LAB 9

Malware Launchers

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 12.1 to Lab 12.4 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 9 Malware Launchers

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 12-1, 12-4

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).

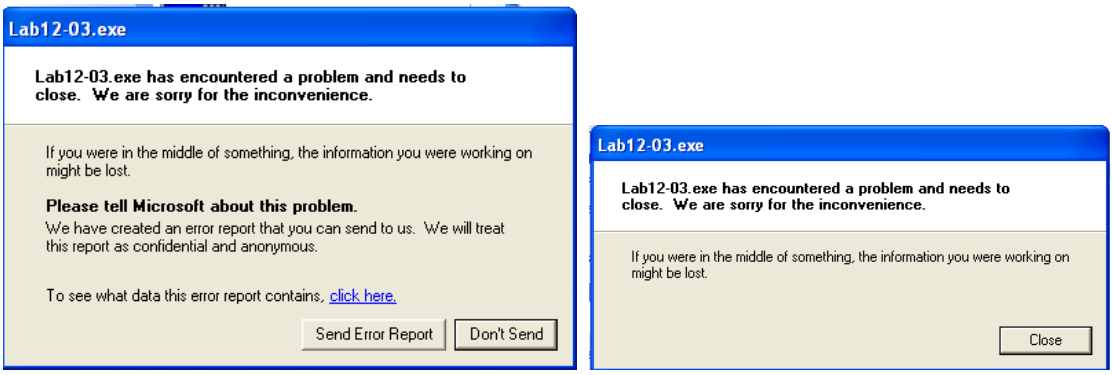
WinDbg, provides full source-level debugging for the Windows kernel, kernel-mode drivers, and system services, as well as user-mode applications and drivers (Microsoft, 2014).

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

Malware 12.3 was crashing upon running several times when using double click or run as admin right click. However that was solved when prompting to run the program from the command line.



Conclusions

The Lab identified several malware techniques to install or implement Launchers that provides a remote shell to the attacker. The tools showed how such module is being implemented and how it can be used. 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 12-1

Answers	Lab12-01. exe, Lab12-01.dll
1	We start by static analysis running Strings with the following command

	Strings Lab12-01.dll -n 5 > temp.txt We view the file created and find some interesting DLL and SYS strings; USER32.dll, KERNEL32.dll, Out of all those Kernel32.dll is of special interest since Kernel attacks gets the highest privilage available, Also we find the following names or functions that can be used; GetLastActivePopup, GetActiveWindow, MessageBoxA, Sleep, CreateThread, CloseHandle, GetModuleHandleA, GetProcAddress, LoadLibraryA, MessageBoxA, GetCommandLineA, GetVersion, ExitProcess,TerminateProcess, GetCurrentProcess, GetCurrentThreadId, SetLastError, GetLastError, SetHandleCount, GetStdHandle, GetFileType, GetStartupInfoA, DeleteCriticalSection, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, WideCharToMultiByte, GetEnvironmentStrings, GetEnvironmentStringsW, GetEnvironmentVariableA, GetVersionExA, HeapDestroy, HeapCreate, VirtualFree, HeapFree, WriteFile, SetFilePointer, EnterCriticalSection, LeaveCriticalSection, HeapAlloc, InterlockedDecrement, InterlockedIncrement, InitializeCriticalSection, GetCPInfo, GetACP, GetOEMCP, VirtualAlloc, HeapReAlloc, SetStdHandle, RtlUnwind, MultiByteToWideChar, LCMapStringA, LCMapStringW, GetStringTypeA, GetStringTypeW, FlushFileBuffers, 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 similar objects at the same time. Also, we see file handling, string handling, creating files and process, change environmental variables; terminating processes, deal with windows and popup, copy files. we also see lots of formatting strings like; H:mm:ss , dddd, MMMM dd, yyyy, M/d/yy We also find the 12 months name as well as the weekdays in full and abbreviated.

Another thing is the following two strings;

Press OK to reboot, Practical Malware Analysis %d

The first seems like a massage outputted to the user; the second is a formatted output string that will have the variable d in it.

Now running string on the Exe file we find the following file names

user32.dll, KERNEL32.dll, Lab12-01.dll, explorer.exe, psapi.dll

We can see the Lab12-01.dll that is associated with the malware probably being referenced. Explorer.exe is a main program in windows which means the attack will be manipulating it in someway. psapi.dll is the library that provides functions and information that deals with processes and drivers in the system and it stands for Process Status Application Programming Interface (Microsoft, 2014)

From all this we can say the program will be dealing with processes and threads related to the explorer, and interacting with the user in some way by showing popup messages

We also notice a Sleep string, which means the malware will run in cycles be active then sleep.

Moving to dynamic analysis, We find that the malware has created four new registry keys and modified 9 others.

