CNIT 58100 CFM: CYBERFORENSICS OF MALWARE – LAB 12

Ibrahim Waziri Jr

PhD in Information Security (CERIAS)

Lab 12

Instructor: **Associate Prof Sam Liles**

Purdue University

2014

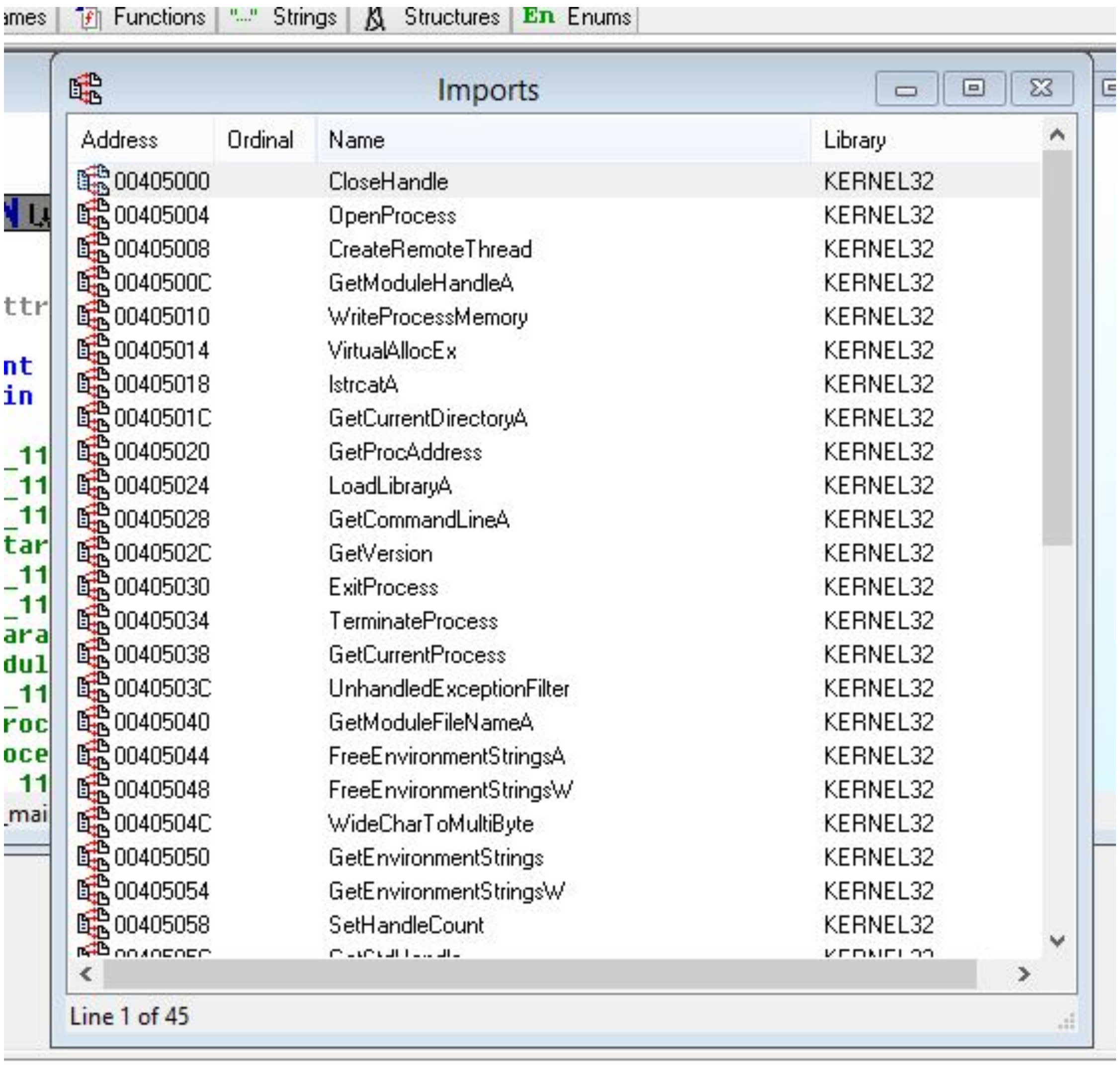
Abstract

This lab covers the skills discussed in chapter 12 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 12 labs. The lab uses 3 different files which are: *Lab12-01.exe, Lab12-02.exe and Lab12-03.exe.* These files are malwares are therefore could be harmful if used for non-training purposes.

