CNIT 58100 CFM: CYBERFORENSICS OF MALWARE – LAB 11

Ibrahim Waziri Jr

PhD in Information Security (CERIAS)

Lab 11

Instructor: **Associate Prof Sam Liles**

Purdue University

2014

Abstract

This lab covers the skills discussed in chapter 11 of the text. The practice covered in these labs is all based on malware analysis. The malware files used are provided as an extension of the text for practical purposes.

Each of the labs consists of multiple questions that require short answers. Depending on the question, certain special tools might be required to fully analyze the malware and find answers to the question.

This paper provides answers to Chapter 11 labs. The lab uses 3 different files which are: *Lab11-01.exe, Lab11-02.exe and Lab11-03.exe.* These files are malwares are therefore could be harmful if used for non-training purposes.

Lab 11-1

1. The malware extracts the file msgina32.dll. First we start by loading the *Lab11-01.exe* into PEview to start a basic static analysis as shown Fig 1 below:

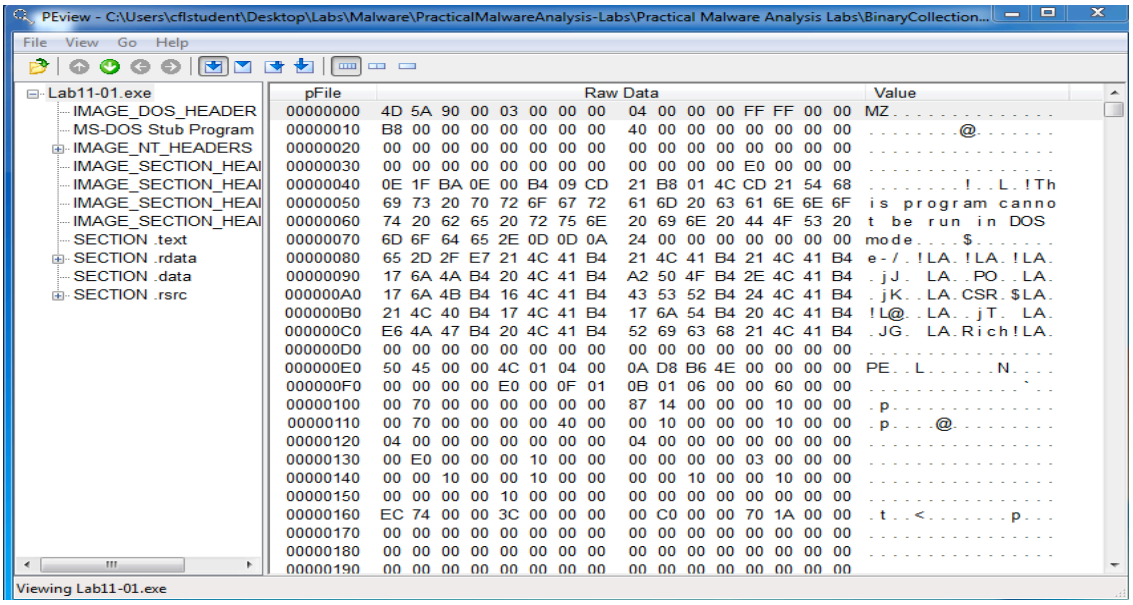


Fig 1: Lab11-01.exe in PEview

Extending the BINARY TGAD 0000 we can see that the malware has an embedded PE file, to analyze that we have to perform a dynamic analysis and monitor the malware using procmon as shown in Fig 2 below:

