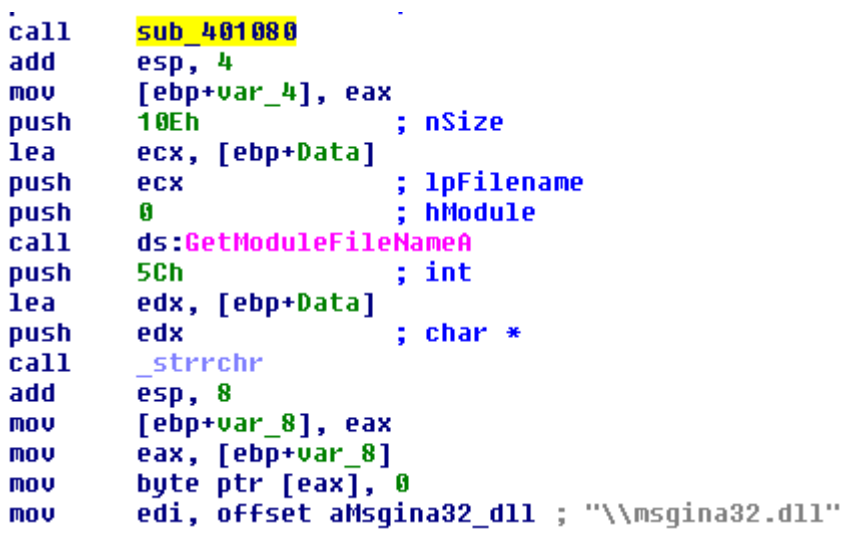
Answers	Lab11-01. exe
1	We start by static analysis running Strings with the following command Strings Lab11-01.exe -n 6 > temp.txt We view the file created and find some interesting DLL and SYS strings; ADVAPI32.dll , USER32.dll, gina.dll, KERNEL32.dll, msgina32.dll, \msgina32.dll, MSVCRT.dll, msutil32.sys Out of all those Kernel32.dll and GINA.dll are of special interest since Kernel attacks gets the highest privilage available, while Gina which is Microsoft's Graphical Identification and Authentication (GINA) interception is a techneck used to steal users credintials. Continuing on the rest of the strings we find the following RegCloseKey, RegSetValueExW, RegCreateKeyW, GetSystemDirectoryW, DisableThreadLibraryCalls, DllRegister, DllUnregister, GetCommandLineA, WriteFile,

<table>
<tr><td></td><td>SetFilePointer, CreateFileA, GinaDLL, \MSGina

From those previuse strings we can hint that the malware is going to interact with the registry, create files, create directorys, register DLLs, use the command line, set pinters. Also, the following strings show that the malware is probebly going to use the follwoing registery location in order to achive persistancy.

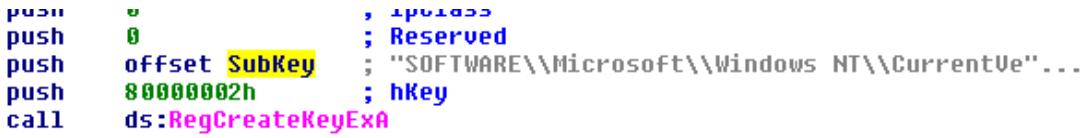
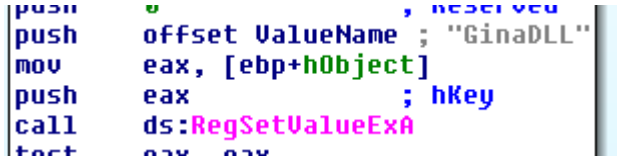
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

Next, we use IDA Pro to get a better undrestanding to the malware excution flow. Going over the code we look for the createfileA, writefileA and we find that GetModuleFileNameA is being used to get the file name which is shown as "\\msgina32.dll"



now that we have a good idea what the file is called we want to see where will the inofrmation stored in it come from so we follow the sub_401080 and we find that IpName is storing the value TGAD so the file created is getting its information from TGAD

</td></tr>
<tr><td>2</td><td>As suspected earlyer the registery will be used to achive presistancy going over the code</td></tr>
</table>