Lab 12-1

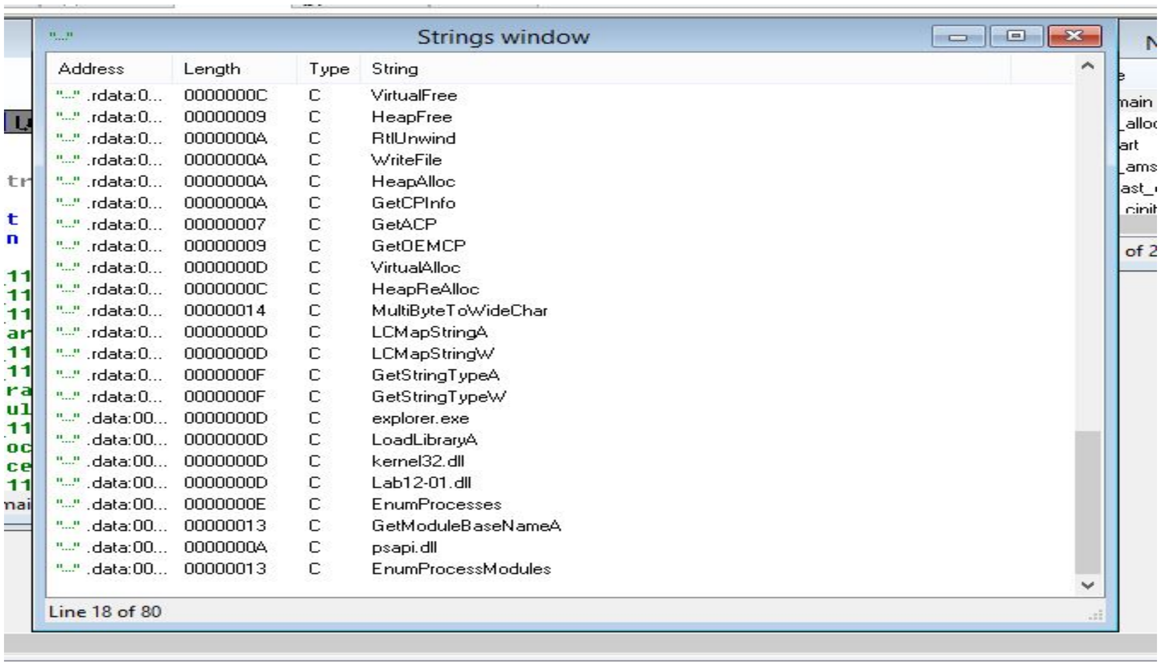
1. Running a basic static analysis and checking out the imports for Lab12-01.exe using IDAPro as shown below, we can see imports like CreateRemoteThread, WriteProcessMemory and VirtualAllocEx. However when we run the malware, pop-up messages are displayed on the screen every minute.



Address	Ordinal	Name	Library
00405000		CloseHandle	KERNEL32
00405004		OpenProcess	KERNEL32
00405008		CreateRemoteThread	KERNEL32
0040500C		GetModuleHandleA	KERNEL32
00405010		WriteProcessMemory	KERNEL32
00405014		VirtualAllocEx	KERNEL32
00405018		lstrcatA	KERNEL32
0040501C		GetCurrentDirectoryA	KERNEL32
00405020		GetProcAddress	KERNEL32
00405024		LoadLibraryA	KERNEL32
00405028		GetCommandLineA	KERNEL32
0040502C		GetVersion	KERNEL32
00405030		ExitProcess	KERNEL32
00405034		TerminateProcess	KERNEL32
00405038		GetCurrentProcess	KERNEL32
0040503C		UnhandledExceptionFilter	KERNEL32
00405040		GetModuleFileNameA	KERNEL32
00405044		FreeEnvironmentStringsA	KERNEL32
00405048		FreeEnvironmentStringsW	KERNEL32
0040504C		WideCharToMultiByte	KERNEL32
00405050		GetEnvironmentStrings	KERNEL32
00405054		GetEnvironmentStringsW	KERNEL32
00405058		SetHandleCount	KERNEL32