11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Lab12-01.exe
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
11:11:...	Lab12-01.exe	3332	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\CWDIllegalInDLLSearch
11:11:...	Lab12-01.exe	3332	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
11:11:...	Lab12-01.exe	3332	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat
11:11:...	Lab12-01.exe	3332	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\Lab12-01.exe
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\System\CurrentControlSet\Control\Terminal Server
11:11:...	Lab12-01.exe	3332	RegQueryValue	HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCompat
11:11:...	Lab12-01.exe	3332	RegCloseKey	HKLM\System\CurrentControlSet\Control\Terminal Server
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
11:11:...	Lab12-01.exe	3332	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode
11:11:...	Lab12-01.exe	3332	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\System\CurrentControlSet\Control\SafeBoot\Option
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers
11:11:...	Lab12-01.exe	3332	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers\TransparentEnabled
11:11:...	Lab12-01.exe	3332	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKCU\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\psapi.dll
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\ntdll.dll
11:11:...	Lab12-01.exe	3332	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\kernel32.dll

Using Procmon, we find as shown in the graph the following three files have been

created which could be used as a host based indicator.

Time...	Process Name	PID	Operation	Path
11:11:...	Lab12-01.exe	3332	CreateFile	C:\WINDOWS\Prefetch\LAB12-01.EXE-1D55B943.pf
11:11:...	Lab12-01.exe	3332	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_12L
11:11:...	Lab12-01.exe	3332	CreateFile	C:\WINDOWS\system32\psapi.dll

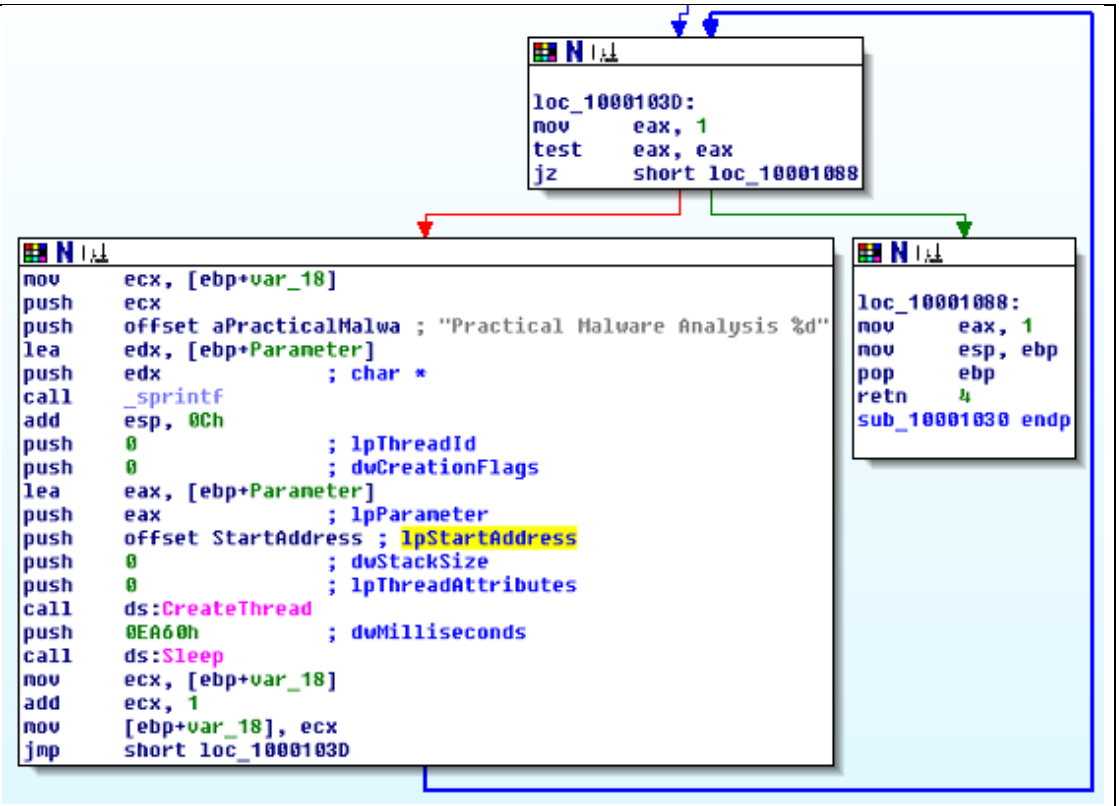
When running the program we found that both strings identified earlier have been used in a massage box that increases the count number every time the user interacts with it. Clicking Ok or X will repop the message box again.

In order to understand how the popup is being used we will examine the malware in IDA Pro.