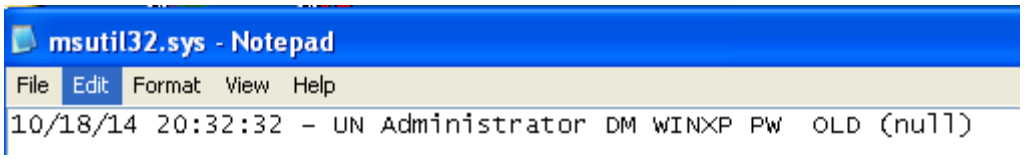
	using the Windows NT string we find the following code  This indecates that the file is creating a registery key in a location that is always going to be excuted when windows is running. Furthemore, we see the value being set as GinaDLL.  This is confirmed using regshot 
3-4	Since the Malware has included several Gina strings such as GINA DLL, msgina32.dll. therefore it is safe to say that it will use the Gina intereption method to steal user password. Which is similer to a man in the middle attack where the values are being piped through the malware before handing it to the system. However, to verify that we go over the excution code in the created file , msgina32.dll using strings and we find the file name msutil32.sys. Using IDA Pro we locate that string by doing that we know that the this function is responsible for giving the file name therefore hte information stored in it could be stored in the function that called this function which is sub_10001570 . By Using Find All occurences we look up the results to find that WlxLoggedOutSAS is the only function that requires information from it. Going over the WlxLoggedOutSAS as shown in the graph below we find the

```
mov     edi, eax
call    ??2@YAPAXI@Z    ; operator new(uint)
mov     eax, [esp+4+pProfile]
mov     esi, [esp+4+pNprNotifyInfo]
mov     ecx, [esp+4+phToken]
mov     edx, [esp+4+pdwOptions]
add     esp, 4
push    eax
mov     eax, [esp+4+pLogonSid]
push    esi
push    ecx
mov     ecx, [esp+0Ch+pAuthenticationId]
push    edx
mov     edx, [esp+10h+dwSasType]
push    eax
mov     eax, [esp+14h+pWlxContext]
```

LoignSID and AuthinicationID and other credintals that are being pushed to the stack. Going back to the code we find that this information is being printed to the file imidiatly Also, the attacker seems to use that as a log as shown in the next graph where he stores values with date and time.

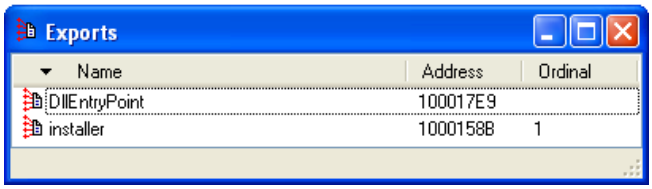
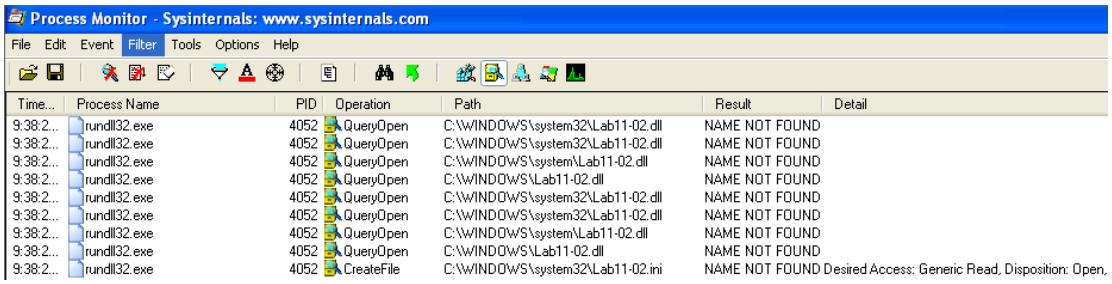
```
lea     eax, [esp+858h+var_800]
push    edi
lea     ecx, [esp+85Ch+var_850]
push    eax
push    ecx             ; wchar_t *
call    _wstrtime
add     esp, 4
lea     edx, [esp+860h+var_828]
push    eax
push    edx             ; wchar_t *
call    _wstrdate
add     esp, 4
push    eax
push    offset aSSS     ; "%s %s - %s "
push    esi             ; FILE *
call    fwprintf
mov     edi, [esp+870h+dwMessageId]
add     esp, 14h
test    edi, edi
jz      short loc_10001637
```