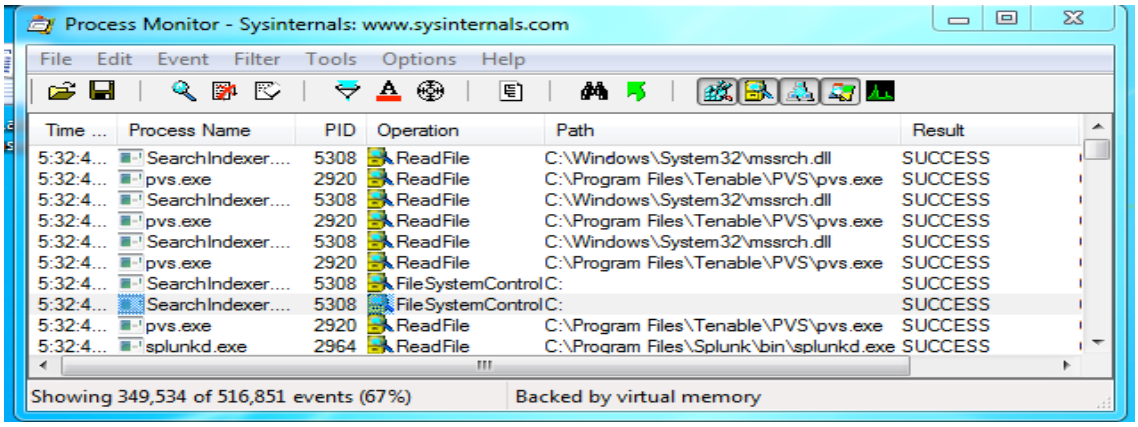


Fig 2: Running Procmon

Running the malware creates a file msgina32.dll as shown in Fig 3 below:

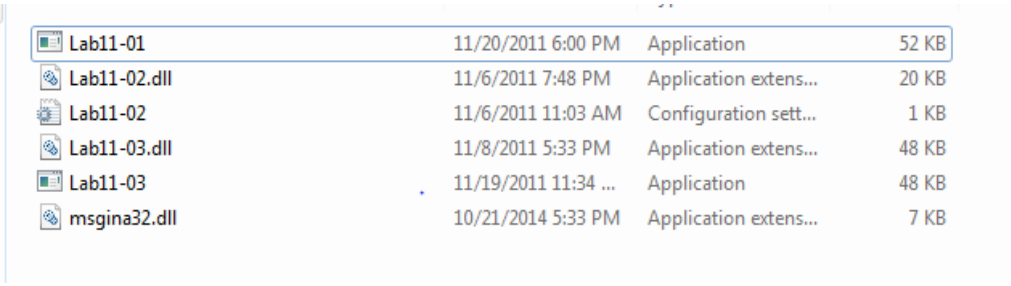


Fig 3: msgina32.dll

Extracting the file using resource hacker as shown in Fig 4 below, and comparing it to msgina32.dll shown in fig 1 above, we see that the two file are identical.