From there we use the strings in the pop up to locate the function that provides it with the information needed and we found that in the Dll file. That passes those values to the exe file that uses explorer.exe to create the popup.

```
loc_401095:                ; size_t
push    0Ch
push    offset aExplorer_exe ; "explorer.exe"
lea     ecx, [ebp+var_108]
push    ecx                ; char *
call    __strnicmp
add     esp, 0Ch
test    eax, eax
jnz     short loc_4010B6
```

We can also see the for loop function going over and coming back to the function after every click

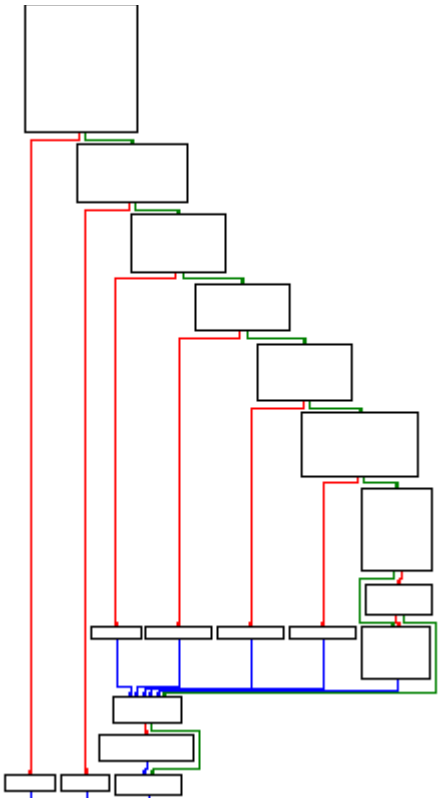
	
2	The process is injected into Explorer.exe as shown in part 1 analysis
3	By going to the WINDWOS TASK MANAGER and crashing explorer.exe and then running it again by browsing any folder and then right click and open it. that will initiate explorer.exe and everything will run normally
4	As shown in part 1 the malware injects itself into explorer.exe using its dll file then it keeps a popup alive by recreating it every time the user tries to click ok or X to close it.

Lab 12-2

Answers	Lab12-02.exe
1	We start with static analysis to find the strings in the exe file. We find the following file names and directories; KERNEL32.dll, user32.dll, ntdll.dll, \svchost.exe We also find some unique strings that can be commands or variables; UNICODE, LOCALIZATION, Sleep

	We also find the following list of functions; GetLastActivePopup, GetActiveWindow, MessageBoxA, CloseHandle, VirtualFree, ReadFile, VirtualAlloc, GetFileSize, CreateFileA, ResumeThread, SetThreadContext, WriteProcessMemory, VirtualAllocEx, GetProcAddress, GetModuleHandleA, ReadProcessMemory, GetThreadContext, CreateProcessA, FreeResource, SizeofResource, LockResource, LoadResource, FindResourceA, GetSystemDirectoryA, GetCommandLineA, GetVersion, ExitProcess, TerminateProcess, GetCurrentProcess, UnhandledExceptionFilter, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, WideCharToMultiByte, GetEnvironmentStrings, GetEnvironmentStringsW, SetHandleCount, GetStdHandle, GetFileType, GetStartupInfoA, HeapDestroy, HeapCreate, HeapFree, RtlUnwind, WriteFile, HeapAlloc, GetCPInfo, GetACP, GetOEMCP, HeapReAlloc, LoadLibraryA, MultiByteToWideChar, LCMapStringA, LCMapStringW, GetStringTypeA, GetStringTypeW, NtUnmapViewOfSection, Those functions hints that the malware will be comparing lots of strings, manipulating files, resources, process, memory and threads. It is also worth mentioning that a lot of randomly created text with A's was found however, it is also unique to say the following strings looked interesting for some reason with most of them started with we and all of them had LKla at the end. a$33.3LKA, wqsyLKla4, wqsvLKla, wqswLKla, wqstLKla143$a7, wqsuLKla, wqpxLKla4, wqpyLKla4, wqpvLKla4, wqpwLKla, wqqxLKla, wqqyLKla, wqqsLKla, Next we try IDA Pro to get some insight to the Disassembly code. from the graph bellow for the sub_40132C we can see the waterfall structure that is being created to gather information and resources needed to establish a new process or thread, the first box at the top is the main function that then calls in order FindResourceA,