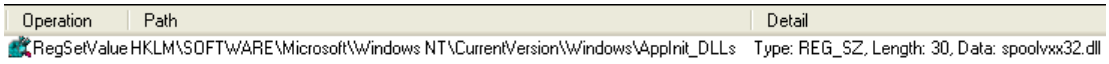
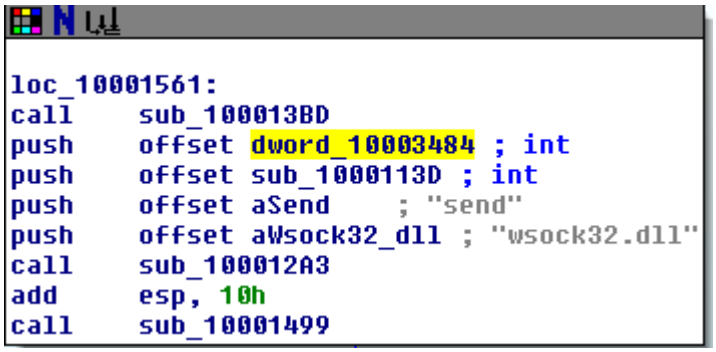
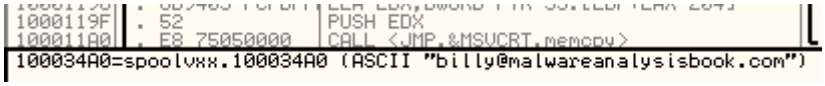
5	Going over the code there seems to be no point in which the exution is started. However since we know that the malware has added itself to an autorun registery directory we can assume that a reboot to the system will activate the malware when the user enters his information. To test that we will reboot the system and try to locate the msutil32.sys After restart we find the following file containing the credentials in the system. logged

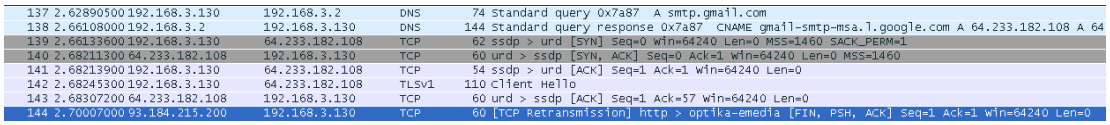
	in as we suspected by date and time. 

Lab 11-2

Answers	Lab11-02. dll, Lab11-02. ini.
1	We start by static analysis running Strings with the following command Strings Lab11-01.exe -n 6 > temp.txt We view the file created and find some interesting file name strings; spoolvxx32.dll, \spoolvxx32.dll, \Lab11-02.ini, kernel32.dll, THEBAT.EXE, OUTLOOK.EXE, MSIMN.EXE, wsock32.dll, ADVAPI32.dll, MSVCRT.dll Out of all those Kernel32.dll could be used to get Kernal privilage and the following email clients (THEBAT.EXE, OUTLOOK.EXE, MSIMN.EXE) could be used to intereact with users email, Also, wsock32.dll is probebly used for windows socket network communication. Also, the string function showen us the following; GetProcAddress, LoadLibraryA, GetSystemDirectoryA, GetModuleFileNameA, GetModuleHandleA, SuspendThread, Thread32First, CreateToolhelp32Snapshot, GetCurrentProcessId, ResumeThread, CreateFileA, RegCloseKey, RegSetValueExA RegOpenKeyExA, OpenThread, RCPT TO:, AppInit_DLLs These commands shows us the malware could be using the system directory, minipulating threads, creating files, minipulating registery keys, load DLL libraries, Use memory addresses. Also, the following strings show that the malware is probebly going to use the follwoing registery location in order to achive persistancy.

	SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows Next we try to IDA Pro to learn more about the excution flow. We find that the malware contains two exports as shown by the graph; Installer, DllEntryPoint. 
2	After our static analysis, we start Dynamic analysis and going over Procmon we find out that rundll32.exe creates a new DLL file using one of the strings we have identified earlier spoolvxx32.dll, we also find that the Rundll32.exe has tried to access the malware files in system32 direcotry and was not succesful. 
3	Based on part two, in order to properly install it we need to move the malware files to system32 directory.
4	Since the malware only created one file we use the filter in procmon using the name of the file to see if there is any process related to it is done. If the file will be presistance it will need to call the file. Using Path, details we find the following shown in the next graph. The malware is affecting a file called AppInit_DLLs. After researching it we find that "The AppInit_DLLs infrastructure provides an easy way to hook system APIs by allowing custom DLLs to be loaded into the address space of every interactive application." (Microsoft, na.) .

