LAB 4

Lab 4 DisAssembly Code Analysis

Under the direction of
Dr. Samuel Liles

Table of Contents

Abstract

This lab is focused on DisAssembly Code 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 6.1 to Lab 6.4 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 4 DisAssembly Code Analysis

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 6-1, 6-4

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).

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).

Issues or problems

Resource Hacker was crashing when trying to upload 6.1 file.

Conclusions

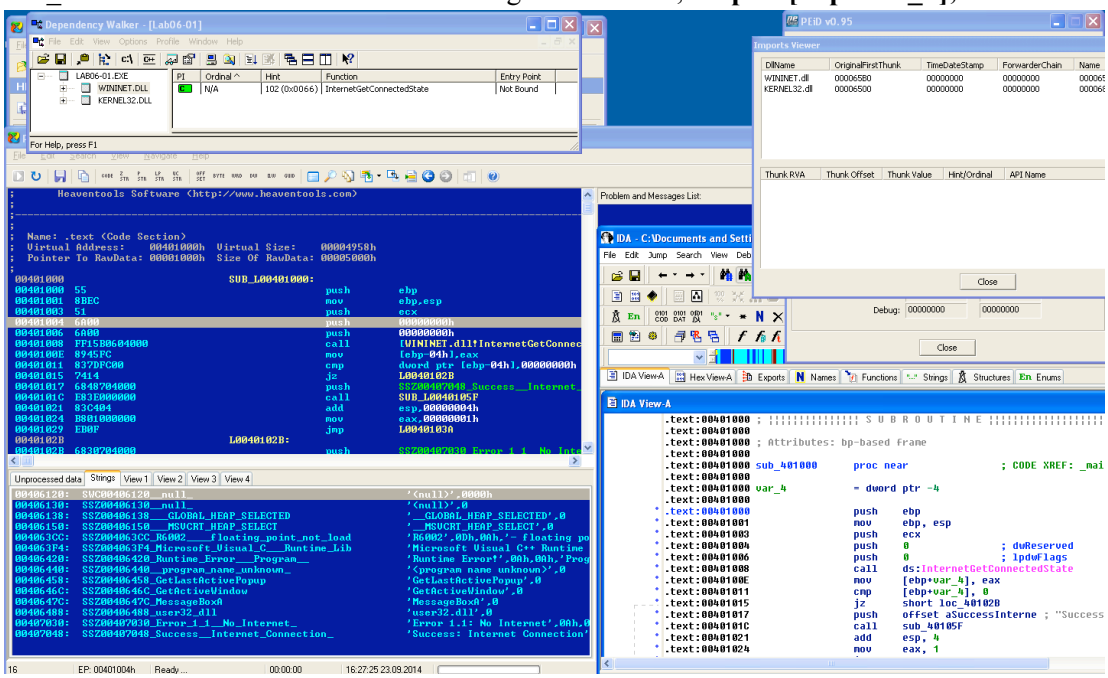
The Lab identified several programs that helps explore the malwares. The tools showed if the files being used are infected or packed. The tools used also showed the resources on the system that is being utilized such as privilege, CPU usage, Network communication.

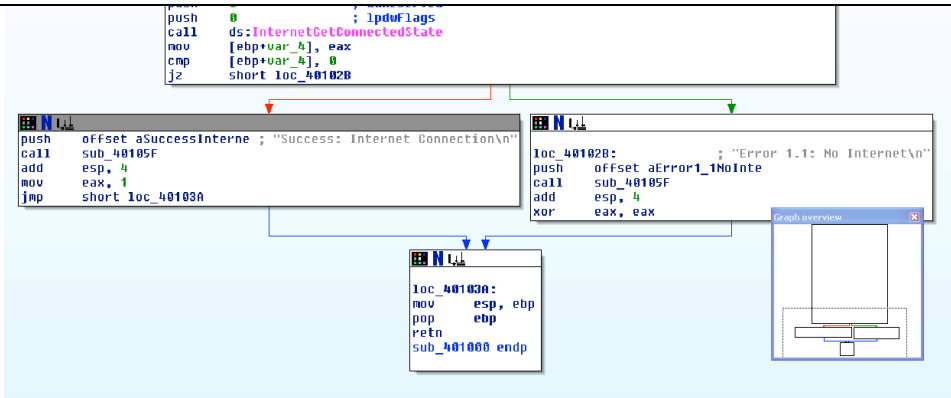
Case studies

No Case studies was given with this lab.

