Lab 09-03

Analyze the malware found in the file Lab09-03.exe using OllyDbg and IDA Pro.

This malware loads three included DLLs (DLL1.dll, DLL2.dll, and DLL3.dll) that are all built to request the same memory load location.

Therefore, when viewing these DLLs in OllyDbg versus IDA Pro, code may appear at different memory locations. The purpose of this lab is to make you comfortable with finding the correct location of code within IDA Pro when you are looking at code in OllyDbg.

Contents

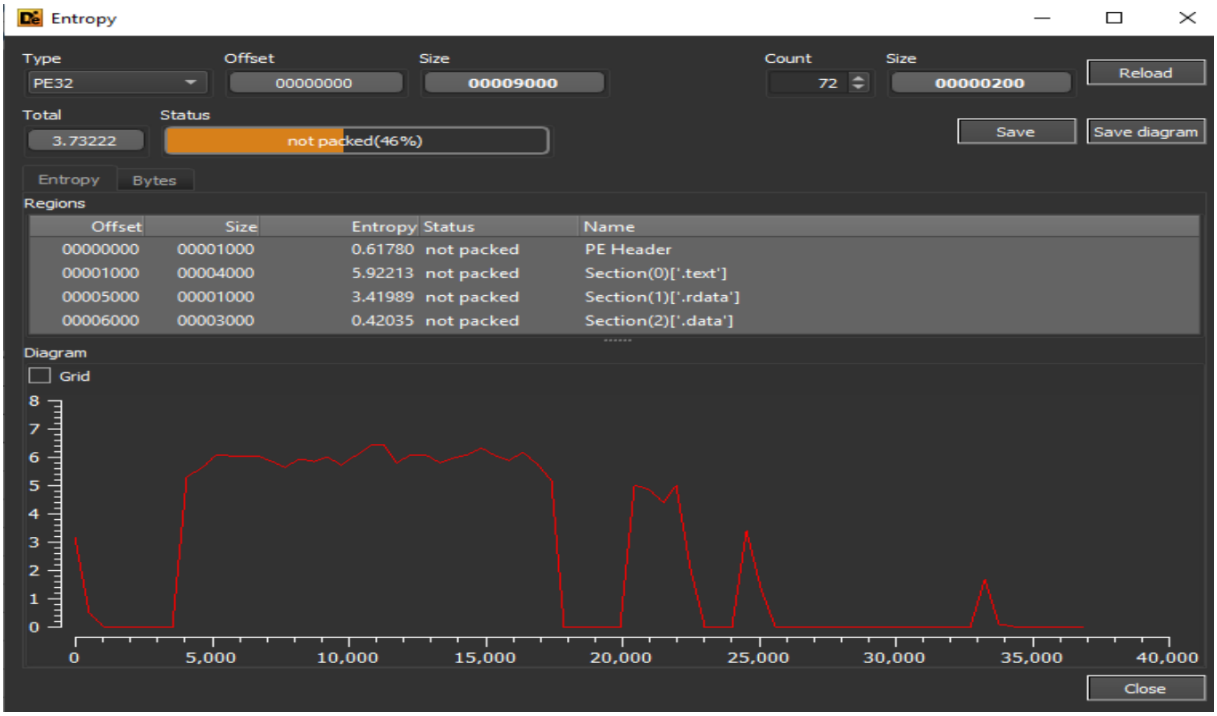
Preeliminary analysis of binaries

In this Lab, we are given four files:

- Lab09-03.exe
- DLL1.dll
- DLL2.dll
- DLL3.dll

Lab09-03.exe

File written in C/C++, total entropy of 3.73, sections sizes are normal, imports are not hidden and strings are clearly readable.



Looks like there is no sign of any packaging nor obfuscating.

Interesting strings found:

- Malwareanalysisbook.com
- DLL1Print
- DLL2ReturnJ
- DLL2Print
- DLL3GetStructure
- DLL3Print

When it comes to Imports, there is an usage of:

- KERNEL32.dll

- NETAPI32.dll
- DLL1.dll
- DLL2.dll

There is no information about DLL3.dll even though we could find string “DLL3Print” and we were provided the file.

With the interesting functions we have:

- Memory management (VirtualAlloc, HeapAlloc)
- Dynamic library loading (LoadLibraryA, GetProcAddress)
- Creating job (NetScheduleJobAdd)

DLL1.dll

As previously briefly analysed executable, the first dll seems not packed nor obfuscated as well.

Import functions:

- Process & thread management
- Memory management (VirtualAlloc, HeapAlloc)

Export functions:

- DLL1Print

Interesting strings:

- DLL 1 mystery data %d

DLL2.dll

The library is uncompressed and remains unobfuscated.

Import functions:

- File management (CreateFileA, WriteFile)
- Memory management (VirtualAlloc, HeapAlloc)
- Dynamic library loading (LoadLibraryA, GetProcAddress)

Export functions:

- DLL2Print
- DLL2ReturnJ

Interesting strings:

- DLL 2 mystery data %d
- temp.txt

DLL3.dll

The last provided library is also not protected in any way from analysis.

Import functions:

- File and handle management (CreateFileA, WriteFile)
- Memory management (VirtualAlloc, HeapAlloc)
- Dynamic library loading (LoadLibraryA, GetProcAddress)
- Critical Section Operations

Export functions:

- DLL3Print
- DLL3GetStructure

Interesting strings:

- DLL 3 mystery data %d
- ping www.malwareanalysisbook.com

IDA & x32dbg Analysis

Let's do a brief analysis of the executable.