Line 1 of 45

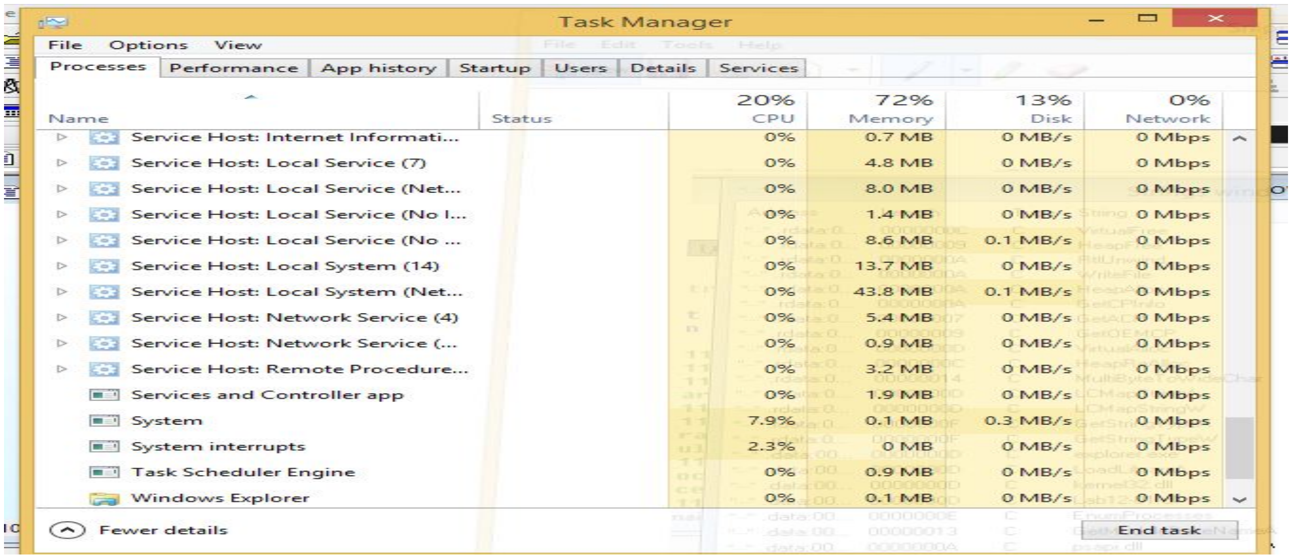
2. Examining the strings in the malware, we see some notable strings like explorer.exe, lab12-01.dll and psapi.dll.



```
Address          Length     Type  String
"..." .rdata:0...  0000000C   C     VirtualFree
"..." .rdata:0...  00000009   C     HeapFree
"..." .rdata:0...  0000000A   C     RtlUnwind
"..." .rdata:0...  0000000A   C     WriteFile
"..." .rdata:0...  0000000A   C     HeapAlloc
"..." .rdata:0...  0000000A   C     GetCPInfo
"..." .rdata:0...  00000007   C     GetACP
"..." .rdata:0...  00000009   C     GetOEMCP
"..." .rdata:0...  0000000D   C     VirtualAlloc
"..." .rdata:0...  0000000C   C     HeapReAlloc
"..." .rdata:0...  00000014   C     MultiByteToWideChar
"..." .rdata:0...  0000000D   C     LCMapStringA
"..." .rdata:0...  0000000D   C     LCMapStringW
"..." .rdata:0...  0000000F   C     GetStringTypeA
"..." .rdata:0...  0000000F   C     GetStringTypeW
"..." .data:00...  0000000D   C     explorer.exe
"..." .data:00...  0000000D   C     LoadLibraryA
"..." .data:00...  0000000D   C     kernel32.dll
"..." .data:00...  0000000D   C     Lab12-01.dll
"..." .data:00...  0000000E   C     EnumProcesses
"..." .data:00...  00000013   C     GetModuleBaseNameA
"..." .data:00...  0000000A   C     psapi.dll
"..." .data:00...  00000013   C     EnumProcessModules
Line 18 of 80
```

The process injected is explorer.exe, because we can see from the system indicator as shown in the task manager utility shown in question 3 below the CPU rate is high.

