CNIT 58100 CFM: CYBERFORENSICS OF MALWARE – LAB 9

Ibrahim Waziri Jr

PhD in Information Security (CERIAS)

Lab 9

Instructor: **Associate Prof Sam Liles**

Purdue University

2014

CNIT 581: CyberForensics of Malware – Lab 9

Abstract

This lab covers skills discussed in chapter 9 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.

The lab focuses on OllyDbg which provides the ability to analyze malware while it is running.

Lab 9-1

This malware is analyzed using OllyDbg and IDAPro and uses the files Lab09-01.exe

1. The file Lab09-01.exe automatically disappears whenever I double-click it, therefore before running the file, we analyze it first using IDA Pro and OllyDbg.
   As shown in Fig 1 below: From IDA Pro analysis we can tell that the main function starts at 0x402AF0.

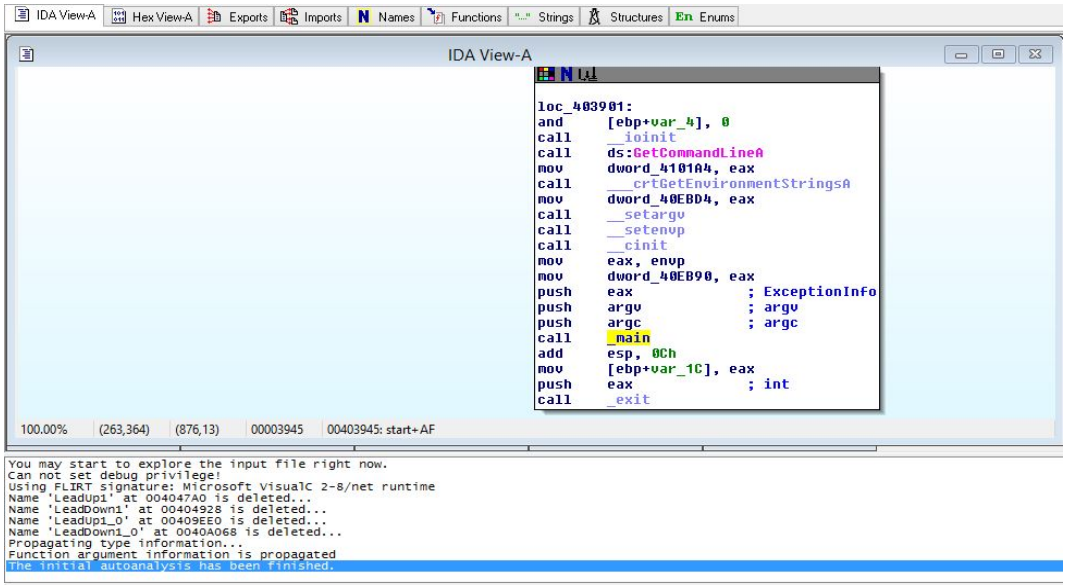


Fig 1: Analysis using IDA Pro

Next we use OllyDbg to analyze the malware. We use the F8 key to step-over until we arrive the address 0x403945 which is the call to the main function. As shown below: we used OllyDbg to locate the main call.