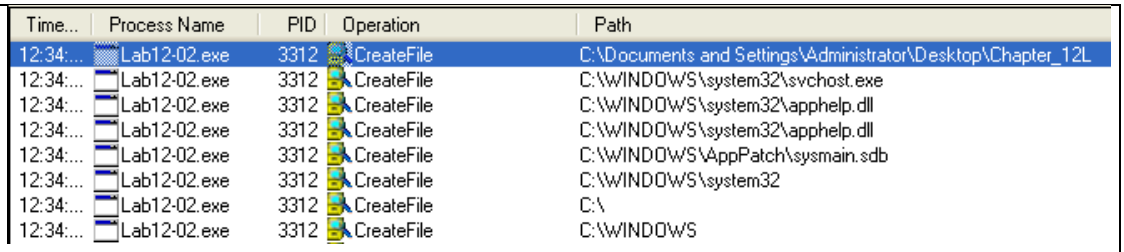
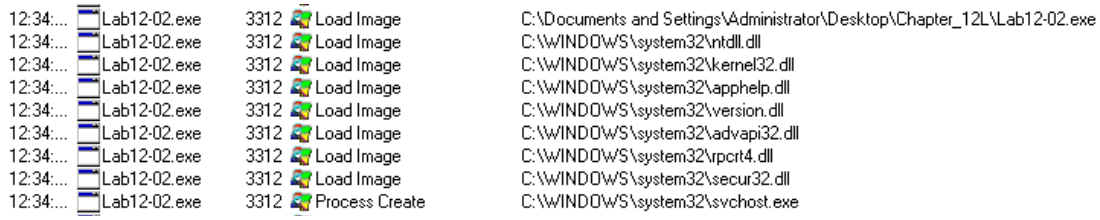
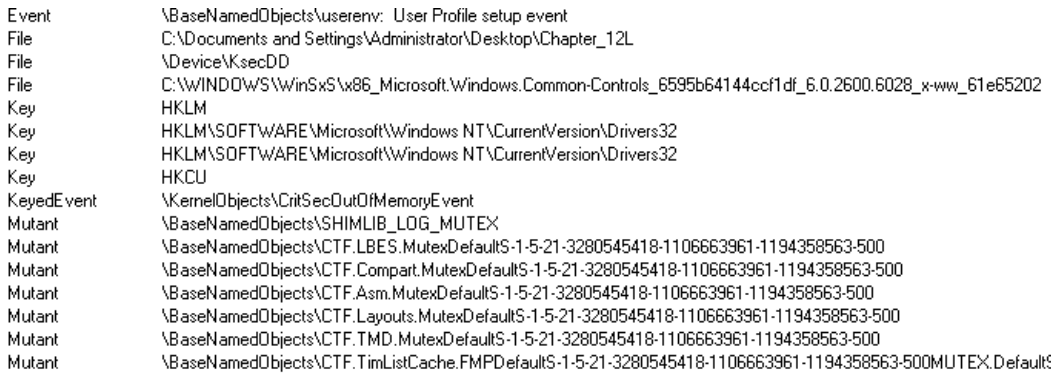
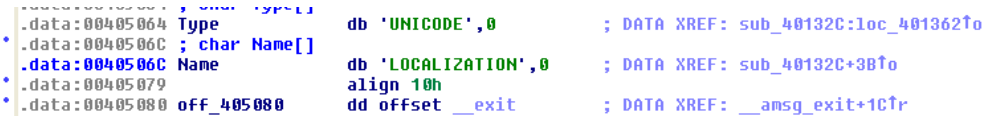
LoadResource, LockResource,VirtualAlloc, once all is passed then the resource is freed.

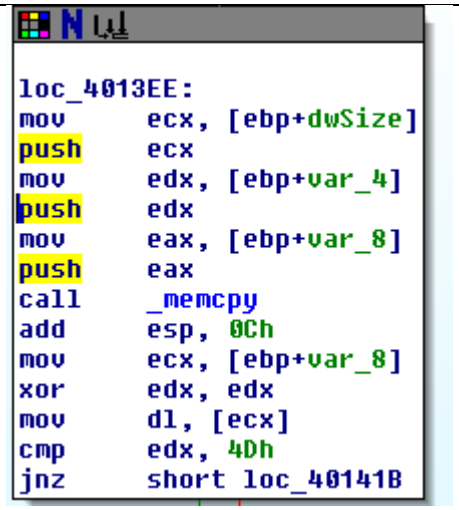


It was also found that the malware created process and pushes the ntdll.dll file which was one of the strings identified earlier as well as the process name NtUnmapViewOfSection. those observations leads me to believe that the malware is overtaking a process or loading its own into another process or using a mutex of some sort to injects its own process. As shown in the following graph.

```
call    ds:ReadProcessMemory
push    offset ProcName ; "NtUnmapViewOfSection"
push    offset ModuleName ; "ntdll.dll"
call    ds:GetModuleHandleA
push    eax             ; hModule
call    ds:GetProcAddress
```

We setup the Dynamic analysis environment to observe the malware in action for further analysis of behavior. Regshot shows us two new keys added while two other have been modified. Using procmon we find that the malware has tried creating several files