3. The only thing that worked for me to successfully stop the pop-ups is running the task manager utility and killing the system process as shown below. (For some reason, it shows system instead of explorer.exe, further analysis to see why the name change didn't yield any result).

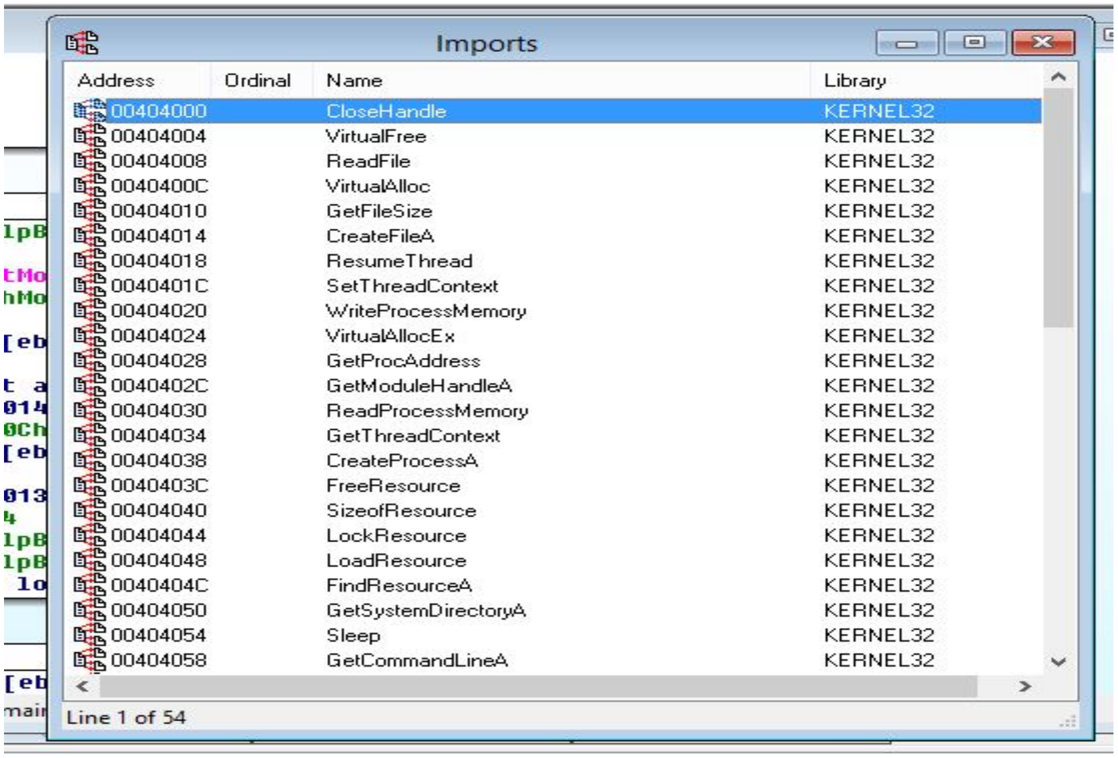

Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task

Name	Status	20% CPU	72% Memory	13% Disk	0% Network
Service Host: Internet Informati...		0%	0.7 MB	0 MB/s	0 Mbps
Service Host: Local Service (7)		0%	4.8 MB	0 MB/s	0 Mbps
Service Host: Local Service (Net...		0%	8.0 MB	0 MB/s	0 Mbps
Service Host: Local Service (No I...		0%	1.4 MB	0 MB/s	0 Mbps
Service Host: Local Service (No ...		0%	8.6 MB	0.1 MB/s	0 Mbps
Service Host: Local System (14)		0%	13.7 MB	0 MB/s	0 Mbps
Service Host: Local System (Net...		0%	43.8 MB	0.1 MB/s	0 Mbps
Service Host: Network Service (4)		0%	5.4 MB	0 MB/s	0 Mbps
Service Host: Network Service (...		0%	0.9 MB	0 MB/s	0 Mbps
Service Host: Remote Procedure...		0%	3.2 MB	0 MB/s	0 Mbps
Services and Controller app		0%	1.9 MB	0 MB/s	0 Mbps
System		7.9%	0.1 MB	0.3 MB/s	0 Mbps
System interrupts		2.3%	0 MB	0 MB/s	0 Mbps
Task Scheduler Engine		0%	0.9 MB	0 MB/s	0 Mbps
Windows Explorer		0%	0.1 MB	0 MB/s	0 Mbps

4. The malware performs DLL injection to launch Lab12-01.dll within explorer.exe. Once Lab12-01.dll is injected, it displays a message box on the screen every minute with a counter that shows how many minutes have elapsed.