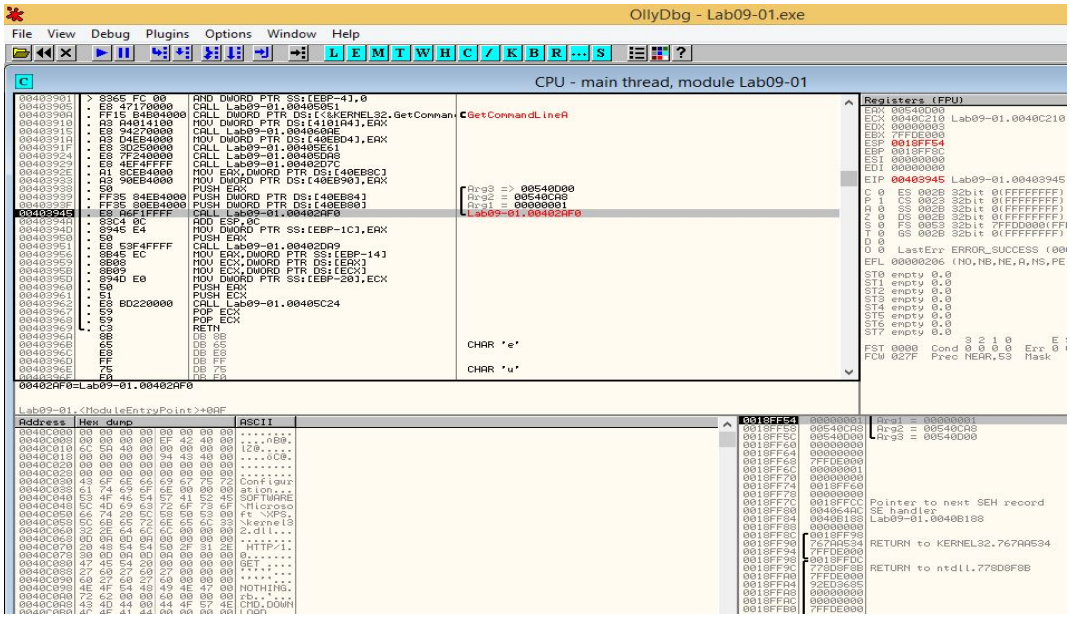


Fig 2: OllyDbg locates the main call at 0x403925.

From this analysis, we can tell that the malware checks to see if the number of command-line arguments equal 1 at address 0x402AFD.

The malware prepares to delete itself from the disk by combining the constructed string with program.

Therefore to install the program, we can get it to install itself by providing it with the –in argument in the command line as shown below: To do that, go to **Debug – Arguments**.

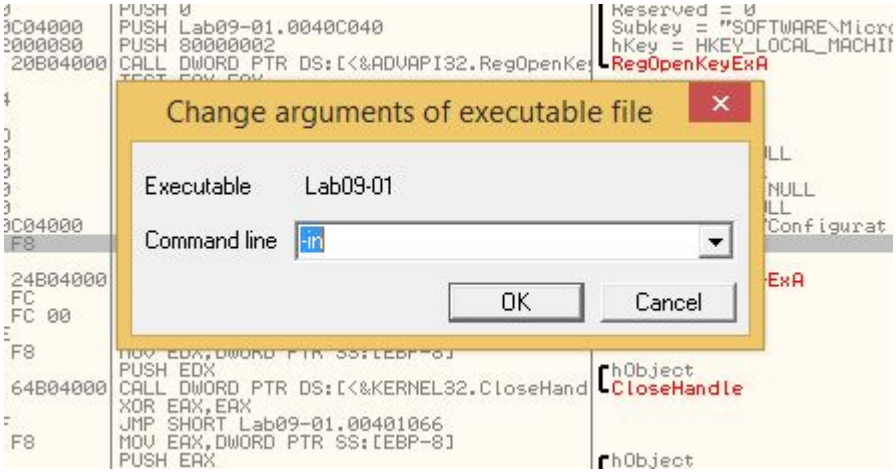


Fig 3: -in argument in WinDbg command line.

2. To check the password requirements for this program, using IDAPro we identify the main function. To identify the main function we first start by looking for calls to __mbscmp at 0040380F as shown below.

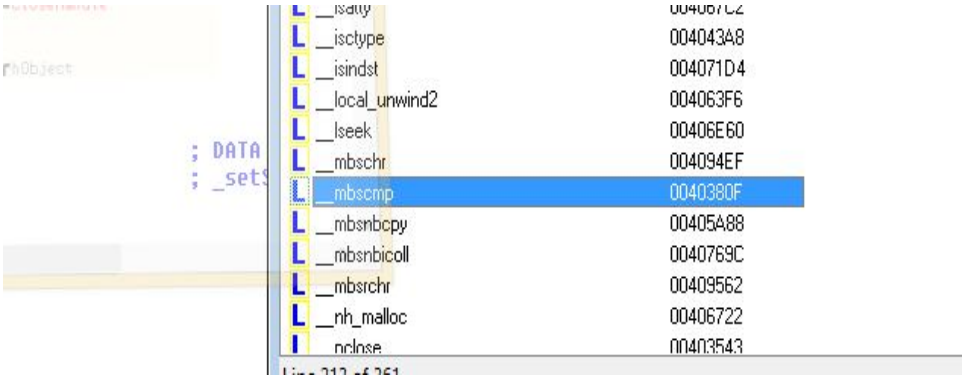


Fig 4: Locating __mbscmp using IDAPro.

