LAB 10

Malware Encoding

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 13.1 to Lab 13.3 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 10 Malware Encoding

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 13-1, 13-3

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).

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 malware techniques to encode information and hide its own operations or outputs. 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 13-1

Answers	Lab13-01. exe,
Analysis	We start by static analysis running Strings with the following command Strings Lab13-01.exe -n 5 > temp.txt We view the file created and find some interesting DLL and SYS strings; KERNEL32.dll, WS2_32.dll, WININET.dll, user32.dll Out of those it is known that Kernel provides system level access to deal with inputs, outputs, memory and operations. While, WE2_32.dll provide the access to the Socket APIs, WINNET.dll provide networking functionality, lastly user32.dll provide access to graphic user interface boxes and messages and other functions. From this it is safe to

	assume the malware is seeking kernel level privilege and actions that will utilize the network and interact with the user using massage boxes. Also we find the following names or functions that can be used; GetLastActivePopup, GetActiveWindow, MessageBoxA, LockResource, LoadResource, GlobalAlloc, SizeofResource, FindResourceA, GetModuleHandleA, InternetReadFile, InternetCloseHandle, InternetOpenUrlA, InternetOpenA, GetCommandLineA, GetVersion, ExitProcess, TerminateProcess, GetCurrentProcess, UnhandledExceptionFilter, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, WideCharToMultiByte, GetEnvironmentStrings, GetEnvironmentStringsW, SetHandleCount, GetStdHandle, GetFileType, GetStartupInfoA, HeapDestroy, HeapCreate, VirtualFree, HeapFree, RtlUnwind, WriteFile, GetLastError, SetFilePointer, HeapAlloc, GetCPInfo, GetACP, GetOEMCP, VirtualAlloc, HeapReAlloc, GetProcAddress, LoadLibraryA, SetStdHandle, FlushFileBuffers, MultiByteToWideChar, LCMapStringA, LCMapStringW, GetStringTypeA, GetStringTypeW, CloseHandle, 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 texts strings that is usually shown in a massage box like; Mozilla/4.0 http://%s/%s/ Could not load exe. Could not locate dialog box.

Could not load dialog box.,

Could not lock dialog box.

We also find the following two interesting strings;

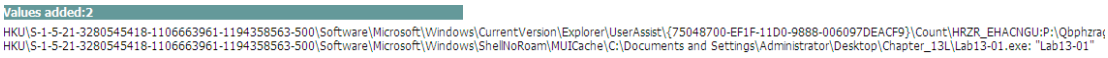
Another thing is the following two strings;

Sleep

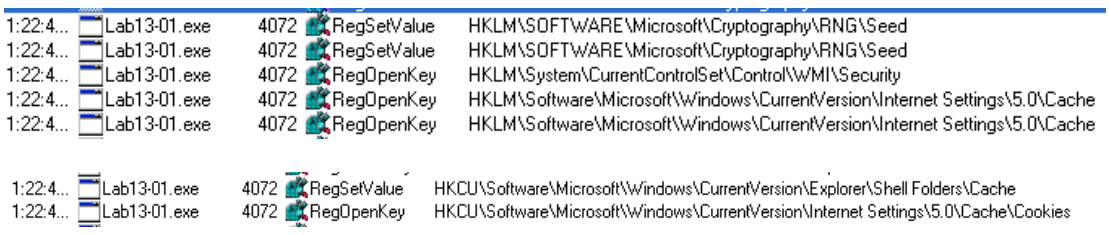
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

From this we could assume that the malware will be working and sleeping at different times or based on a specific event or command. Also, the full alphabetic string is very intriguing since it could hint to an encoding or a shuffling algorithm to hide information or data.

Moving to dynamic analysis, RegShot We find that the malware has affected 35 registry keys and created two under startup services.



Using Procmon, we filter for registry changes and analyze it to find the following interesting observations that collaborate what we see in Regshot. The malware has requested maximum level access multiple times for lots of registry locations; it also asked to set values where registry keys are present. However, going deep into the locations of those changes we find some interesting directories such as Cryptography, Cookies and Explorer Shell folder as shown in the graph bellow.



Moving to the File System Activity we also see lots of Files being created, on the network activity window we see lots of networking sessions connecting and

disconnecting as shown below it was also collaborated by the ApateDNS tool that shown a DNS Lookup to www.practicalmalwareanalysis.com.