	Since the malware is being added to APPInit_DLLs it will be loaded to any process called by the user. 
5	Since tha malware is clearly targeting the email clients we use them to locate the function that interacts with it. By doing that we will zone in to the function that will minipulate the inputs going to those functions. This points us to the sub_100014B6 function. In the function we see that after every client is being called a function in the middle is being used which is aSend. This is Called an inline hook using the send function. 
6	In order to know exctly what the malware does we look at the code being used before the Send function whcih is sub_1000113D. we find that the malware is comparing strings to RCPT TO: and if argument is not null it adds the following address. This address was obtained by using NOPs when the comparison starts in order to see the decripted msg billy@malwareanalysisbook.com. 
7	The process that the malware attack are the 3 mail clients identified earlier (THEBAT.EXE, OUTLOOK.EXE, MSIMN.EXE). Everything in the code is using and working with emails so thats why its only targeting email clients becasue its pointless to

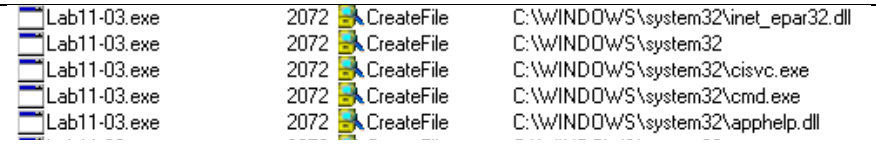
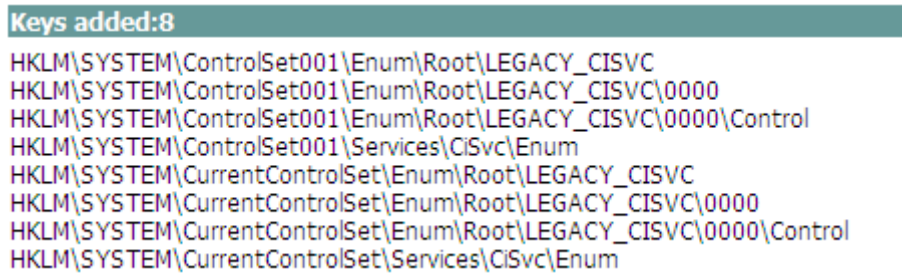
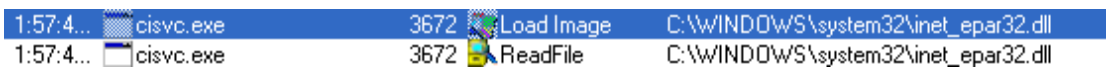
	work with others and break when there is no RCPT TO : in its values.
8	As shown in section 6 of this quastion the email was found to be billy@malwareanalysisbook.com.
9	I would set up ApateDNSI so there is a network working. Install the Bat which is one of the clients the malware interacts with. Then run wireshark and listen to msgs being sent out using TLS. 

137	2.62890500	192.168.3.130	192.168.3.2	DNS	74 Standard query 0x7a87 A smtp.gmail.com
138	2.66108000	192.168.3.2	192.168.3.130	DNS	144 Standard query response 0x7a87 CNAME gmail-smtp-msa.l.google.com A 64.233.182.108 A 64
139	2.66133600	192.168.3.130	64.233.182.108	TCP	62 ssdp > urd [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
140	2.68211300	64.233.182.108	192.168.3.130	TCP	60 urd > ssdp [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
141	2.68213900	192.168.3.130	64.233.182.108	TCP	54 ssdp > urd [ACK] Seq=1 Ack=1 Win=64240 Len=0
142	2.68245300	192.168.3.130	64.233.182.108	TLSv1	110 Client Hello
143	2.68307200	64.233.182.108	192.168.3.130	TCP	60 urd > ssdp [ACK] Seq=1 Ack=57 Win=64240 Len=0
144	2.70007000	93.184.215.200	192.168.3.130	TCP	60 [TCP Retransmission] http > optika-emedia [FIN, PSH, ACK] Seq=1 Ack=1 Win=64240 Len=0

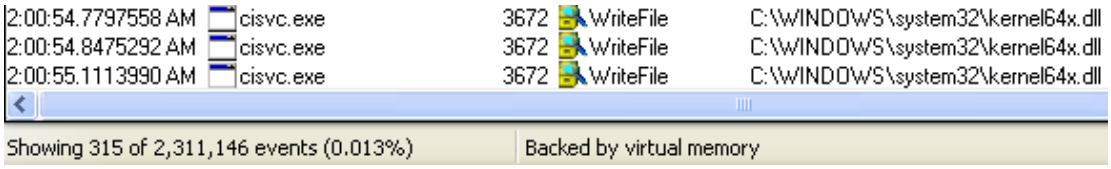
Lab 11-3

Answers	Lab11-03.exe; Lab11-03.dll
1	Running Strings we find the following interesting files cmd.exe, cisvc.exe, Lab11-03.dll, user32.dll, command.com, KERNEL32.dll, Which shows us that the command line might be used. the Library will be loaded. and cisvc.exe is a background process that works as an Index service and tracks files. Moreover, we see the following interesting line net start cisvc which is a command used on a command line that adds to the fact that we had a cmd.exe string earlier. Also, the following paths have been found C:\WINDOWS\System32\%s C:\WINDOWS\System32\inet_epar32.dll Also we find the following names or functions that can be used; GetLastActivePopup, GetActiveWindow, MessageBoxA, IsBadReadPtr, UnmapViewOfFile, CreateFileMappingA, GetFileSize, CreateFileA, CopyFileA,