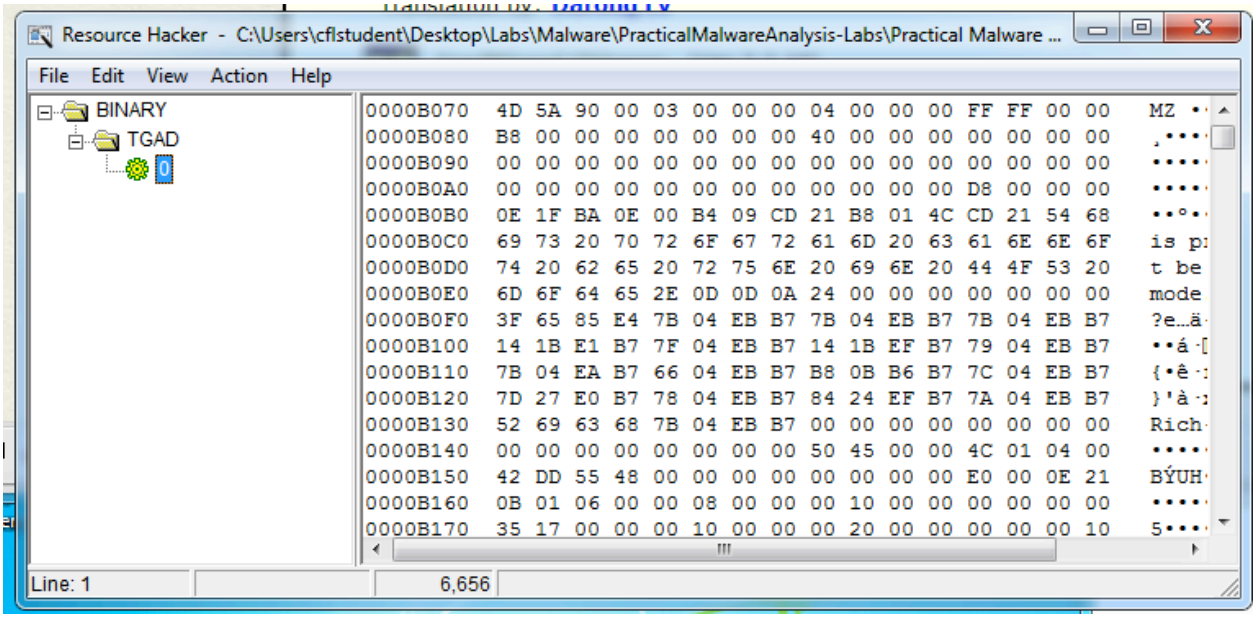


Fig 4: Resource Hacker

We then load the same malware into IDA Pro to confirm the similarities of our findings, we can see from fig 5 and 6 below, that the main function calls two functions: sub_401080 which extracts TGAD resource section to msgina32.dll and sub_401000 which sets the GINA registry value.

```
rep movsb
push    offset aWb      ; "wb"
push    offset aMsgina32_dll_0 ; "msgina32.dll"
call    _fopen
add     esp, 8
```

Fig 5: Sub_401080

```
; int __cdecl sub_401000(BYTE *lpData,DWORD cbData)
sub_401000 proc near

hObject= dword ptr -4
lpData= dword ptr  8
cbData= dword ptr  0Ch

push    ebp
mov     ebp, esp
push    ecx
push    0               ; lpdwDisposition
lea     eax, [ebp+hObject]
push    eax             ; phkResult
push    0               ; lpSecurityAttributes
push    0F003Fh         ; samDesired
push    0               ; dwOptions
push    0               ; lpClass
push    0               ; Reserved
push    offset SubKey   ; "SOFTWARE\\Microsoft\\Windows NT\\CurrentVe"...
push    80000002h       ; hKey
call    ds:RegCreateKeyExA
test    eax, eax
```

Fig 6: sub_401000

From this we can say that the malware extracts and drops the file msgina32.dll onto disk from a resource section named TGAD.

2. The malware installs msgina32.dll as a GINA DLL by adding it to the registry location HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL and that causes the DLL to be loaded after a restart. To verify this we first analyze the msgina32.dll

   We begin by loading the malware into IDA Pro and analyzing the DLL main as shown in Fig 7 below:

```
.text:10001054                  sub     esp, 208h
.text:1000105A                  cmp     eax, 1
.text:1000105D                  jnz     short loc_100010B7
.text:1000105F                  push    esi
.text:10001060                  mov     esi, [esp+20Ch+hLibModule]
.text:10001067                  push    esi             ; hLibModule
.text:10001068                  call    ds:DisableThreadLibraryCalls
.text:1000106E                  lea     eax, [esp+20Ch+LibFileName]
.text:10001072                  push    104h            ; uSize
.text:10001077                  push    eax             ; lpBuffer
.text:10001078                  mov     dword_100033F0, esi
.text:1000107E                  call    ds:GetSystemDirectoryW
.text:10001084                  lea     ecx, [esp+20Ch+LibFileName]
```

Fig 7: DLLmain of msgina32.dll

As shown above, the DLL first checks the fdwReason argument passed in to indicate why the DLL entry-point function is being called. The malware then checks for DLL_PROCESS_ATTACH, which is called when a process is starting up or when LoadLibrary is used to load the DLL.

If a different DllMain is called during the a DLL_PROCESS_ATTACH, the code shown in Fig 8: is called. The copde gets a handle to msgina.dll nt he Windoes systems directory via the call to LoadLibrary.

```
.text:10001078                  mov     dword_100033F0, esi
.text:1000107E                  call    ds:GetSystemDirectoryW
.text:10001084                  lea     ecx, [esp+20Ch+LibFileName]
.text:10001088                  push    offset String2  ; "\\MSGina"
.text:1000108D                  push    ecx             ; lpString1
.text:1000108E                  call    ds:lstrcatW
.text:10001094                  lea     edx, [esp+20Ch+LibFileName]
.text:10001098                  push    edx             ; lpLibFileName
.text:10001099                  call    ds:LoadLibraryW
.text:1000109F                  xor     ecx, ecx
.text:100010A1                  mov     hModule, eax
.text:100010A6                  test    eax, eax
.text:100010A8                  setnz   cl
.text:100010AB                  mov     eax, ecx
.text:100010AD                  pop     esi
.text:100010AE                  add     esp, 208h
.text:100010B4                  retn    0Ch
.text:100010B7 ; ---------------------------------------------------------------------------
.text:100010B7
.text:100010B7 loc_100010B7:                            ; CODE XREF: DllMain(x,x,x)+D↑j
```

Fig 8: DLLMain of msgina32.dll

3. The malware appears to be stealing credentials by performing GINA interception. msgina32.dll intercept all user credentials submitted to the system for authentication.