1:23:21.7896572 PM	Lab13-01.exe	4072	TCP Reconnect	WinXP.localdomain:1787 -> 2.3.168.192.in-addr.arpa:http
1:23:21.7898600 PM	Lab13-01.exe	4072	TCP Disconnect	WinXP.localdomain:1787 -> 2.3.168.192.in-addr.arpa:http

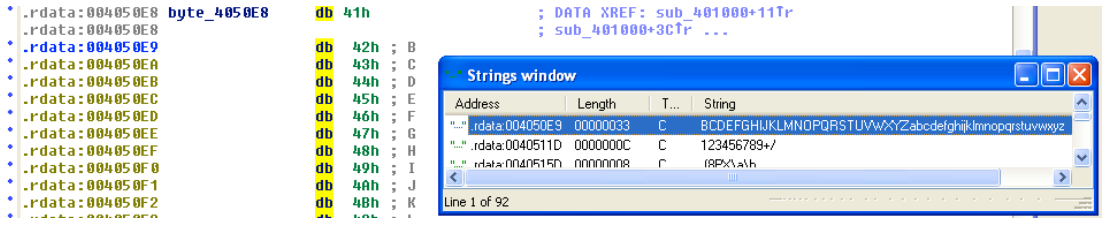
Moving to the next window we see thread and process activity in which the malware has been creating threads.

1:22:46.4735739 PM	Lab13-01.exe	4072	Process Start	SUCCESS	Parent PID: 1776, (
1:22:46.4735753 PM	Lab13-01.exe	4072	Thread Cre...	SUCCESS	Thread ID: 2228

Using Process Explorer we can see the malware creating lots of Mutants which means it will be controlling and running lots of threads and commands across different targets.

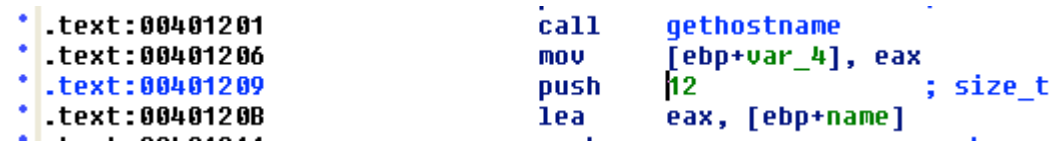
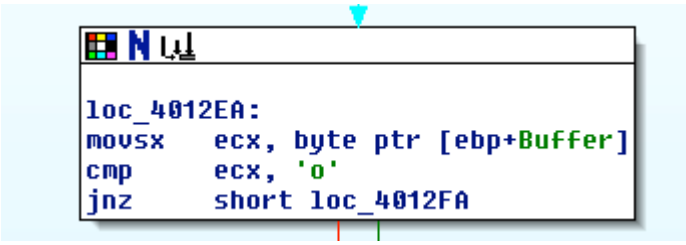
Mutant	\BaseNamedObjects_!MSFTHISTORY!_
Mutant	\BaseNamedObjects\c:!documents and settings!administrator!local settings!temporary internet files!content.ie5!
Mutant	\BaseNamedObjects\c:!documents and settings!administrator!cookies!
Mutant	\BaseNamedObjects\c:!documents and settings!administrator!local settings!history!history.ie5!
Mutant	\BaseNamedObjects\WininetStartupMutex
Mutant	\BaseNamedObjects\WininetConnectionMutex
Mutant	\BaseNamedObjects\WininetProxyRegistryMutex
Mutant	\BaseNamedObjects\RasPbFile
Mutant	\BaseNamedObjects\ZonesCounterMutex
Mutant	\BaseNamedObjects\!IETld!Mutex
Mutant	\BaseNamedObjects\ZoneAttributeCacheCounterMutex
Mutant	\BaseNamedObjects\ZonesCacheCounterMutex
Mutant	\BaseNamedObjects\ZoneAttributeCacheCounterMutex
Mutant	\BaseNamedObjects\ZonesLockedCacheCounterMutex
Process	Lab13-01.exe(4072)

Now that the static and dynamic analysis has been conducted we can go over the code and functions in the Malware DisAssembler code to learn more about it. After the malware starts running and when things are satisfying its condition it reaches the function call sub_ 401190 which takes us to an XOR function which is more likely to be used for encoding or scrambling data. That is being done using the string stored at byte_4050E8 as shown in the graph bellow



ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

	Counting those we get 64 characters which mean the encoding using the xor function is a 64 bit base encoding. 
1	Comparing both static and dynamic analysis we find that the following link was not shown in the static analysis running strings www.practicalmalwareanalysis.com link.
2	We find an XOR encoding using 64 bit string as shown in the analysis in the function at sub_ 401190
3	As shown in the analysis the byte_4050E8 provide the string that is being used for encoding which was highlighted above in red and it is used to decode the link to the practical analysis malware site
4	The string that was used as mentioned above is

	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
5	There is no evidence of multiple encoding functions therefore it should be the same base 64 encoding function used above
6	The function is at 0x004010B1 as shown in the analysis
7	From this code below we can see that the string sent to the encoding function includes the first 12 chars from the system host name. And since we have a GET command which is 3 chars and a space is needed in between total is going to be 16 for the full command line. 
8	Since the function takes 12 chars it would be safe to assume that if the hostname is less it will pad the rest to get the encoding function running. Specially since the encoding function is simple and does not account for smaller strings.
9	The Program sends a beacon to the attacker with an encoding host name and waits for a response starting with the letter o, if received the program terminates. 