Review questions

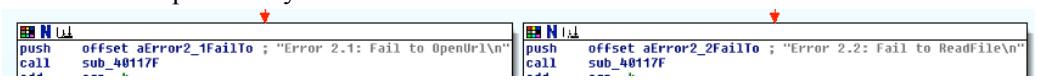
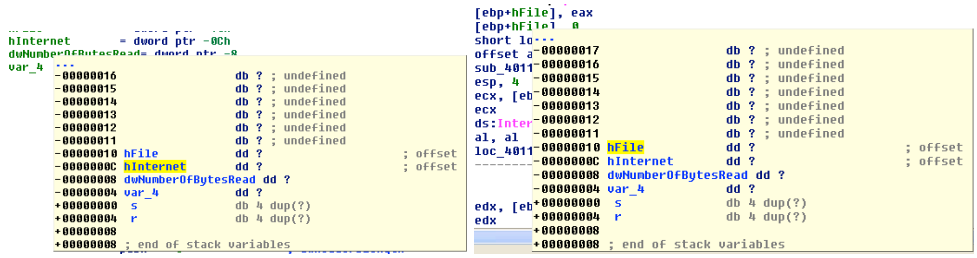
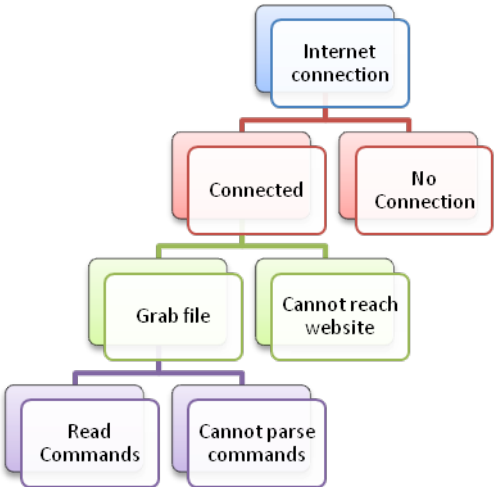
Lab 6-1

Answers	Lab06-01. exe
1	The file is packed with two DLL files used KERNEL32.DLL & WININET.dll with the following function InternetGetConnectedState. Based on the static analysis the file seems to be checking the internet connectivity for further usage. The important function in the main segment of code is a compare function which is a at .text:00401000 sub_401000 which includes the following if statement; **cmp [ebp+var_4], 0** 
2	From the Graph provided by IDA we can notice that the function at 0x40105F is being called twice in the program after a comparison. From the text on both sides of the branch we can assume that the comparison is checking for internet connectivity based on the text shown "Success ...." & "Error ...." and since the function is being called immediately after those two text strings that include text command output such as \n it is safe to assume that the text pushed to the stack is to be printed by the function Located at 0x40105F.

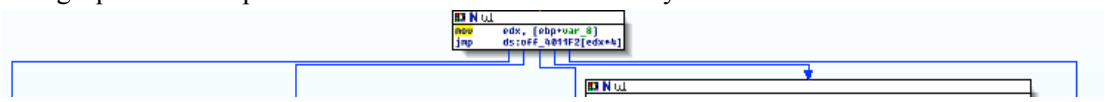
	
3	**strings -n 14 Lab06-01.exe > strings.txt** Using the string program with the above command creates a text file in which the last three lines include the following; **Error 1.1: No Internet** **Success: Internet Connection** All readable text in the file could be associated with network connectivity and since the main function shown by IDA is doing a comparison followed with those two sentences the program is a simple code to check for internet connectivity.