Analyzing it and following the addresses of implementation shows that the program supports 4 command-line switches which are: -in, -re, -c, and –cc.

Googling what these switches do tells us that:

- -in - install a service
- -re - Uninstalls a service
- -c - Set a configuration key
    - -cc - Prints a configuration key

3. Full analysis of the main function 0x402510 in IDAPro will reveal the password, considering we are using the free version of IDAPro, we have limitations as to what can be done. However to bypass the password check we can always patch the code. In order to patch the binary, we first change the bytes of the function at 0x402510. We cant tell the assembly instruction for this behavior as a result of the limitations on IDAPro Free version.

   We attempted to patch the file using OllyDbg, to do that, we locate the function address, and in our case it is 0x402510. Right click on the start address and selecting **Binary – Edit** as shown in Fig 5 below:

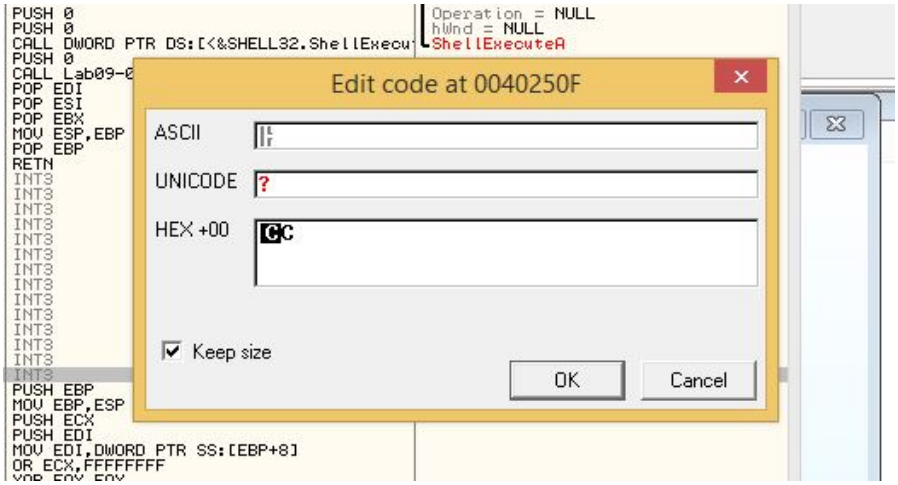


Fig 5: Binary Patching.

4. Analysis and viewing the strings using IDAPro shows Fig 6 below:

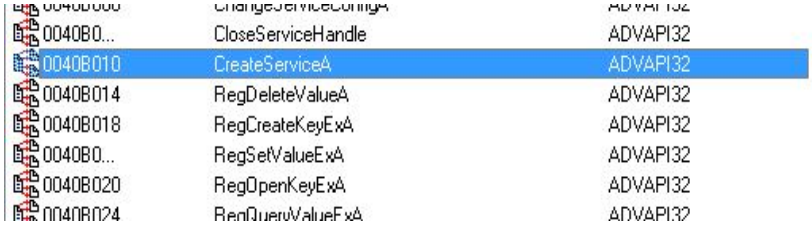


Fig 6: CreateServiceA string

From the figure above we can tell that the malware creates registry using the command CreateServiceA.