Lab 13-2

Answers	Lab13-02.exe
Analysis	We start by static analysis running Strings with the following command Strings Lab13-02.exe -n 5 > temp.txt We view the file created and find some interesting DLL and SYS strings;

	GDI32.dll, user32.dll, KERNEL32.dll, Out of those it is known that Kernel provides system level access to deal with inputs, outputs, memory and operations. While, GDI32.dll is a file the provided access to the Graphics device interface module that is used for 2-dimensional objects, lastly user32.dll provide access to graphic user interface boxes and messages and other functions. From this it is safe to assume the malware is seeking kernel level privileges and actions that will use or display dimensional objects and interacts with the user using massage boxes. Also we find the following names or functions that can be used; GetLastActivePopup, GetActiveWindow, MessageBoxA, CloseHandle, WriteFile, CreateFileA, GlobalFree, GlobalUnlock, GlobalLock, GlobalAlloc, GetTickCount, ReleaseDC, GetDC, GetDesktopWindow, GetSystemMetrics, DeleteObject, DeleteDC, GetDIBits, GetObjectA, BitBlt, SelectObject, CreateCompatibleBitmap, CreateCompatibleDC, GetCommandLineA, GetVersion, ExitProcess, TerminateProcess, GetCurrentProcess, UnhandledExceptionFilter, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, WideCharToMultiByte, GetEnvironmentStrings, GetEnvironmentStringsW, SetHandleCount, GetStdHandle, GetFileType, GetStartupInfoA, HeapDestroy, HeapCreate, VirtualFree, HeapFree, RtlUnwind, GetLastError, SetFilePointer, HeapAlloc, GetCPInfo, GetACP, GetOEMCP, VirtualAlloc, HeapReAlloc, GetProcAddress, LoadLibraryA, SetStdHandle, 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 the following two interesting string;

Sleep

temp%08x

A sleep which means the malware might not be active all the time, and the second strings shows a formatting output string with a variable in the middle that can be changed.

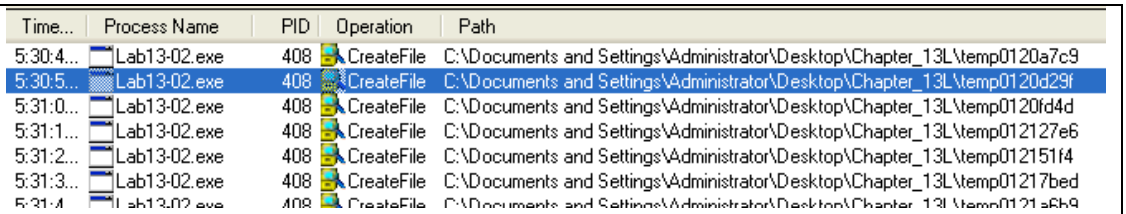
Next we moved to dynamic analysis, RegShot shows that 20 new values have been added which can be confirmed by procmon image below showing write access and maximum allowed. Specifically to the windows native WordPad application.

Keys added:20

HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\IP
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Options
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Recent File List
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\RTF
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Settings
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Text
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Word6
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Applets\Wordpad\Write
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\OpenWithList
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\BagMRU\2\0\3
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\BagMRU\2\0\3\0
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\BagMRU\2\0\4
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\Bags\38
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\Bags\38\Shell
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\Bags\39
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\Bags\39\Shell
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\Bags\40
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\Bags\40\Shell
HKU\S-1-5-21-3280545418-1106663961-1194358563-500\Software\Microsoft\Windows\ShellNoRoam\DUIBags\ShellFolders\{450D8FBA-AD25-11D0-98A8-0800361B1103}