   From Fig 8 above, we can see that a coded handles to msgina.dll in the Windows system directory via the call to LoadLibraryW at 10001099. The malware saves the handle in a global variable that IDA Pro calls hModule at 100010A1. The use of this variable allows the DLL's exports to properly call functions in the msgina.dll Windows DLL. We analyze each export function beginning with WlxLoggedOnSAS as shown in Fig 9 below:

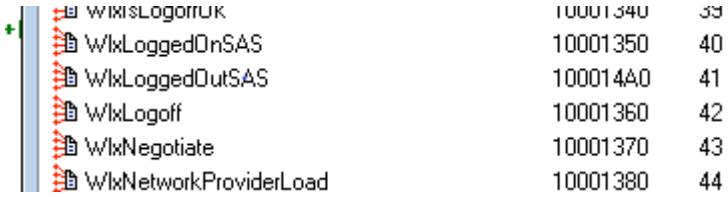


Fig 9: Export

   The WlxLoggedOnSAS export is short and simply passes through to the true WlxLoggedOnSAS contained in msgina.dll. There are 2 Wlx functions which are: WlxLoggedOnSAS and WlxLoggedOutSAS. The export analyzes WlxLoggedOutSAS within msgina.dll using GetProcAddress and then calling it.

4. The malware logs stolen credentials to %systemRoot%\System32\msutil32.sys.

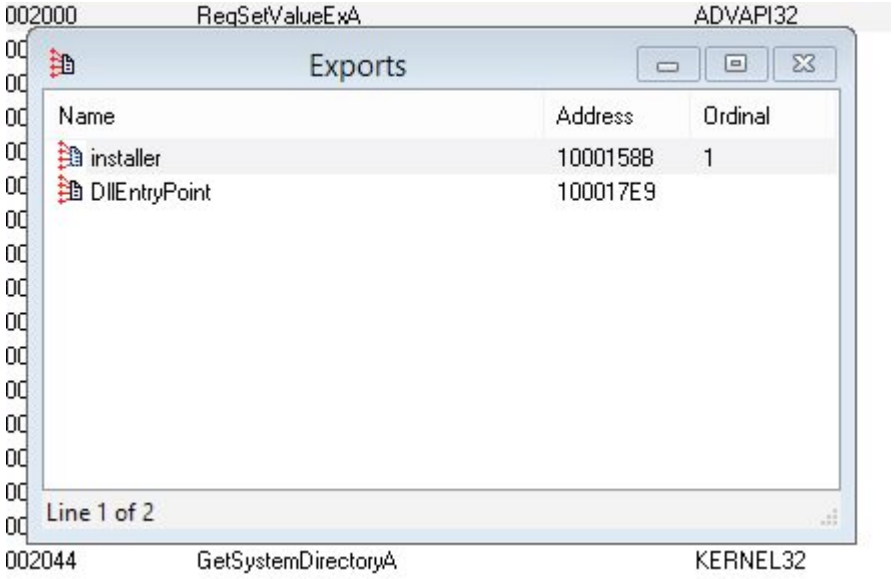
```
.text:1000158E          call    _vsnwprintf
.text:10001593          push    offset word_10003320 ; wchar_t *
.text:10001598          push    offset aMsutil32_sys ; "msutil32.sys"
.text:1000159D          call    _wfopen
.text:100015A2          mov     esi, eax
.text:100015A4          add     esp, 18h
.text:100015A7          test    esi, esi
.text:100015A9          jz      loc_1000164F
.text:100015AF          lea     eax, [esp+858h+var_800]
.text:100015B3          push    edi
.text:100015B4          lea     ecx, [esp+85Ch+var_850]
.text:100015B8          push    eax
.text:100015B9          push    ecx             ; wchar_t *
.text:100015BA          call    _wstrtime
.text:100015BF          add     esp, 4
.text:100015C2          lea     edx, [esp+860h+var_828]
.text:100015C6          push    eax
.text:100015C7          push    edx             ; wchar_t *
.text:100015C8          call    _wstrdate
.text:100015CD          add     esp, 4
.text:100015D0          push    eax
.text:100015D1          push    offset aSSS     ; "%s %s - %s "
.text:100015D6          push    esi             ; FILE *
.text:100015D7          call    fwprintf
```

As shown in Fig 10 above, the call to vsnwprintf fills in the format string passed in by WlxLoggedOutSAS. The malware opens the file msutil32.sys which is created under system 32, and also records the date, time and information logged.