Checking the imports and locating the function at 0x00000018 we can tell that the malware creates the registry key **HKLM\Software\Microsoft\XPS** as shown in Fig 7 below:

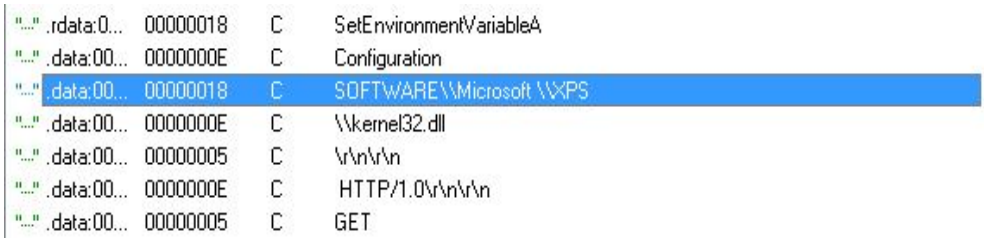


Fig 7: IDAPro Malware registry

Fig 7: OllyDbg Malware registry.

When the program copies itself into the Windows System directory, it may change the filename to match the service name.

5. Going through the strings again, we see that it contains the figure shown in Fig 8 below:

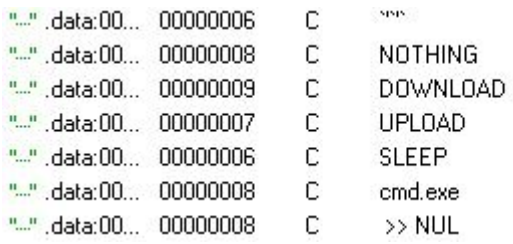


Fig 8: Supported Commands.

From the figure above, we can conclude that the malware can be instructed ti execute any of the five commands via the network: SLEEP, UPLOAD, DOWNLOAD, CMD or NOTHING.

Further research tells us what each command does:

- SLEEP - Sleeps for seconds
- UPLOAD - Upload ports filename.
- DOWNLOD - Download ports filename
- CMD - CMD port command
- NOTHING - No operation

UPLOAD and DOWNLOAD commands are usually reserved from their standard usage.

6. From previous labs, the solution to network-based signatures for the programs always ends up as the link to the creators website http://www.practicalmalwareanalysis.com/. The malware does not provide any HTTP headers with its request.

Lab 9-2

This program is analyzed using OllyDbg, before analysis using OllyDbg, we first load the malware into IDAPro to view its strings.

Running this malware does shows or does anything, the process launch and exit in Process Explorer, the process seems to terminate immediately. We will have to debug this malware to find out what is really going on.