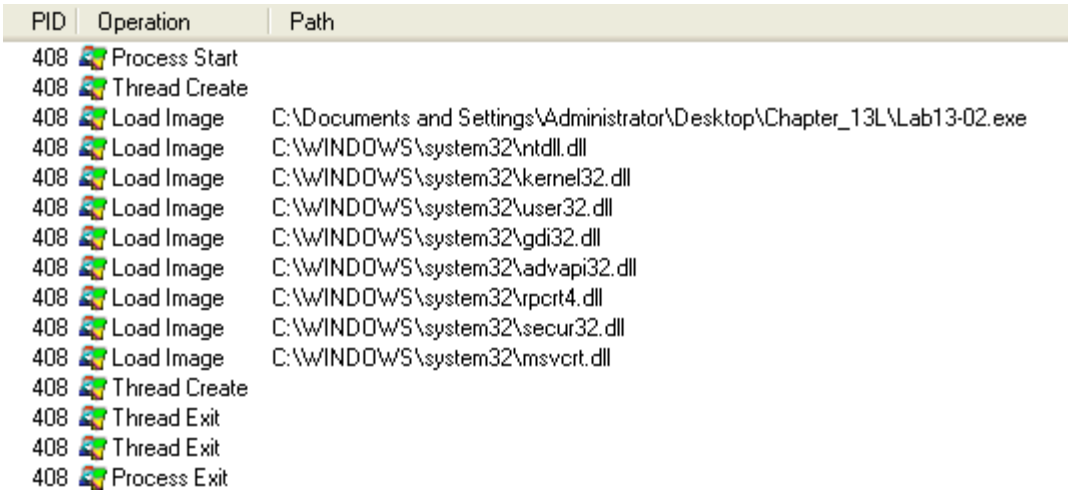
Time...	Process Name	PID	Operation	Path	Result	Detail
5:30:3...	Lab13-02.exe	408	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Query Value
5:30:3...	Lab13-02.exe	408	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers	SUCCESS	Desired Access: Query Value
5:30:3...	Lab13-02.exe	408	RegOpenKey	HKLM	SUCCESS	Desired Access: Maximum Allowed
5:30:4...	Lab13-02.exe	408	RegOpenKey	HKCU	SUCCESS	Desired Access: Read/Write
5:30:4...	Lab13-02.exe	408	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\ThemeManager	SUCCESS	Desired Access: Query Value
5:30:4...	Lab13-02.exe	408	RegOpenKey	HKCU\Control Panel\Desktop	SUCCESS	Desired Access: Query Value

Next we check the procmon file and system activity to find that the malware is creating files with the special string format identified by the static analysis temp%08 xs in the same directory of the malware file.



Time...	Process Name	PID	Operation	Path
5:30:4...	Lab13-02.exe	408	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\temp0120a7c9
5:30:5...	Lab13-02.exe	408	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\temp0120d29f
5:31:0...	Lab13-02.exe	408	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\temp0120fd4d
5:31:1...	Lab13-02.exe	408	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\temp012127e6
5:31:2...	Lab13-02.exe	408	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\temp012151f4
5:31:3...	Lab13-02.exe	408	CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\temp01217bed

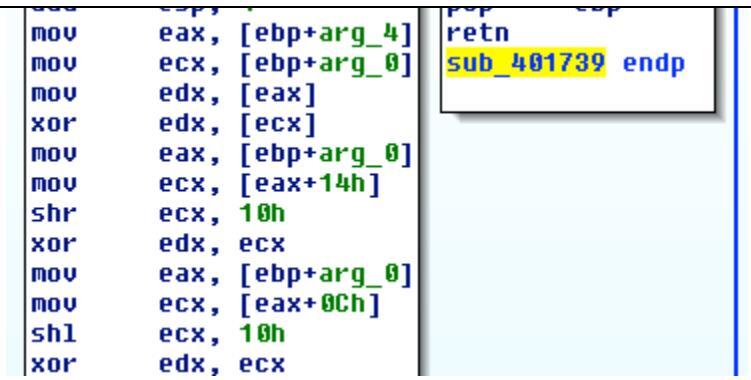
As for network activity we can confirm using both the ApateDNS and the procmon that none was established. Lastly we see thread and process activity that include loading lots of dynamic libraries while manipulating threads.