	 Time... Process Name PID Operation Path 12:34:... Lab12-02.exe 3312 CreateFile C:\Documents and Settings\Administrator\Desktop\Chapter_12L 12:34:... Lab12-02.exe 3312 CreateFile C:\WINDOWS\system32\svchost.exe 12:34:... Lab12-02.exe 3312 CreateFile C:\WINDOWS\system32\apphelp.dll 12:34:... Lab12-02.exe 3312 CreateFile C:\WINDOWS\system32\apphelp.dll 12:34:... Lab12-02.exe 3312 CreateFile C:\WINDOWS\AppPatch\sysmain.sdb 12:34:... Lab12-02.exe 3312 CreateFile C:\WINDOWS\system32 12:34:... Lab12-02.exe 3312 CreateFile C:\ 12:34:... Lab12-02.exe 3312 CreateFile C:\WINDOWS We also find lots of images being load and a new process created using svhost.exe  12:34:... Lab12-02.exe 3312 Load Image C:\Documents and Settings\Administrator\Desktop\Chapter_12L\Lab12-02.exe 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\ntdll.dll 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\kernel32.dll 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\apphelp.dll 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\version.dll 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\advapi32.dll 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\rpcrt4.dll 12:34:... Lab12-02.exe 3312 Load Image C:\WINDOWS\system32\secur32.dll 12:34:... Lab12-02.exe 3312 Process Create C:\WINDOWS\system32\svchost.exe Using Process Explorer we find the following entry for the malware creating a Mutex  Event \BaseNamedObjects\userenv: User Profile setup event File C:\Documents and Settings\Administrator\Desktop\Chapter_12L File \Device\KsecDD File C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.6028_x-ww_61e65202 Key HKLM Key HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Drivers32 Key HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Drivers32 Key HKCU KeyedEvent \KernelObjects\CritSecOutOfMemoryEvent Mutant \BaseNamedObjects\SHIMLIB_LOG_MUTEX Mutant \BaseNamedObjects\CTF.LBES.MutexDefaultS-1-5-21-3280545418-1106663961-1194358563-500 Mutant \BaseNamedObjects\CTF.Compart.MutexDefaultS-1-5-21-3280545418-1106663961-1194358563-500 Mutant \BaseNamedObjects\CTF.Asm.MutexDefaultS-1-5-21-3280545418-1106663961-1194358563-500 Mutant \BaseNamedObjects\CTF.Layouts.MutexDefaultS-1-5-21-3280545418-1106663961-1194358563-500 Mutant \BaseNamedObjects\CTF.TMD.MutexDefaultS-1-5-21-3280545418-1106663961-1194358563-500 Mutant \BaseNamedObjects\CTF.TimListCache.FMPDefaultS-1-5-21-3280545418-1106663961-1194358563-500MUTEX.DefaultS From all this we find that the program is running another program in a place that was located for something else previously.
2	By replacing an existing process with its own
3	The malicious payload is stored in the 0x040132C location  .data:00405064 Type db 'UNICODE',0 ; DATA XREF: sub_40132C:loc_401362↑o .data:0040506C ; char Name[] .data:0040506C Name db 'LOCALIZATION',0 ; DATA XREF: sub_40132C+3B↑o .data:00405079 align 10h .data:00405080 off_405080 dd offset __exit ; DATA XREF: __amsg_exit+1C↑r
4	Going back to the waterfall graph we can see that the function called after getting all the information is at 0x0040141B which has an XOR function which is being used to get the encoded information.

	
5	It would be very complicated and not in the nature of a regular malware to be using more than one method to encrypt data so the method of encryption will also be xor but in a different function. Using the XOR command we can find the other locations that have the command. Going over them we find a lot of them are jumping to the following location sub_401000 then it is being redirected to loc_401016 that includes the XOR operations for strings.

```
loc_4013EE:
mov     ecx, [ebp+dwSize]
push    ecx
mov     edx, [ebp+var_4]
push    edx
mov     eax, [ebp+var_8]
push    eax
call    _memcpy
add     esp, 0Ch
mov     ecx, [ebp+var_8]
xor     edx, edx
mov     dl, [ecx]
cmp     edx, 4Dh
jnz     short loc_40141B
```

```
loc_401016:                             ; CODE XREF: sub_401000+B↑j
mov     ecx, [ebp+var_4]
cmp     ecx, [ebp+arg_4]
jnb     short loc_401033
mov     edx, [ebp+arg_0]
add     edx, [ebp+var_4]
mov     al, [edx]
xor     al, [ebp+arg_8]
mov     ecx, [ebp+arg_0]
add     ecx, [ebp+var_4]
mov     [ecx], al
jmp     short loc_40100D
```

Lab 12-3

Answers	Lab12-03.exe
1	Static Analysis running the command Strings.exe Lab12-03.exe > temp.txt and then exploring the file created we find the following interesting three files. Kernel32.dll which provides kernel level access and functionality, and Also a log file, assuming its really a log file that means the file could be use to store data or operations created by the malware. practicalmalwareanalysis.log, KERNEL32.dll, USER32.dll.

Also, we find the following functions which provide access to file handling, pointers, hooks, windows, processes, environmental variables, string manipulation as well as local libraries.

GetModuleHandleA, AllocConsole, CloseHandle, WriteFile, SetFilePointer, CreateFileA, UnhookWindowsHookEx, GetMessageA, SetWindowsHookExA, ShowWindow, FindWindowA, CallNextHookEx, GetWindowTextA, GetForegroundWindow, GetCommandLineA, GetVersion, ExitProcess, TerminateProcess, GetCurrentProcess, UnhandledExceptionFilter, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, WideCharToMultiByte, GetEnvironmentStrings, GetEnvironmentStringsW, SetHandleCount, GetStdHandle, GetFileType, GetStartupInfoA, HeapDestroy, HeapCreate, VirtualFree, HeapFree, RtlUnwind, HeapAlloc, GetCPInfo, GetACP, GetOEMCP, VirtualAlloc, HeapReAlloc, GetProcAddress, LoadLibraryA, MultiByteToWideChar, LCMapStringA, LCMapStringW, GetStringTypeA, GetStringTypeW, ConsoleWindowClass,

However, the most interesting fact is finding the following strings, which indicates that the malware is interested in entering those commands which is less likely, or logging it which makes sense with the log file identified earlier.