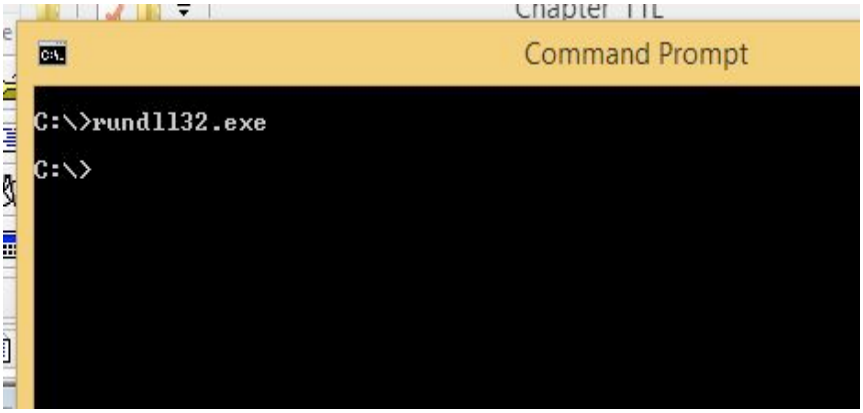
5. The system must be rebooted for this question to be answered, considering the virtual machine in the lab is set to wipe out as soon as the computer restarts, answering this question will be difficult.

Lab 11-2

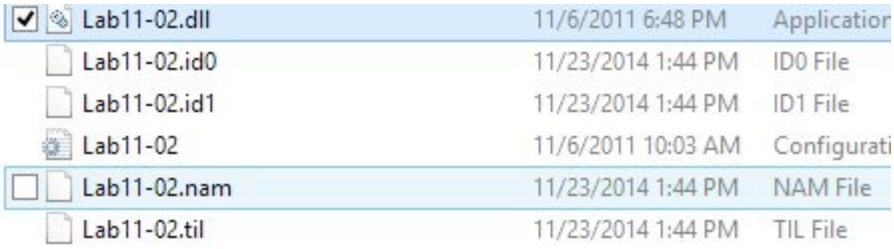
1. From IDAPro we can see that the program has 2 exports, installer & DllEntryPoint as shown in Figure below:



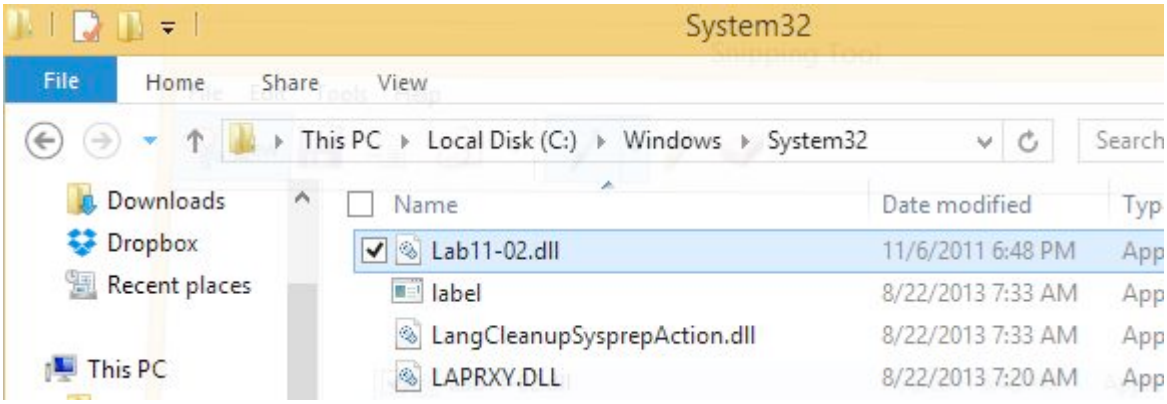
2. An attempt to install the malware using rundll32.exe resulted to my computer and the whole virtual machine crashing and showing a blue screen. Another attempt was made to install the malware using cmd, and as shown below and it seems to be running with no problems.