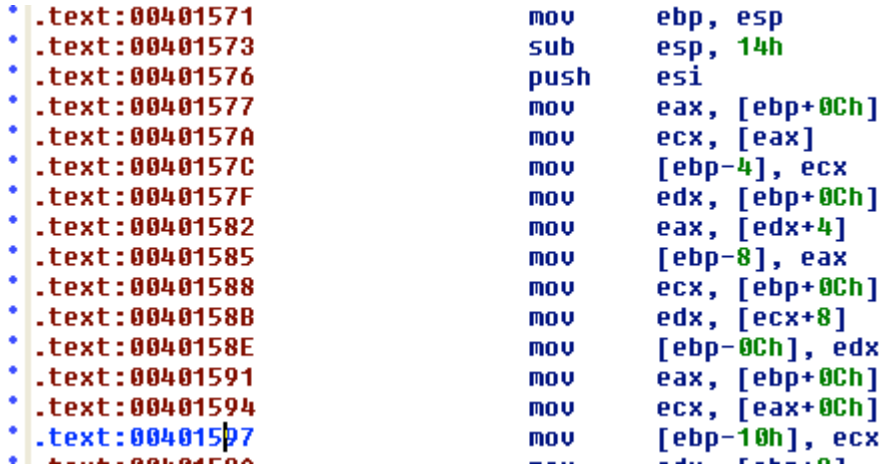


PID	Operation	Path
408	Process Start	
408	Thread Create	
408	Load Image	C:\Documents and Settings\Administrator\Desktop\Chapter_13L\Lab13-02.exe
408	Load Image	C:\WINDOWS\system32\ntdll.dll
408	Load Image	C:\WINDOWS\system32\kernel32.dll
408	Load Image	C:\WINDOWS\system32\user32.dll
408	Load Image	C:\WINDOWS\system32\gdi32.dll
408	Load Image	C:\WINDOWS\system32\advapi32.dll
408	Load Image	C:\WINDOWS\system32\rpcrt4.dll
408	Load Image	C:\WINDOWS\system32\secur32.dll
408	Load Image	C:\WINDOWS\system32\msvcrt.dll
408	Thread Create	
408	Thread Exit	
408	Thread Exit	
408	Process Exit	

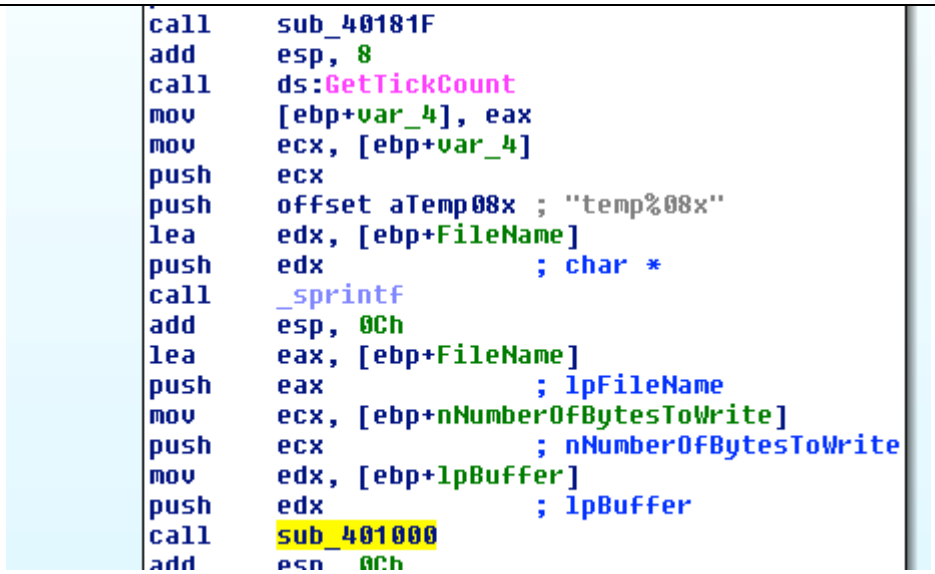
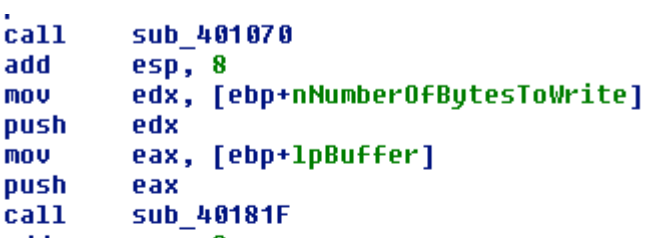
The analysis shows that the malware is loading several dynamic libraries while acquiring writing access that is used to create several files in its directory. However, those files could not be viewed when opened using WordPad or other word document programs.

This leads us to believe that an encoding function is being used at some point to hide the data. Therefore, we open IDA Pro and go over the code to get a better understanding of the malware. Going over the code we can see lots of mov and xor operations that are being used in the sub_401739 function.

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


We can also find lots of shuffling using the mov function in the following code


Starting from 0x00401570.</td></tr>
<tr><td>1</td><td>As shown in the analysis the malware creates files in its directory starting with temp and ending with a random number</td></tr>
<tr><td>2</td><td>As shown in the analysis the malware is using the xor and move function in order to hide its information.</td></tr>
<tr><td>3</td><td>The Best place to put the encoding function is before it's outputted to the file which is in sub_401000.</td></tr>
</table>

	
4	From the graph above we can see that the function for encoding is sub_401739 and its being called by a function inside the sub_40181F which is before the function that starts writing into the file.
5	We can see from the picture bellow that sub_401070 is being called before any encoding is being done which means that the output provided by that function is what is being encoded  Inside the function we see lots of commands that do with the graphical interface, such as GetSystemMetrics, GetDesktopWindow, GetDC which leads us to believe that the source is a screan capture.
6	The algorithm being used is in both functions sub_401739 and 0x00401570 locations. However it is unclear to say what scientific method or published algorithm that is.
7	If the algorithm is a two way algorithm meaning text running in gets encoded and

	encoded text running in gets you the text back then we can use the program with some modification to get the file back by swapping the inputs to be the files instead of the screen capture function. However, to verify that it's encoding the screen capture we can simply go in and NOP the encoding function which will create files that are not encoded.