Lab 6-2

Answers	Lab06-02. exe
1	The file is packed with two DLL files used KERNEL32.DLL & WININET.dll with the following functions InternetCloseHandle, InternetGetConnectedState, InternetOpenA, InternetOpenUrlA, InternetReadFile. Based on the static analysis the file seems to be checking the internet connectivity for further usage at .text:00401008 **call ds:InternetGetConnectedState**. The result returned is used in a comparison which is the most important function in the main segment of code at .text:00401011 that says **cmp [ebp+var_4], 0** .text:00401000 .text:00401000 sub_401000 proc near ; CODE XREF: _main+6↓p .text:00401000 .text:00401000 var_4 = dword ptr -4 .text:00401000 .text:00401000 push ebp .text:00401001 mov ebp, esp .text:00401003 push ecx .text:00401004 push 0 ; dwReserved .text:00401006 push 0 ; lpdwFlags .text:00401008 call ds:InternetGetConnectedState .text:0040100E mov [ebp+var_4], eax .text:00401011 cmp [ebp+var_4], 0 .text:00401015 jz short loc_40102B .text:00401017 push offset aSuccessInterne ; "Success: Internet Connection\n" .text:0040101C call sub_40117F .text:00401021 add esp, 4 .text:00401024 mov eax, 1 .text:00401029 jmp short loc_40103A
2	From the Graph provided by IDA we can notice that the function at 0x40117F is being called every time there is a text string and command pushed to the stack "Success ...." & "Error ...." and since the function is being called immediately after those text strings that include text command output such as \n it is safe to assume that the text pushed to the

	stack is to be printed by the function Located at 0x40117F. 
3	The 2nd subroutine is at sub_401040 and it tries to open a connection to the internet using internet browser to the following link **http://www.practicalmalwareanalysis.com/cc.htm** If access was successful it will read the content if not an error message will be displayed as shown in the above image. If reading was successful the program will continue if not another error message will be displayed.
4	It seems like an array of variables is being created that is then used to store information retrieved from the website. As shown in the following pictures. 
5	Using the ApateDNS we see a request has been sent to the following website **http://www.practicalmalwareanalysis.com/cc.htm** And using Process Monitor we can find the following TCP connection being requested. 
6	The Malware acts as following diagram, if internet is available it will try to read commands from the website and parse it. If errors happen each one will give a different error based on the current stage in the  malware.