Lab 12-2

1. As usual we start by some analysis to determine what the program does, and looking at the imports using IDAPro as shown in the figure below:

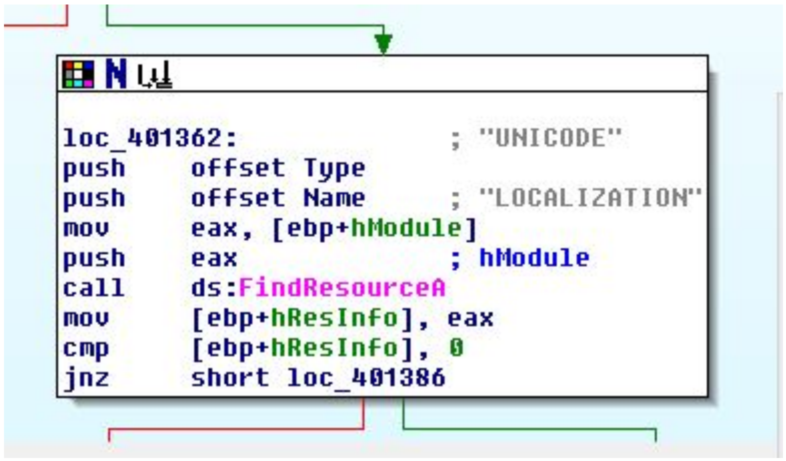


Address	Ordinal	Name	Library
00404000		CloseHandle	KERNEL32
00404004		VirtualFree	KERNEL32
00404008		ReadFile	KERNEL32
0040400C		VirtualAlloc	KERNEL32
00404010		GetFileSize	KERNEL32
00404014		CreateFileA	KERNEL32
00404018		ResumeThread	KERNEL32
0040401C		SetThreadContext	KERNEL32
00404020		WriteProcessMemory	KERNEL32
00404024		VirtualAllocEx	KERNEL32
00404028		GetProcAddress	KERNEL32
0040402C		GetModuleHandleA	KERNEL32
00404030		ReadProcessMemory	KERNEL32
00404034		GetThreadContext	KERNEL32
00404038		CreateProcessA	KERNEL32
0040403C		FreeResource	KERNEL32
00404040		SizeofResource	KERNEL32
00404044		LockResource	KERNEL32
00404048		LoadResource	KERNEL32
0040404C		FindResourceA	KERNEL32
00404050		GetSystemDirectoryA	KERNEL32
00404054		Sleep	KERNEL32
00404058		GetCommandLineA	KERNEL32

Seeing imports like CreateProcessA, GEtThreadContext, and SetThreadContext indicates that this program creates new processes and is modifying the execution context of process. The imports ReadProcessMemory and WriteProcessMemory tells us that the program is reading and writing directly to process memory spaces. The imports LockResource and SizeOfResource tells us where data important to the process may be stored.