[SHIFT], [ENTER], [BACKSPACE], BACKSPACE, [TAB], [CTRL], [DEL], [CAPS LOCK], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, [Window:

From this it's most likely the malware will be some kind of a key logger storing key strokes into the log file.
Next, we use IDA Pro to look into the functions within the malware to learn more about its workflow and coding;
From Main the program calls the fn function

```
call    ds:GetModuleHandleA
push    eax             ; hmod
push    offset fn       ; lpfn
```

In the fn Function we find Another Call function at Loc_4010A1

```
loc_4010A1:
mov     eax, [ebp+lParam]
mov     ecx, [eax]
push    ecx             ; Buffer
call    sub_4010C7
add     esp, 4
```

Following up the call we find the malware creates the log file and starts writing strings to it and if an interaction is made and it is not a string then it's compared to the inputs such as Enter, Tab, Space, and Backspace before writing to the log file created. As

shown in the following two graphs.

```
push    ebp
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
```

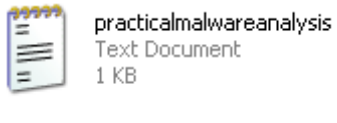
```
loc_401249:             ; lpOverlapped
push    0
lea     edx, [ebp+NumberOfBytesWritten]
push    edx             ; lpNumberOfBytesWritten
push    7               ; nNumberOfBytesToWrite
push    offset aShift   ; "[SHIFT]"
mov     eax, [ebp+hObject]
push    eax             ; hFile
call    ds:WriteFile
jmp     loc_40142C
```

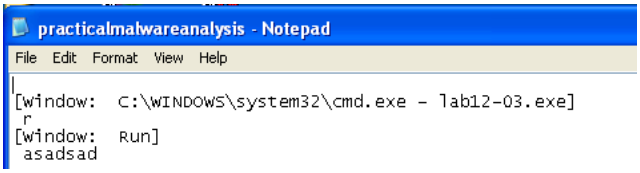
```
loc_401265:             ; lpOverlapped
push    0
lea     ecx, [ebp+NumberOfBytesWritten]
push    ecx             ; lpNumberOfBytesWritten
push    8               ; nNumberOfBytesToWrite
push    offset aEnter   ; "\n[ENTER]"
mov     edx, [ebp+hObject]
push    edx             ; hFile
call    ds:WriteFile
jmp     loc_40142C
```

The malware uses Hooks to intercept inputs before passing it back to the system. As shown in the following graph.

```
push    ecx             ; nCode
push    0               ; hhk
call    ds:CallNextHookEx ; Pass the hook information to the
                        ; next hook procedure
pop     ebp
retn    0Ch
fn endp
```

To verify the findings we will set the environment and monitor the system for results and effects. As expected we find the file working and creating the log file with the data being inputted by the user.





2	As shown in part 1 the malware uses hooks
3	As shown in part 1 the malware creates a log file named practicalmalwareanalsis.log

Lab 12-4

Answers	Lab12-04.exe
1	Starting with Static Analysis we find lots of Dll files and exe file names in the malware. KERNEL32.dll, ADVAPI32.dll, MSVCRT.dll, psapi.dll, sfc_os.dll, urlmon.dll, \winup.exe, winlogon.exe, We also find the following directories and web links. \system32\wupdmgr.exe, http://www.practicalmalwareanalysis.com/updater.exe Most interesting is the sfc_os.dll which works on the operating system File Protection. Also, urlmon.dll is used to manipulate links and embedded objects. Winlogon.exe handles interface functions that are independent of authentication policy (Microsoft, 2014). Wupdmgr.exe is another file responsible for updating the operating system files. Psapi.dll is the library that provides functions and information that deals with processes and drivers in the system and it stands for Process Status Application Programming Interface (Microsoft, 2014). From this we can see lots of security files are being targeted by the malware. The following are functions that manipulate process, threads, handles, directories, files, resources, and privileges. CloseHandle, OpenProcess, GetCurrentProcess, CreateRemoteThread, GetProcAddress, LoadLibraryA, WinExec, WriteFile, CreateFileA, SizeofResource, LoadResource, FindResourceA, GetModuleHandleA, GetWindowsDirectoryA, MoveFileA, GetTempPathA, AdjustTokenPrivileges, LookupPrivilegeValueA, OpenProcessToken, SeDebugPrivilege, EnumProcessModules, GetModuleBaseNameA, EnumProcesses, GetWindowsDirectoryA, WinExec, GetTempPathA, URLDownloadToFileA With Static Analysis done we can see that the malware is targeting system files specifically the ones that handles files, security, and objects. More analysis needs to be

done before we can know the functionality of the malware and its effects.