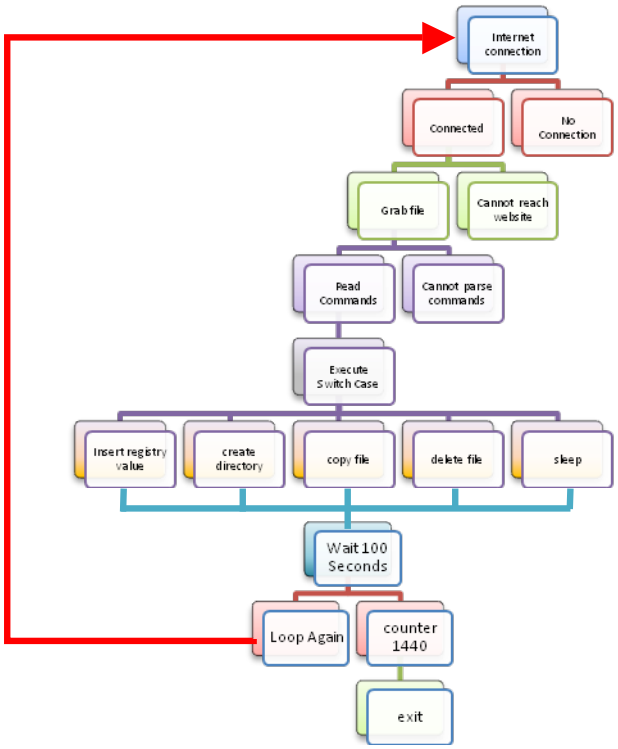
Lab 6-3

Answers	Lab06-03.exe
1	Everything is the same in the main method except for the fact that a new function is being added which is creating a directory using the subroutine sub_401130
2	sub_401130(char,LPCSTR lpExistingFileName) it has two parameters first is Char which is parsed from the commands retrieved from the website 2nd is lpExistingFileName which is the local file name Lab06-02
3	The graph shows 5 paths out of the function which says that it has a switch in it  Which is also reflected in the following code showing 5 locations to jump to.  off_4011F2 dd offset loc_40115A dd offset loc_40116C dd offset loc_40117F dd offset loc_4011BC dd offset loc_4011D4 align 10h
4	The function can do the following, print error message which is the default case, create directory, copy/delete file, sleep, it also sets a value in the registry as shown in the following graph  push edx ; hKey call ds:RegSetValueExA
5	Based on the case file we could find two or at least one value, the first that will always be available is the registry key created by the function, the 2nd is the created directory C:\\temp\cc.exe.
6	Similarly to 6-2 the malware will check for connection and if found it will grab the file and parse the command for execution and based on the command parsed one of the five cases in the switch will be executed. 

Lab 6-4

Answers	Lab06-04.exe
1	It is the same methods and functions in 6-3.

2	The following for loop has been added which mean we can be using more than one case from the switch case function .text:00401251 mov eax, [ebp+var_C] .text:00401254 add eax, 1 .text:00401257 mov [ebp+var_C], eax .text:0040125A .text:0040125A loc_40125A: ; CODE XREF: _main+1F↑j .text:0040125A cmp [ebp+var_C], 5A0h .text:00401261 jge short loc_4012AF .text:00401263 mov ecx, [ebp+var_C] .text:00401266 push ecx .text:00401267 call sub_401040 .text:0040126C add esp, 4 .text:0040126F mov [ebp+var_8], al .text:00401272 movsx edx, [ebp+var_8] .text:00401276 test edx, edx .text:00401278 jnz short loc_40127E .text:0040127A xor eax, eax .text:0040127C jmp short loc_4012B1 .text:0040127E ; --- .text:0040127E .text:0040127E loc_40127E: ; CODE XREF: _main+48↑j .text:0040127E movsx eax, [ebp+var_8] .text:00401282 push eax .text:00401283 push offset aSuccessParsedC ; "Success: Parsed command is %c\n" .text:00401288 call sub_4012B5 .text:0040128D add esp, 8 .text:00401290 mov ecx, [ebp+argv] .text:00401293 mov edx, [ecx] .text:00401295 push edx ; lpExistingFileName .text:00401296 mov al, [ebp+var_8] .text:00401299 push eax ; char .text:0040129A call sub_401150 .text:0040129F add esp, 8 .text:004012A2 push 0EA60h ; dwMilliseconds .text:004012A7 call ds:Sleep .text:004012AD jmp short loc_401251
3	In the previous labs the following text has been used Internet Explorer 7.5/pma Now the text has became Internet Explorer 7.50/pma%d. now that shows us that %d is a variable and by going through the code that variable has the same value as the for loop counter which means it will open a new instance every time it runs which is once every minute. Moreover, sprintf function now takes the following parameters when creating the user-agent connection (char *,const char *,...).
4	The compare code for Var_C which is the counter for the for loop shown above, is compared to FA0H which is as shown in the graph 1440 min, 1440/60 = 24 hours .text:0040125A cmp [ebp+var_C], 1440
5	Every time the malware communicates it instantiates a new instance that includes the counter Var_C as explained previously. So we will have one network indicator for every minute the program has been running or more accurately every time the for loop is executed.

6	Similarly to 6-3 the malware will check for connection and if found it will grab the file and parse the command for execution and based on the command parsed one of the five cases in the switch will be executed. This is one loop out of 1440 loops in case communication is active once disconnected the program will terminate. This counter indicates how long the malware has been online without any disconnection. 

References

Davis, S. (2011, October). *ApateDNS*. Retrieved from https://www.mandiant.com/blog/research-tool-release-apatedns/

Gröbert, F. (2010, 02 07). *PEiD*. Retrieved 02 18, 2014, from https://code.google.com/p/kerckhoffs/downloads/

Heaventools Software. (2009, 10 14). *Heaventools*. Retrieved from http://heaventools.com/download.htm

Hex-Rays SA. (2014, July). *Freeware Download Page*. Retrieved from https://www.hex-rays.com/index.shtml

Johnson, A. (2011, 09 16). *Resource Hacker*. Retrieved from http://www.angusj.com/resourcehacker/

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