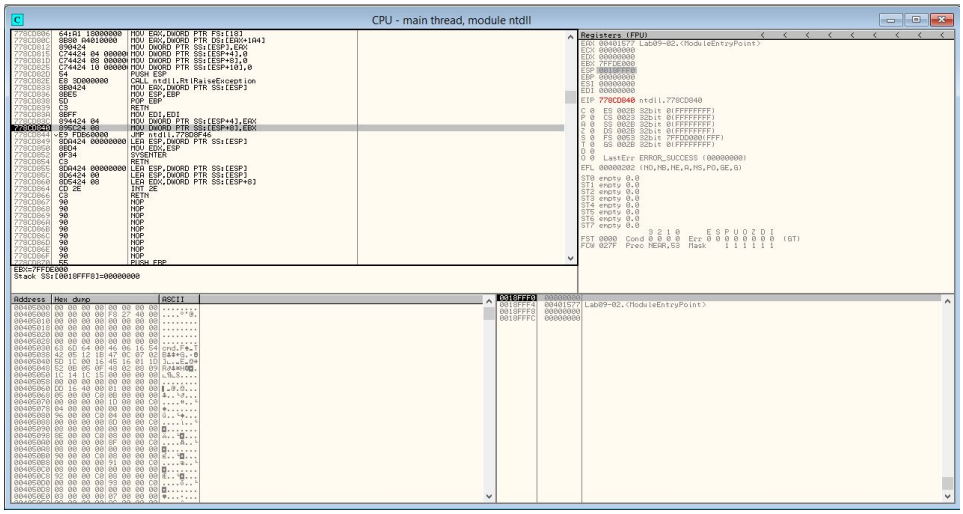


Fig 9: OllyDbg Analysis

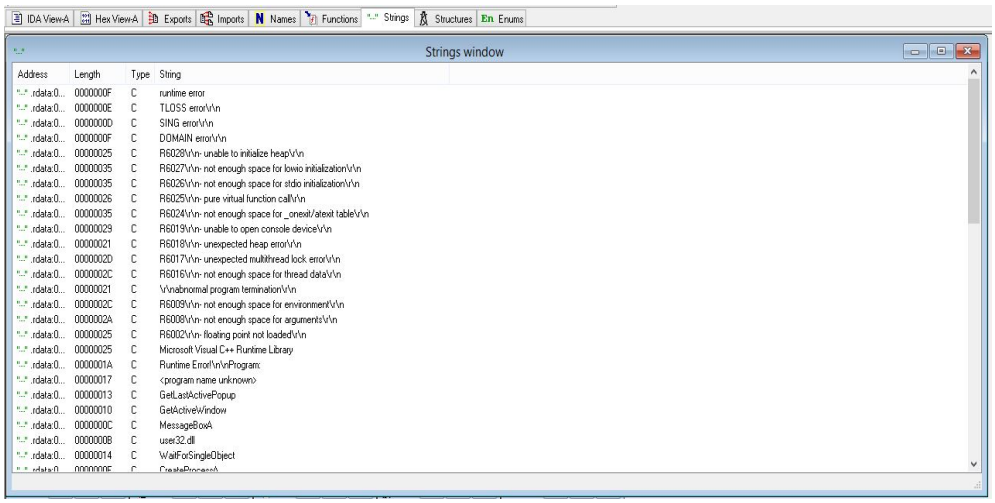


Fig 10: IDAPro Analysis.

Using figure 9 and 10 analysis, we were able to find the following answers:

1. The only most interesting import and string that appear statically in the binary is ***cmd***
2. Running the program does nothing, we are not sure if the program runs and terminates quickly or doesn't run at all.
3. The sample can run its malicious payload by renaming the file ocl.exe before running it. As shown below

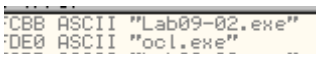


Fig 11: Ocl.exe

4. Lots of strings are being relocated and pushed to the stack, a string which is used by attackers to obfuscate strings from simple string utilities and basic analysis techniques is being built on the stack. That is what is happening at 0x00401133
5. The arguments being passed to subroutine 0x00401089 is not available. For some reason beyond our understanding, the string is not readable.
6. As with all the previous malwares, the malware uses the domain *practicalmalwareanalysis.com*
7. The encoding routine used to obfuscate the domain name is the logic gate XOR. This logic gate is a bit adder, meaning it adds two or more binary numbers with the help and combination of others logic gates such as AND. The malware will XOR the encoded DNS name with the string 1qaz2wsx3edc to decode the domain.
8. From the figure below:

```
lea     eax, [ebp+StartupInfo]
push    eax             ; lpStartupInfo
push    0               ; lpCurrentDirectory
push    0               ; lpEnvironment
push    0               ; dwCreationFlags
push    1               ; bInheritHandles
push    0               ; lpThreadAttributes
push    0               ; lpProcessAttributes
push    offset CommandLine ; "cmd"
push    0               ; lpApplicationName
call    ds:CreateProcessA
```

Fig 12: Malware Info

The malware is setting the stdout, stderr and stdin handles (used in the STARTUOINFO structure of CreateProcessA) to the socket. Since CreateProcessA is called with cmd as an argument, this will create a reverse shell by tying the command shell to the socket.