However we couldn't get the any file, but it seems like it creates other files as shown in the figure below:



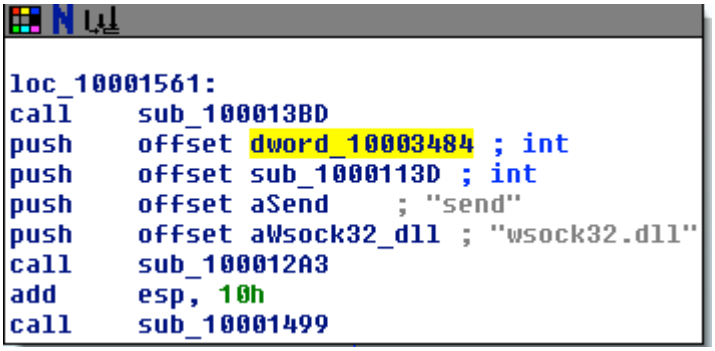
We decided to move the file to System32 to see what it does, as shown in the figure below:



3. After answering question 2 above, we realize that lab11-02.dll must reside in C:\Windows\System32 for the program to install properly

4. No it is not installed for persistence considering it has to be moved to a certain location (C:\Windows\System32) for it to be installed.

5. Using IDAPro as shown in the figure below: It looks like the malware uses a function SEND as the user-space rootkit technique by using a file probably which it creates known as wsock32.dll

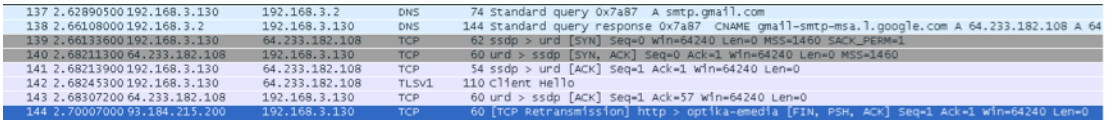


6. If by hooking code, it means the SEND function discussed in 5 above, then the SEND function is used to make the program to send an instruction it is designed to. (Malware function yet to be analyzed).

7. The malware seems to have an email function, because it is targeting MSIMN.exe, THEBAT.exe and OUTLOOK.exe. All those files are email clients

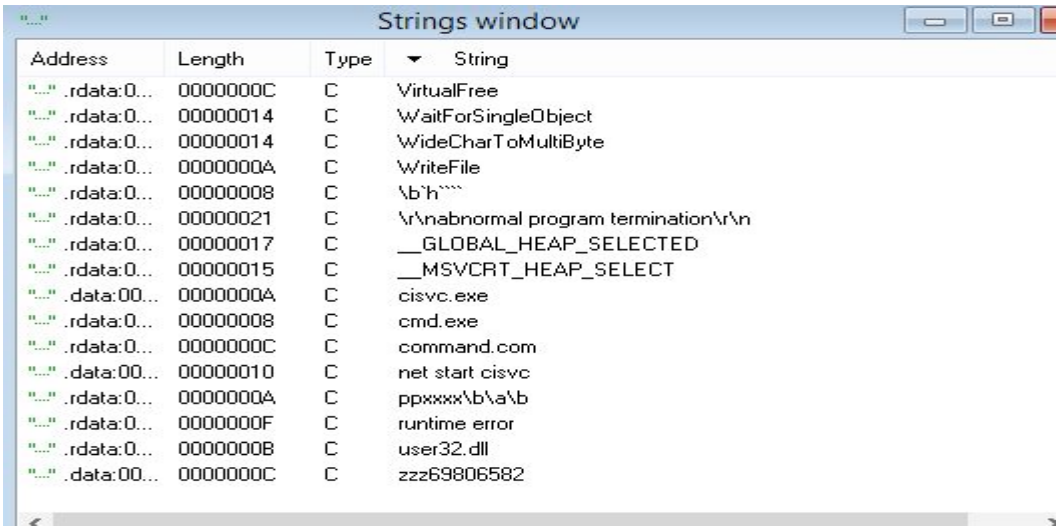
8. It looks like the Lab11-02.ini contains an email billy@malwareanalysisbook.com as shown in the figure below