Lab 13-3

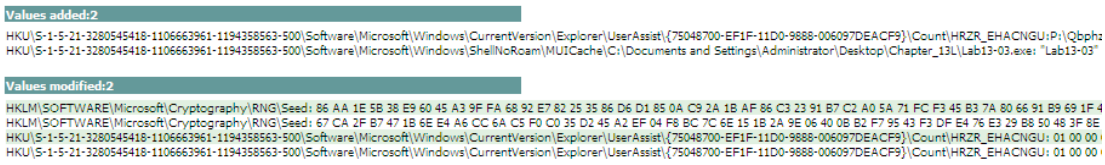
Answers	Lab13-03.exe
Analysis	We start by static analysis running Strings with the following command Strings Lab13-03.exe -n 5 > temp.txt We view the file created and find some interesting files; user32.dll, KERNEL32.dll, WS2_32.dll, cmd.exe Out of those it is known that Kernel provides system level access to deal with inputs, outputs, memory and operations. While, WS2_32.dll provides the access to the Socket APIs, user32.dll provides access to graphic user interface boxes and messages and other functions. CMD.exe give access to command line which means it could be used to create a reverse shell. From this it is safe to assume the malware is seeking kernel level privilege and actions that will utilize the network to send commands to a command shell and interact with the user using massage boxes. Also we find the following names or functions that can be used; GetLastActivePopup, GetActiveWindow, MessageBoxA, ExitProcess, LocalFree, WriteConsoleA, GetStdHandle, lstrlenA, FormatMessageA, GetLastError, WriteFile, ReadFile, WaitForSingleObject, CreateThread, CreateProcessA, CloseHandle, DuplicateHandle, GetCurrentProcess, CreatePipe, wsprintfA, WSASocketA, MultiByteToWideChar, RaiseException, GetCommandLineA, GetVersion, RtlUnwind, HeapFree, TerminateProcess,

	HeapReAlloc, HeapAlloc, HeapSize, SetUnhandledExceptionFilter, UnhandledExceptionFilter, GetModuleFileNameA, FreeEnvironmentStringsA, FreeEnvironmentStringsW, WideCharToMultiByte, GetEnvironmentStrings, GetEnvironmentStringsW, SetHandleCount, GetFileType, GetStartupInfoA, HeapDestroy, HeapCreate, VirtualFree, SetFilePointer, FlushFileBuffers, VirtualAlloc, IsBadWritePtr, IsBadReadPtr, IsBadCodePtr, GetCPInfo, GetACP, GetOEMCP, GetProcAddress, LoadLibraryA, SetStdHandle, LCMapStringA, LCMapStringW, GetStringTypeA, GetStringTypeW, ReadFile, WriteConsole, ReadConsole, WriteFile, DuplicateHandle, GetStdHandle, CreateThread, 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, and creating files, process, pipes, and handles, change environmental variables; terminating processes, deal with windows and popup, copy files. we also see lots of texts strings that is usually used as informative massages that can be used to inform the attacker or interact with the user; Object not Initialized Data not multiple of Block Size Empty key Incorrect key length Incorrect block length ERROR: API = %s. Error code = %d. Message = %s. www.practicalmalwareanalysis.com

	also, we find the following two strings that can be used for encoding CDEFGHIJKLMNOPQRSTUVWXYZABcdefghijklmnopqrstuvwxyzab0123456789+/ Ijklmnopqrstuvwx That is confirmed by the following strings that looks obfuscated and looks more or less like email address or other text that is encoded. .?AVexception@@ .?AVios_base@std@@ .?AV?$basic_ios@DU?$char_traits@D@std@@@std@@ .?AV?$basic_istream@DU?$char_traits@D@std@@@std@@ .?AV?$basic_ostream@DU?$char_traits@D@std@@@std@@ .?AV?$basic_streambuf@DU?$char_traits@D@std@@@std@@ .?AV?$basic_filebuf@DU?$char_traits@D@std@@@std@@ .?AV?$basic_ios@GU?$char_traits@G@std@@@std@@ .?AV?$basic_istream@GU?$char_traits@G@std@@@std@@ .?AV?$basic_ostream@GU?$char_traits@G@std@@@std@@ .?AV?$basic_filebuf@GU?$char_traits@G@std@@@std@@ .?AV?$basic_streambuf@GU?$char_traits@G@std@@@std@@ .?AVruntime_error@std@@ .?AVfailure@ios_base@std@@ .?AVfacet@locale@std@@ .?AV_Locimp@locale@std@@ .?AVlogic_error@std@@ .?AVlength_error@std@@ .?AVout_of_range@std@@ .?AVtype_info@@

From the static analysis so far we could assume that the malware will be using some encoding function, communicating with the attacker, receiving commands, and transferring information, using the website www.practicalmalwareanalysis.com utilizing the dynamic libraries mentioned. Working and sleeping at different times or based on a specific event or command.