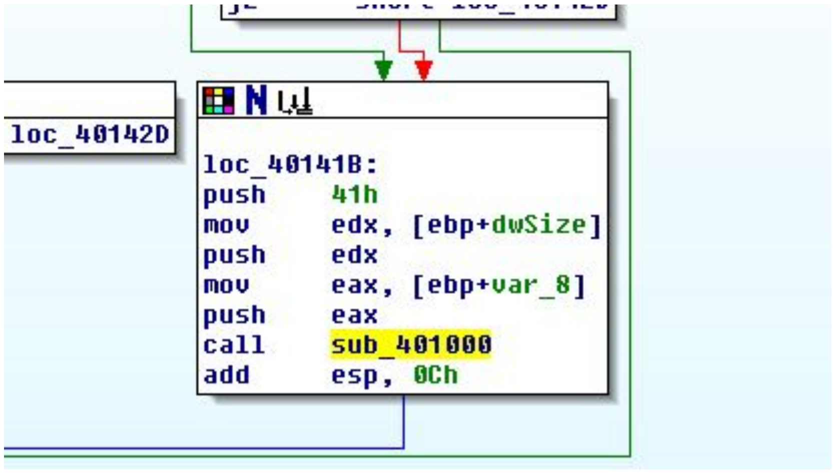
2. From all indications the program hides execution by replacing running process by its own running process.

3. As shown in figure below from IDAPro, the malicious payload is stored in the programs resource section. The resource has type UNICODE and the name LOCALIZATION.



```
loc_401362:                 ; "UNICODE"
push    offset Type
push    offset Name         ; "LOCALIZATION"
mov     eax, [ebp+hModule]
push    eax                 ; hModule
call    ds:FindResourceA
mov     [ebp+hResInfo], eax
cmp     [ebp+hResInfo], 0
jnz     short loc_401386
```

4. Searching for the 0x0040141B address using IDAPro, it seems like the malicious payload stored in the program's resource section is XOR-encoded as shown in the figure below:



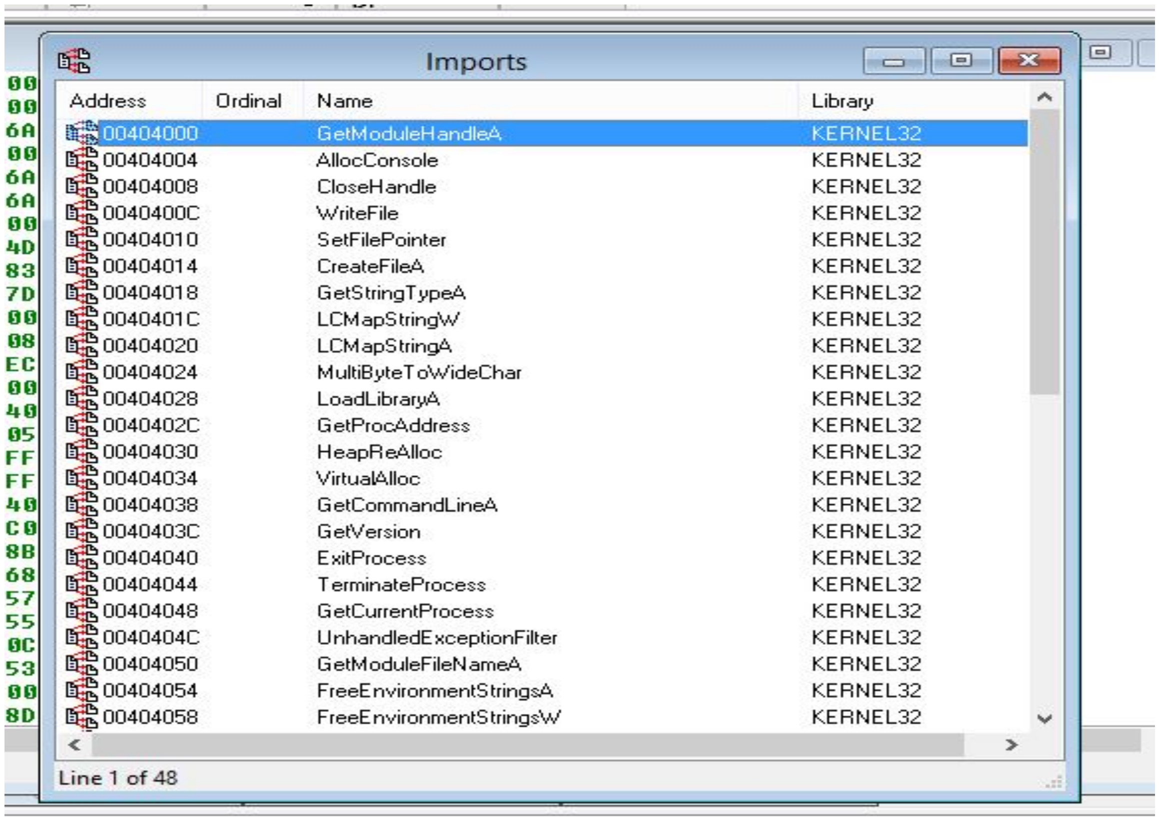
```
loc_40142D

loc_40141B:
push    41h
mov     edx, [ebp+dwSize]
push    edx
mov     eax, [ebp+var_8]
push    eax
call    sub_401000
add     esp, 0Ch
```