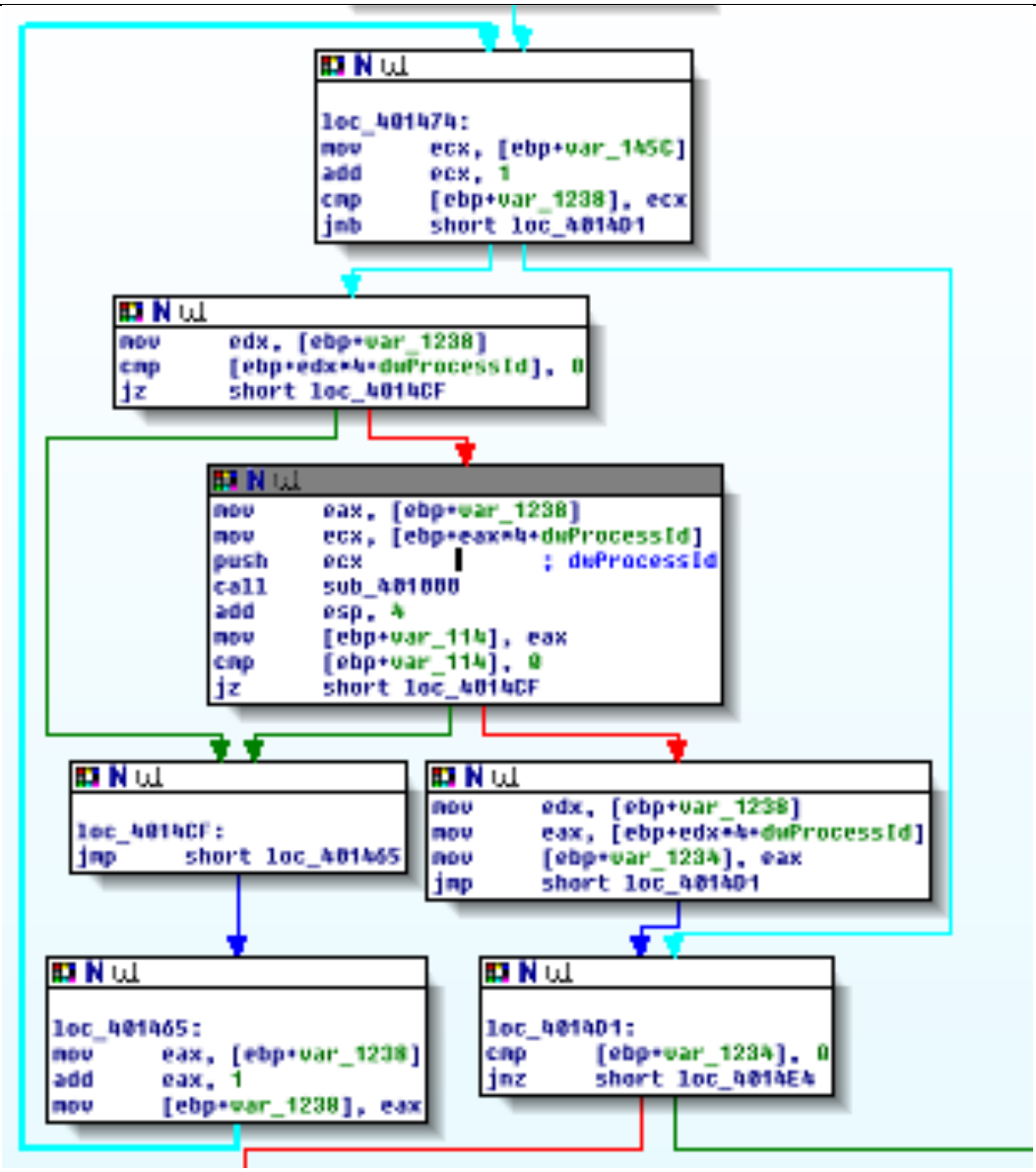
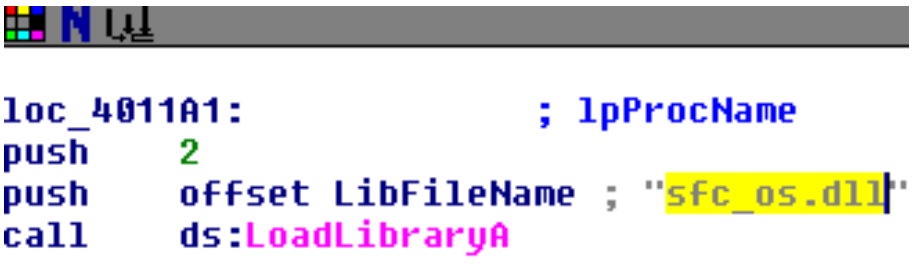
Finally we find the following irregular string <not real>. At this moment it is unknown if its of any important or irrelevant