Moving to dynamic analysis, RegShot We find that the malware has affected 2 registry keys and created two under startup services. Looking closely, we see the Microsoft Cryptography in the path which confirms the use of encoding, as well as adding something to the explorer directory which could lead to persistency

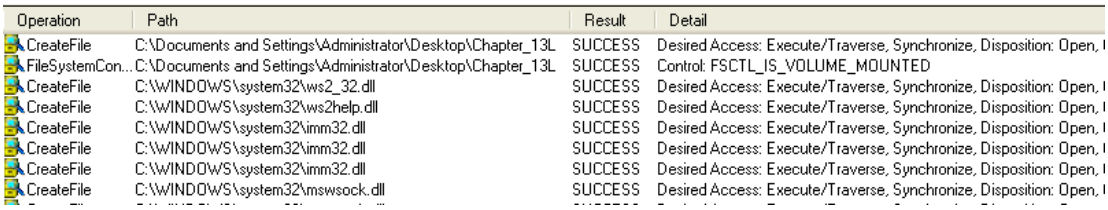


Using Procmon, we filter for registry changes and analyze it to find the following interesting observations that collaborate what we see in Regshot. The malware has requested maximum level access multiple times for lots of registry locations; it also asked to set values where registry keys are present. However, going deep into the locations of those changes we find some interesting directories such as Cryptography, WinSock2 as shown in the graph bellow, which confirms RegShot findings.

Operation	Path	Result	Detail
RegOpenKey	HKLM	SUCCESS	Desired Access: Maximum Allowed
RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\Parameters	SUCCESS	Desired Access: Maximum Allowed
RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_Catalog9	SUCCESS	Desired Access: Maximum Allowed
RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_Catalog9\Catalog_Entries	SUCCESS	Desired Access: Maximum Allowed
RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5	SUCCESS	Desired Access: Maximum Allowed
RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_Catalog5\Catalog_Entries	SUCCESS	Desired Access: Maximum Allowed
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: A2 2A 90
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: C2 67 EB
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: 6B 0E EE
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: CA 36 20
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: D7 55 17
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: 53 47 59
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: AE 90 B3
RegSetValue	HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed	SUCCESS	Type: REG_BINARY, Length: 80, Data: 67 CA 2F
RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\Parameters	SUCCESS	Desired Access: Maximum Allowed

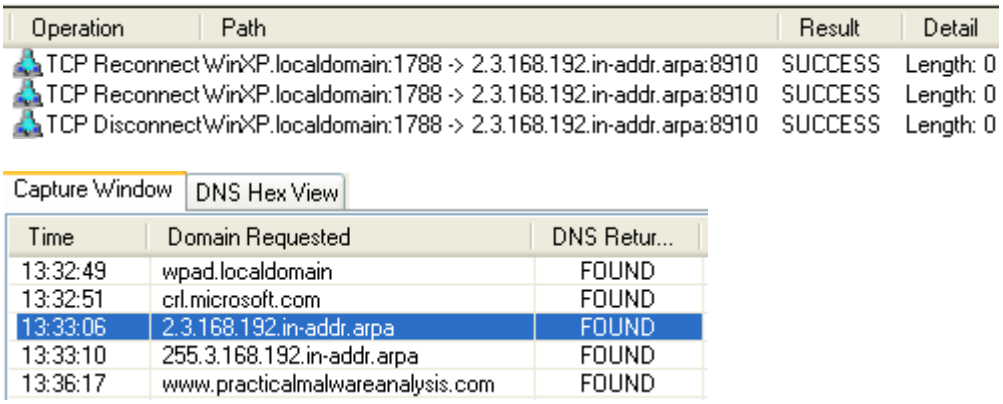
Moving to the File System Activity we also see lots of Files being created that already exists in windows System32 directory, interestingly we also see a unique Volume

Mounted operation to the file system.



Operation	Path	Result	Detail
CreateFile	C:\Documents and Settings\Administrator\Desktop\Chapter_13L	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I
FileSystemCon...	C:\Documents and Settings\Administrator\Desktop\Chapter_13L	SUCCESS	Control: FSCTL_IS_VOLUME_MOUNTED
CreateFile	C:\WINDOWS\system32\ws2_32.dll	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I
CreateFile	C:\WINDOWS\system32\ws2help.dll	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I
CreateFile	C:\WINDOWS\system32\imm32.dll	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I
CreateFile	C:\WINDOWS\system32\imm32.dll	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I
CreateFile	C:\WINDOWS\system32\imm32.dll	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I
CreateFile	C:\WINDOWS\system32\mswsock.dll	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposition: Open, I

On the network activity window we see couple of networking sessions connecting as shown below it was also collaborated by the ApateDNS tool that shown a DNS Lookup to www.practicalmalwareanalysis.com.