9. To capture the malware dynamically, run the Lab11-02.exe, rundll32.exe and the installer file. The result is shown in Figure below:

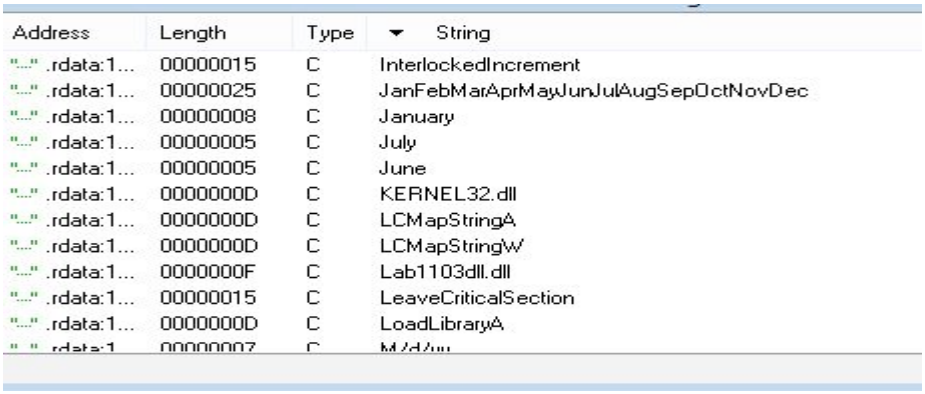


Lab 11-3

1. Lab11-03.exe contains the strings inet_epar32.dll and net start cisvc as shown in figure below: This tells us that it might start the CiSvc indexing service.



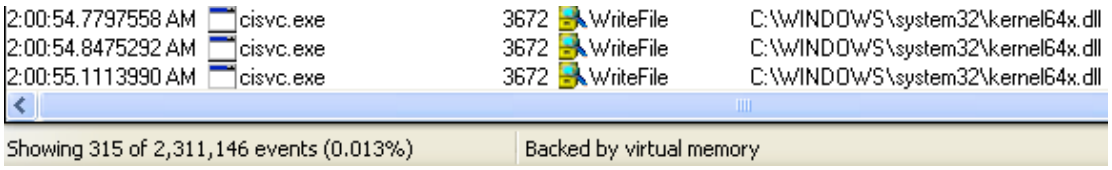
Also the file Lab11-03.dll contains the strings kernel32.dll as shown in Figure below, and it is located in System32. This tells us that the malware writes data to cisvc.exe and starts the indexing service.



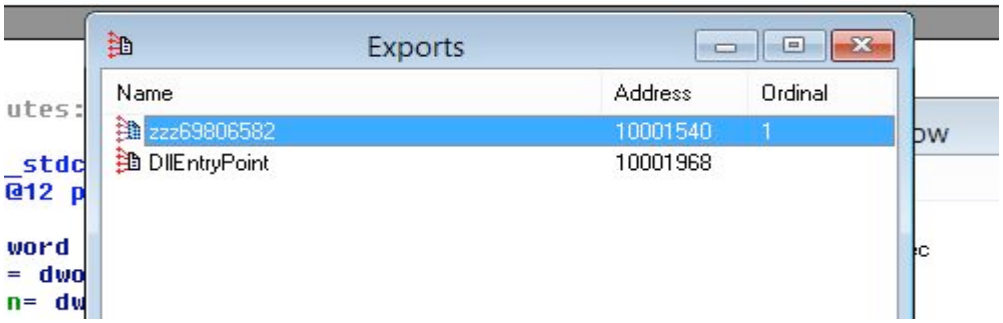
2. After running the malware, it seems like it writes data and starts indexing service. The malware appears to replicate itself and writes to C:\Windows\System32. However we cant seems to locate the file.

3. It appears the malware persistently installs Lab11-03.dll by trojanizing the indexing service by entry-point direction. It redirects entry point to run shellcode and then load the DLL.

4. Using procmon as shown in figure below we can tell that the malware infects cisvc.exe.



5. From IDAPro we can see that is has an export zzz69806582. It appears that the malware infects cisvc.exe to load inet_epar32.dll and call its export.



6. From question 1 above, we can tell that it stores the data it collects in System32.

Conclusion

These labs aim to teach how to understand malware behaviors and familiarize us with the most common characteristics of software that identifies it to be a malware.