Lab 9-3

We start by running this file using OllyDbg as shown in Fig 13 below:

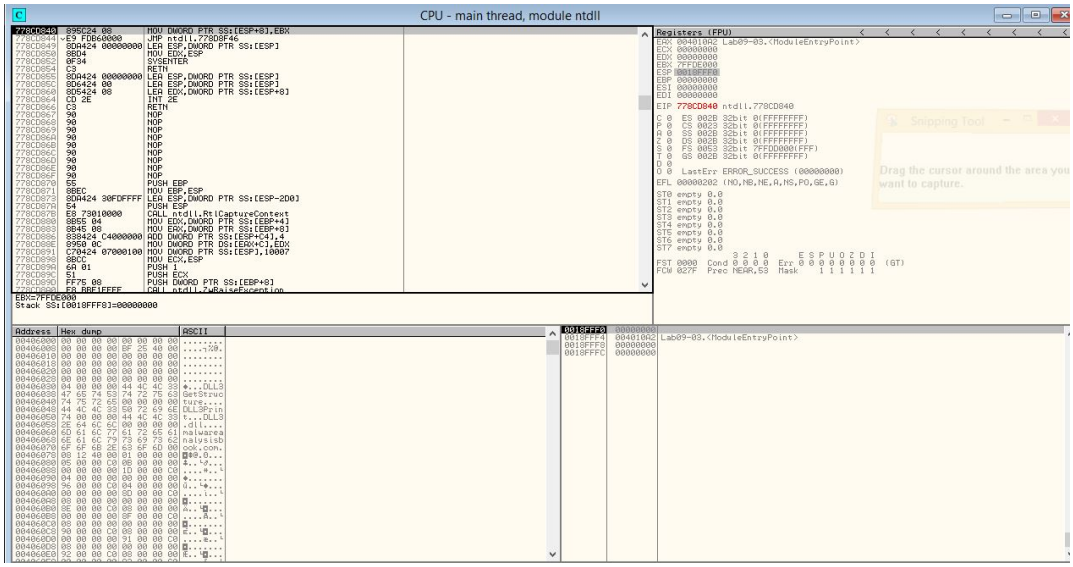


Fig 13: OllyDbg

We then run the malware in IDAPro to view the functions, strings and imports as shown in Fig 14, 15 and 16 respectively.