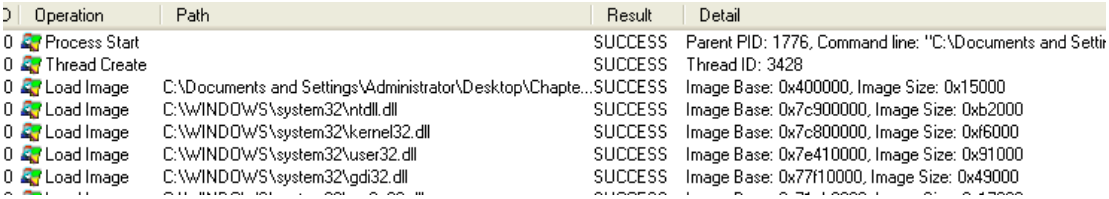


Operation	Path	Result	Detail
TCP Reconnect	WinXP.localdomain:1788 -> 2.3.168.192.in-addr.arpa:8910	SUCCESS	Length: 0
TCP Reconnect	WinXP.localdomain:1788 -> 2.3.168.192.in-addr.arpa:8910	SUCCESS	Length: 0
TCP Disconnect	WinXP.localdomain:1788 -> 2.3.168.192.in-addr.arpa:8910	SUCCESS	Length: 0

Capture Window | DNS Hex View

Time	Domain Requested	DNS Retur...
13:32:49	wpad.localdomain	FOUND
13:32:51	crl.microsoft.com	FOUND
13:33:06	2.3.168.192.in-addr.arpa	FOUND
13:33:10	255.3.168.192.in-addr.arpa	FOUND
13:36:17	www.practicalmalwareanalysis.com	FOUND

Moving to the next window we see thread and process activity in which the malware has been creating and exiting threads while loading images of dynamic links into memory.



	Operation	Path	Result	Detail
0	Process Start		SUCCESS	Parent PID: 1776, Command line: "C:\Documents and Settir
0	Thread Create		SUCCESS	Thread ID: 3428
0	Load Image	C:\Documents and Settings\Administrator\Desktop\Chapte...	SUCCESS	Image Base: 0x400000, Image Size: 0x15000
0	Load Image	C:\WINDOWS\system32\ntdll.dll	SUCCESS	Image Base: 0x7c900000, Image Size: 0xb2000
0	Load Image	C:\WINDOWS\system32\kernel32.dll	SUCCESS	Image Base: 0x7c800000, Image Size: 0xf6000
0	Load Image	C:\WINDOWS\system32\user32.dll	SUCCESS	Image Base: 0x7e410000, Image Size: 0x91000
0	Load Image	C:\WINDOWS\system32\gdi32.dll	SUCCESS	Image Base: 0x77f10000, Image Size: 0x49000

Now that the static and dynamic analysis has been conducted we can go over the code and functions in the Malware DisAssembler code to learn more about it. Using IDA Pro, we can see that the addresses we suspected earlier could be the input that creates the function names in the malware since most functions are basically encoded in such a way that it's unclear what it is used for.

1	From Going over the text we find more that 50 functions with names that were not shown in the static analysis.
2	We find the following function with lots of XOR and MOV functions sub_40132B, sub_401AC2, sub_40223A, sub_4027ED, sub_402DA8, sub_403166. Due to the nature of its look it's not clear what algorithm it's using or if it's known or published method of encoding.
3-4-5	IDA Pro identified the encryption method to be AES and a 64base encryption which also shows that both strings that has been suspected in the static analysis The first one is a 64 bit string and the 2nd one is what is being used for the AES algorithm. CDEFGHIJKLMNOPQRSTUVWXYZABcdefghijklmnopqrstuvwxyzab0123456789+/ Ijklmnopqrstuvwx
6	For the 64bit encryption the code is enough to decode the information however for the AES lots of more variables needs to be identified in order to get the decoded information.
7	The malware as shown in analysis is communicating with the attacker using the command shell and both algorithms are used to encode sending and received data.
8	Using the OllDbg the encoding functions for massages being sent to the attacker can be replaced with NOP operations which means information won't be encoded and therefore can be viewed using wireshark or other applications that will network massages. On the other hand, all massages being sent from the attacker are most probably commands and hence needs to be decrypted before execution in which at that point in time it will be available in memory as clear text.

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/