5. From figure above, and seeing the call, we can tell that the strings are also XOR encoded using the sub function sub_401000.

Lab 12-3

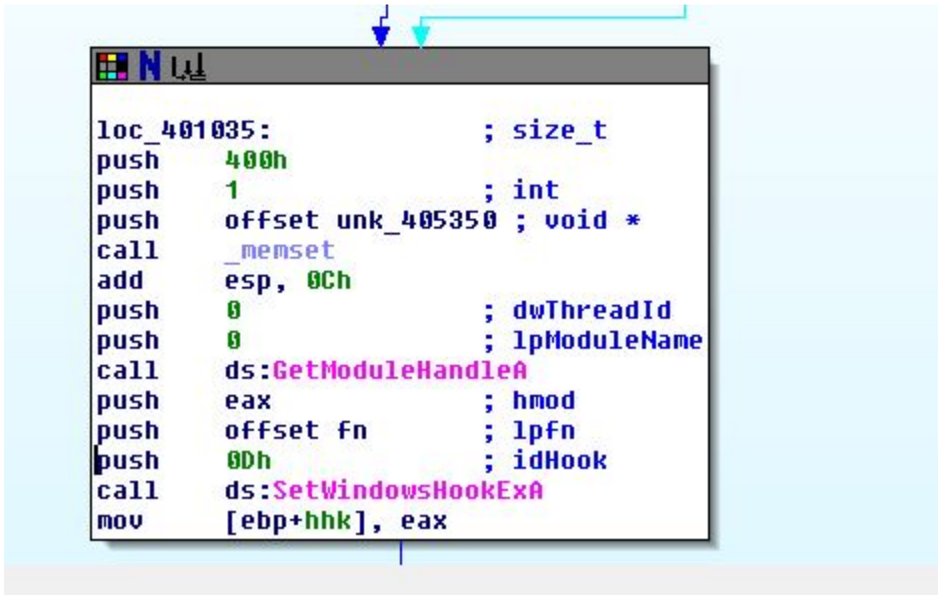
1. Looking at the imports using IDAPro and seeing imports like SetWindowsHookExA, we can tell that the program is a keylogger as shown below:



Address	Ordinal	Name	Library
00404000		GetModuleHandleA	KERNEL32
00404004		AllocConsole	KERNEL32
00404008		CloseHandle	KERNEL32
0040400C		WriteFile	KERNEL32
00404010		SetFilePointer	KERNEL32
00404014		CreateFileA	KERNEL32
00404018		GetStringTypeA	KERNEL32
0040401C		LCMapStringW	KERNEL32
00404020		LCMapStringA	KERNEL32
00404024		MultiByteToWideChar	KERNEL32
00404028		LoadLibraryA	KERNEL32
0040402C		GetProcAddress	KERNEL32
00404030		HeapReAlloc	KERNEL32
00404034		VirtualAlloc	KERNEL32
00404038		GetCommandLineA	KERNEL32
0040403C		GetVersion	KERNEL32
00404040		ExitProcess	KERNEL32
00404044		TerminateProcess	KERNEL32
00404048		GetCurrentProcess	KERNEL32
0040404C		UnhandledExceptionFilter	KERNEL32
00404050		GetModuleFileNameA	KERNEL32
00404054		FreeEnvironmentStringsA	KERNEL32
00404058		FreeEnvironmentStringsW	KERNEL32