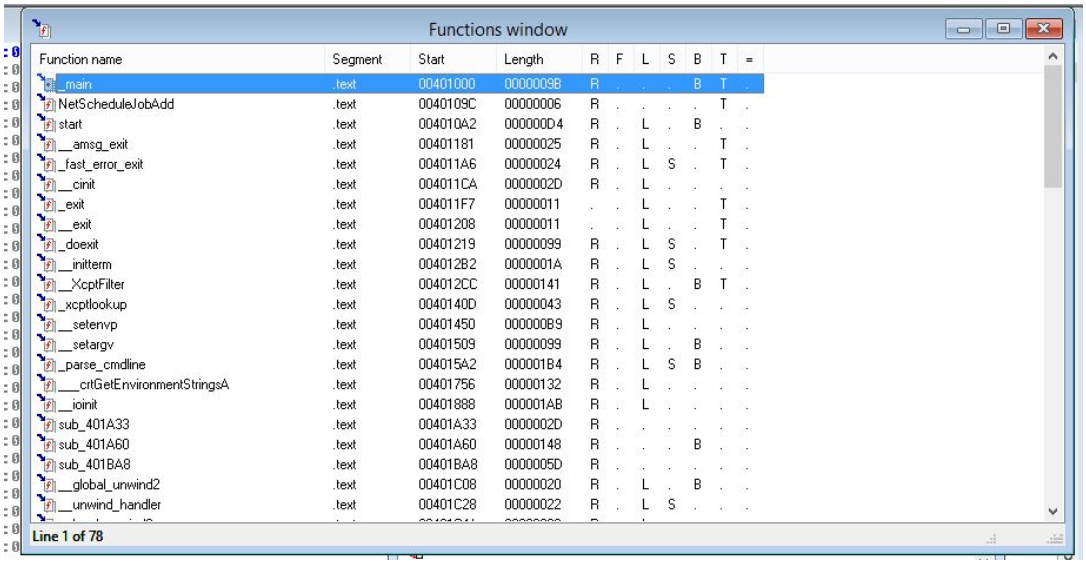


Fig 14: Functions

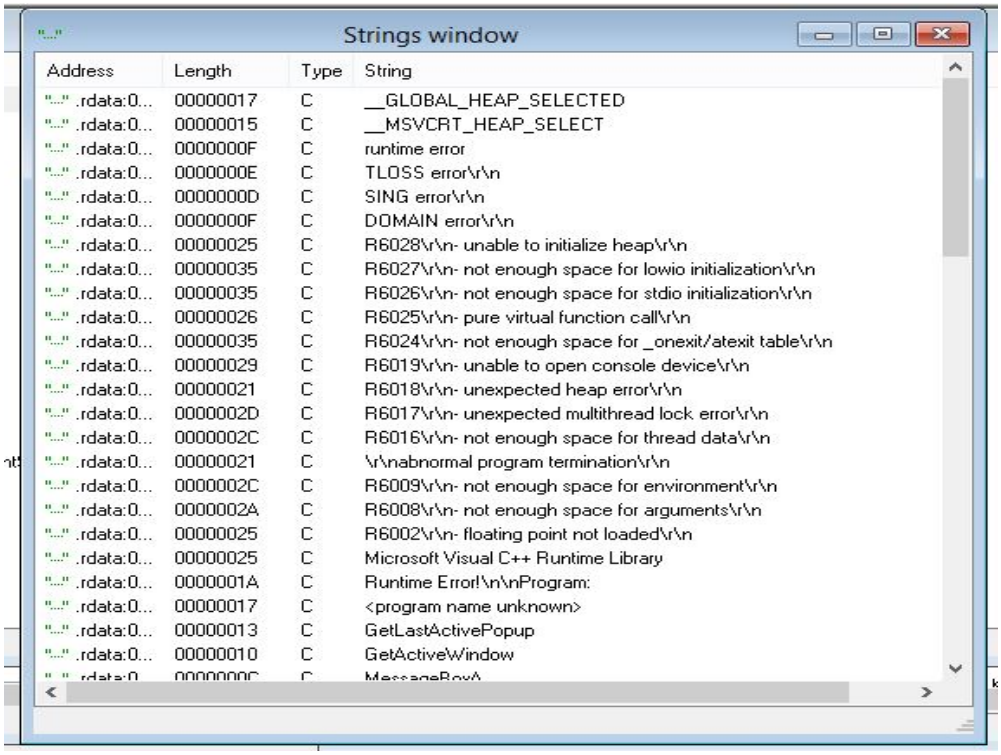


Fig 15: Strings

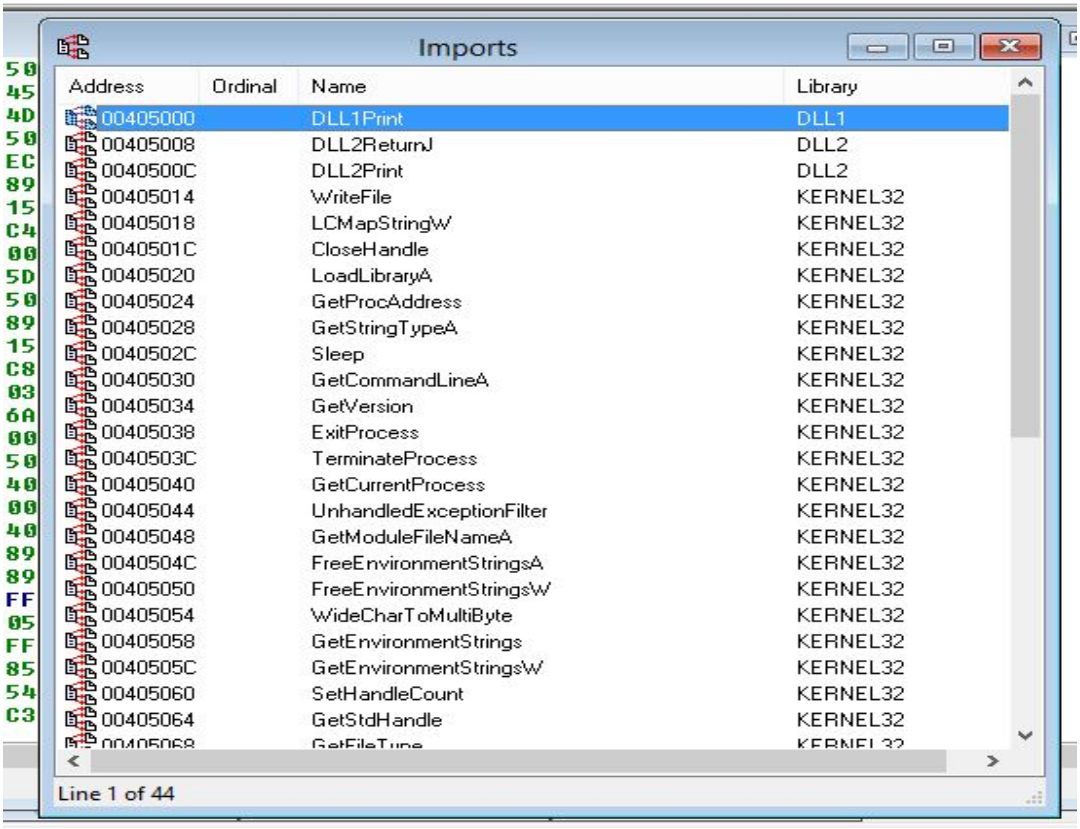


Fig 16: Imports.

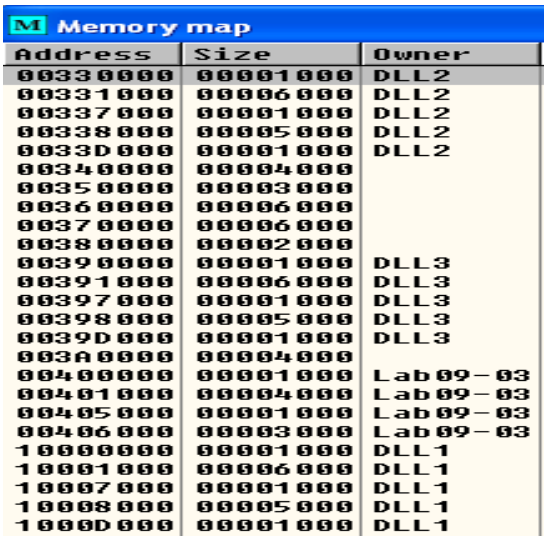
From these figures extracted from OllyDbg and IDAPro we can answer the following questions:

1. The DLLs imported by Lab09-03.exe as shown in the imports table in Fig 16 contains Kernel32.dll, NetAPI32.dll, DLL1.dll and DLL2.dll. The malware dynamically loads user32.dll and DLL3.dll
2. The base address requested by DLL1.dll, DLL2.dll and DLL3.dll are all the same and it is the base address 0x10000000 as shown in Fig 17 below:

Fig 17

3. The assigned based address for DLL1.dll, DLL2.dll and DLL3.dll using OllyDbg are as follows:
    - DLL1.dll is located at 0x10000000
    - DLL2.dll is located at 0x330000 and
    - DLL3.dll is located at 0x390000

These can be proved from Fig 18 below:



4. When Lab09-03.exe calls an import function from DLL1.dll. DLL1Print is called, and it prints “DLL 1 mystery data” followed by the contents of a global variable.
5. It returns a finlename of temp.txt which is passed to the call WriteFile
6. It gets the data for the second parameter from DLL3GetStructure, which it dynamically resolves.
7. Mystery data 1 is the current process identifier, mystery data 2 is the handle to the open temp.txt file and mystery data 3 is the location in memory of the string ping *www.malwareanalysisbook.com*
8. When using IDAPro, select Manual Load, that way it will matches the load address used by OllyDbg.

Conclusion

The labs aim to provide hands on experience of how to use the most popular user-mode debugger for malware analysis which is OllyDbg. Upon completion of this lab we see that the Lab09-01.exe file is a HTTP reverse backdoor sample, which registers a key to fetch server configuration information and make HTTP requests to the remote system. The Lab09-03.exe demonstrates how to determine where three DLLs are loaded into using OllyDbg.