	GetCommandLineA, ExitProcess, TerminateProcess, GetCurrentProcess, GetFileAttributesA, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, GetEnvironmentStringsW, GetStdHandle, GetFileType, GetStartupInfoA, GetModuleHandleA, GetEnvironmentVariableA, GetVersionExA, WriteFile, GetExitCodeProcess, WaitForSingleObject, CreateProcessA, SetFilePointer, GetProcAddress, LoadLibraryA, GetStringTypeA, GetStringTypeW, CompareStringA, CompareStringW, SetEnvironmentVariableA, Out of all this we see lots A's and W's which means the malware is going to be doing a lot of comparison or dealing with similer objects at the same time. Also, we see file handling, string handling, creating files and process, change evniromnatal variables, terminating processes, deal with windows and popups, copy files. Now running string on the DLL file we find the following files user32.dll, Lab1103dll.dll, KERNEL32.dll we also see lots of formating strings like; H:mm:ss , dddd, MMMM dd, yyyy, M/d/yy We also find the 12 months name as well as the weekdays. Also, we see the following path C:\WINDOWS\System32\kernel64x.dll and the name kernel64x.dll We also see <SHIFT> which is not something that would be there however with the months and days it might hint that this might be used as a keylogger. Furthemore, we see the following two strings; CreateMutexA, OpenMutexA. which means the malware will probebly hijack a process that is already running. Finally we see Sleep, which means the malware will run in cycles be active then sleep. So this could be a malware that keylogs files, use command line, and popup windows, and sleep in between all those.
2	The malware popup a command line showing that indexing servicses has been started,

	 We also see the following files have been created including Inet_epar32.dll, cisvc.exe, cmd.exe, apphelp.dll. However, the ones that are being writen to are  Which includes Kernel64x.dll and while going over procmon we see a lot of writing to this single file which probebly means its a keylogger program updating the file.
3	Running SC query we find the following file which was created by the malware  We also check regshot we find 8 keys created  Under services CiSvc has been added. whcih means its going to be run everytime the system is retareted.
4	The malware infects cisvc.exe which when we filter procmon shows us lots of writes and loads to other files such as kernel64x.dll as well as loading inet_epar32.dll 

5	Using IDA Pro on the Lab11-03.dll we find that there is a single exporting function which is called zzz69806582 that had a single function call which is creating threads .
	 
	`public zzz69806582`
	`zzz69806582 proc near`
	`var_4= dword ptr -4`
	`push ebp`
	`mov ebp, esp`
	`push ecx`
	`push 0 ; lpThreadId`
	`push 0 ; dwCreationFlags`
	`push 0 ; lpParameter`
	`push offset StartAddress ; lpStartAddress`
	`push 0 ; dwStackSize`
	`push 0 ; lpThreadAttributes`
	`call ds:CreateThread`

6	We cam see in less than 3 minutes we had more than 315 write commands by cisvc.exe to the file kernel64x.dll on procmon.
	 
	2:00:54.7797558 AM cisvc.exe 3672 WriteFile C:\WINDOWS\system32\kernel64x.dll
	2:00:54.8475292 AM cisvc.exe 3672 WriteFile C:\WINDOWS\system32\kernel64x.dll
	2:00:55.1113990 AM cisvc.exe 3672 WriteFile C:\WINDOWS\system32\kernel64x.dll
	Showing 315 of 2,311,146 events (0.013%) Backed by virtual memory
	After examing the file we see the following
	`C:\WINDOWS\system32\cmd.exe - sc query "CiSvc": 0xd`
	`C:\WINDOWS\system32\cmd.exe:`
	Which are some of the commands that has been excuted while analyzing the malware

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/

Microsoft. (2014, na.). *Download Center*. Retrieved from http://www.microsoft.com/en-us/download/confirmation.aspx?id=8279

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

Yuschuk, O. (2014, Feb). *OllyDbg* . Retrieved from http://www.ollydbg.de/