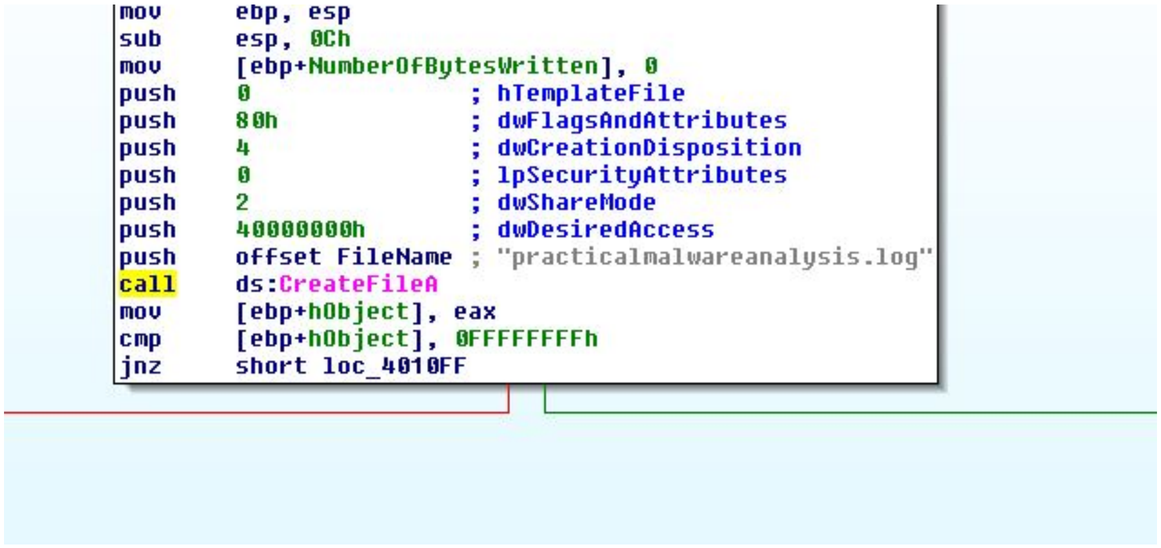
Line 1 of 48

2. From the figure shown below, we can tell that the program uses hook injection to steal keystrokes.



```
loc_401035:                 ; size_t
push    400h
push    1                   ; int
push    offset unk_405350 ; void *
call    _memset
add     esp, 0Ch
push    0                   ; dwThreadId
push    0                   ; lpModuleName
call    ds:GetModuleHandleA
push    eax                 ; hmod
push    offset fn           ; lpfn
push    0Dh                 ; idHook
call    ds:SetWindowsHookExA
mov     [ebp+hhk], eax
```

3. With reference to figure below: The program uses the file practicalmalwareanalysis.log to store the keystrokes.



```
mov     ebp, esp
sub     esp, 0Ch
mov     [ebp+NumberOfBytesWritten], 0
push    0                ; hTemplateFile
push    80h              ; dwFlagsAndAttributes
push    4                ; dwCreationDisposition
push    0                ; lpSecurityAttributes
push    2                ; dwShareMode
push    40000000h        ; dwDesiredAccess
push    offset FileName  ; "practicalmalwareanalysis.log"
call    ds:CreateFileA
mov     [ebp+hObject], eax
cmp     [ebp+hObject], 0FFFFFFFFh
jnz     short loc_4010FF
```

Conclusion

These labs aim to teach how to understand the methods malware authors use to avoid detection, which is called covert launching techniques. From these labs we learned how to recognize code constructs and other coding patterns that helped us identified way that malware is covertly launched.