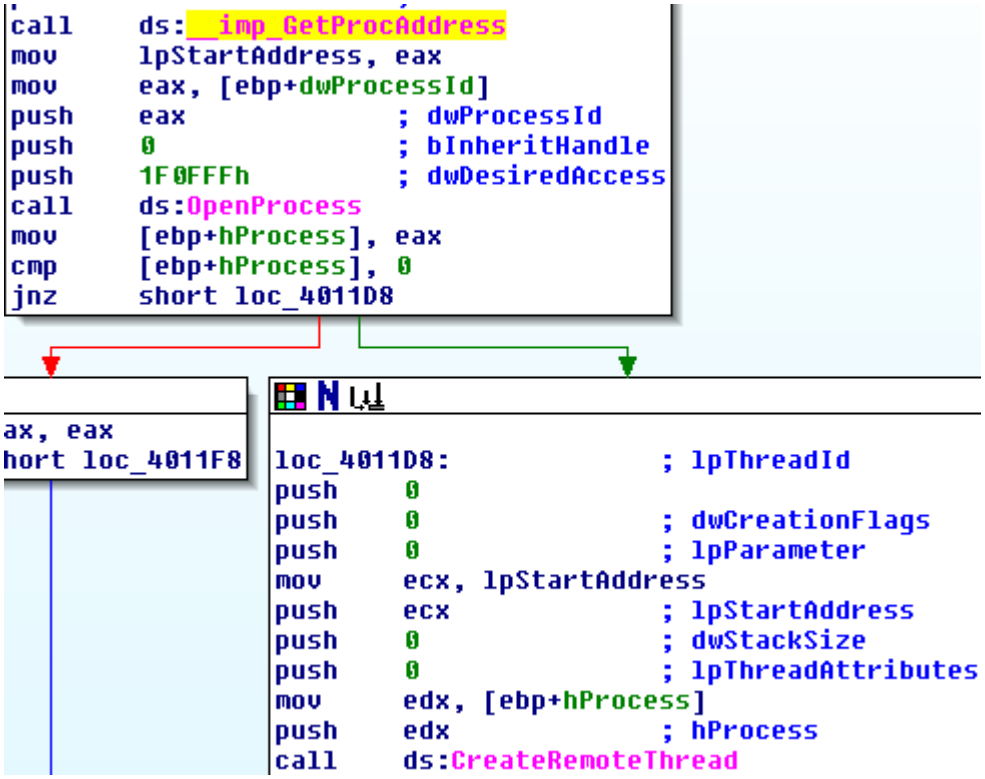
Next we try to analyze the code Using IDA Pro;

As soon as the program runs the malware tries to load the psapi.dll and then runs the windows update manager \system32\wupdmgr.exe and after that calls the \winup.exe. if not all conditions are met the malware calls the sub_401000 function in which it manipulates two strings Using OllDybg we find that the strings are winlogon.exe, <not real>. In IDA Pro the strings are being compared to the function that was called immediately before the sub_401000 routine which is

```
mov     ecx, [ebp+eax*4+dwProcessId]
push    ecx                     ; dwProcessId
call    sub_401000
```

Looking closely we also notice that there is a conditional loop in which this call is done we can see that if the value returned is zero then the loop will continue however if it was anything else it will leave the loop. This is most likely a method to identify the processID of a specific process and since we got two strings in this block of code it's more likely to be the winlogon.exe string more than <not real>. Therefore, the malware is actively looking for the process ID for the winlogon.exe.

	
2	Winlogin.exe has been the malware target as shown in part 1
3	sfc_os.dll is the Dll loaded as shown in the graph which is a library responsible for file privileges as indicated in part 1 

4	CreateRemoteThread (HANDLE hProcess, LPSECURITY_ATTRIBUTES lpThreadAttributes, DWORD dwStackSize, LPTHREAD_START_ROUTINE lpStartAddress,LPVOID lpParameter,DWORD dwCreationFlags,LPDWORD lpThreadId) This is the function header and the fourth argument is , LPTHREAD_START_ROUTINE lpStartAddress,  As shown in the graph the lpStartAddress is the value stored in ecx which was moved to it from the lpStartAddress variable which was the result of the return from the function _imp_GetProcAddress.
5	From everything shown so far we find that the malware hijacks the update manager to download its own file from the link identified by strings in the static analysis, http://www.practicalmalwareanalysis.com/updater.exe. The File downloaded will be renamed to wupdmgr.exe.

6	The malware is a program that will take over control of the update process to include its own updates which gives it system level access to add/remove/ delete files in a higher level that cannot be logged if needed. In order to do that the malware attacked the psapi.dll to gain access to the winlogon and get its own commands running utilizing privileges that are given to winlogon after disabling file protection from Microsoft windows.

References

Cogswell, B., & Russinovich, M. (2014, Sept). *Autoruns for Windows*. Retrieved from http://technet.microsoft.com/en-us/sysinternals/bb963902

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. (na.). *AppInit DLLs and Secure Boot*. Retrieved from http://msdn.microsoft.com/en-us/library/windows/desktop/dn280412(v=vs.85).aspx

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

Microsoft. (2014). *Process Status API*. Retrieved from http://msdn.microsoft.com/en-us/library/windows/desktop/ms684884(v=vs.85).aspx

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

SecurityXploded. (2013, Aug). *Show Drivers*. Retrieved from http://securityxploded.com/show-drivers.php

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