During executing the malware it preloads two externally provided libraries: a dll1 and dll2.

A quick debugging session show that after loading the libraries:

- dll1 gets a currentprocess id and stores it in a variable

```
.text:10001000 ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:10001000 _DllMain@12     proc near               ; CODE XREF: DllEntryPoint+4B↓p
.text:10001000
.text:10001000 hinstDLL        = dword ptr  8
.text:10001000 fdwReason       = dword ptr  0Ch
.text:10001000 lpvReserved     = dword ptr  10h
.text:10001000
.text:10001000                 push    ebp
.text:10001001                 mov     ebp, esp
.text:10001003                 call    ds:GetCurrentProcessId
.text:10001009                 mov     dword_10008030, eax
.text:1000100E                 mov     al, 1
.text:10001010                 pop     ebp
.text:10001011                 retn    0Ch
.text:10001011 _DllMain@12     endp
```

- dll2 creates an empty file "temp.txt" located in malware's directory.

```
.text:10001000 ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:10001000 _DllMain@12     proc near               ; CODE XREF: DllEntryPoint+4B↓p
.text:10001000
.text:10001000 hinstDLL        = dword ptr  8
.text:10001000 fdwReason       = dword ptr  0Ch
.text:10001000 lpvReserved     = dword ptr  10h
.text:10001000
.text:10001000                 push    ebp
.text:10001001                 mov     ebp, esp
.text:10001003                 push    0               ; hTemplateFile
.text:10001005                 push    80h             ; dwFlagsAndAttributes
.text:1000100A                 push    2               ; dwCreationDisposition
.text:1000100C                 push    0               ; lpSecurityAttributes
.text:1000100E                 push    0               ; dwShareMode
.text:10001010                 push    40000000h       ; dwDesiredAccess
.text:10001015                 push    offset FileName ; "temp.txt"
.text:1000101A                 call    ds:CreateFileA
.text:10001020                 mov     dword_1000B078, eax
.text:10001025                 mov     al, 1
.text:10001027                 pop     ebp
.text:10001028                 retn    0Ch
.text:10001028 _DllMain@12     endp
```

Analysis of the main code of the executable tells us that there are three calls to the libraries stored with the malware:

```
.text:00401006              call    ds:DLL1Print
.text:0040100C              call    ds:DLL2Print
.text:00401012              call    ds:DLL2ReturnJ
```

DLL1Print prints the text into console "DLL 1 mystery data %d\n" where %d is a current process id that was saved at the dll entry preload.

DLL2Print prints the text into console "DLL 2 mystery data %d\n" where %d is a handle to a created file "temp.txt".

DLL2ReturnJ returns the handle of a "temp.txt" into the executable for further usage.

Later, at **0x401023** we have a piece of code that's intended to input the string "malwareanalysisbook.com" into the "temp.txt" using the handle gathered from DLL2ReturnJ call.

```
.text:00401023                push    offset aMalwareanalysi ; "malwareanalysisbook.com"
.text:00401028                mov     ecx, [ebp+hFile]
.text:0040102B                push    ecx             ; hFile
.text:0040102C                call    ds:WriteFile
.text:00401032                mov     edx, [ebp+hFile]
.text:00401035                push    edx             ; hObject
.text:00401036                call    ds:CloseHandle
```

And that's followed by dynamically loading third library called "DLL3.dll" and its two functions: DLL3Print and DLL3Structure. A quick usage of x-ref at LoadLibraryA call reveals that it dynamically attaches also user32.dll later in the code.

DLL3 entry main stores a string “ping malwareanalysisbook.com” in the memory and initializes a AT_INFO structure.

```
.text:00A31000
.text:00A31000                  push    ebp
.text:00A31001                  mov     ebp, esp
.text:00A31003                  push    ecx
.text:00A31004                  mov     [ebp+lpMultiByteStr], offset aPingWwwMalware ; "ping www.malwareanalysisbook.com"
.text:00A3100B                  push    32h ; '2'       ; cchWideChar
.text:00A3100D                  push    offset WideCharStr ; lpWideCharStr
.text:00A31012                  push    0FFFFFFFFh      ; cbMultiByte
.text:00A31014                  mov     eax, [ebp+lpMultiByteStr]
.text:00A31017                  push    eax             ; lpMultiByteStr
.text:00A31018                  push    0               ; dwFlags
.text:00A3101A                  push    0               ; CodePage
.text:00A3101C                  call    ds:MultiByteToWideChar
.text:00A31022                  mov     stru_A3B0A0.Command, offset WideCharStr
.text:00A3102C                  mov     stru_A3B0A0.JobTime, 36EE80h
.text:00A31036                  mov     stru_A3B0A0.DaysOfMonth, 0
.text:00A31040                  mov     stru_A3B0A0.DaysOfWeek, 7Fh
.text:00A31047                  mov     stru_A3B0A0.Flags, 11h
.text:00A3104E                  mov     al, 1
.text:00A31050                  mov     esp, ebp
.text:00A31052                  pop     ebp
.text:00A31053                  retn    0Ch
.text:00A31053 _DllMain@12      endp
.text:00A31053
```

DLL3Print prints the text into console “DLL 3 mystery data %d\n” where %d is a memory location of string “ping malwareanalysisbook.com”.

DLL3GetStructure returns a pointer to a structure initialized at DLL3 entry.

Questions

1. What DLLs are imported by Lab09-03.exe?

Lab09-03.exe imports:

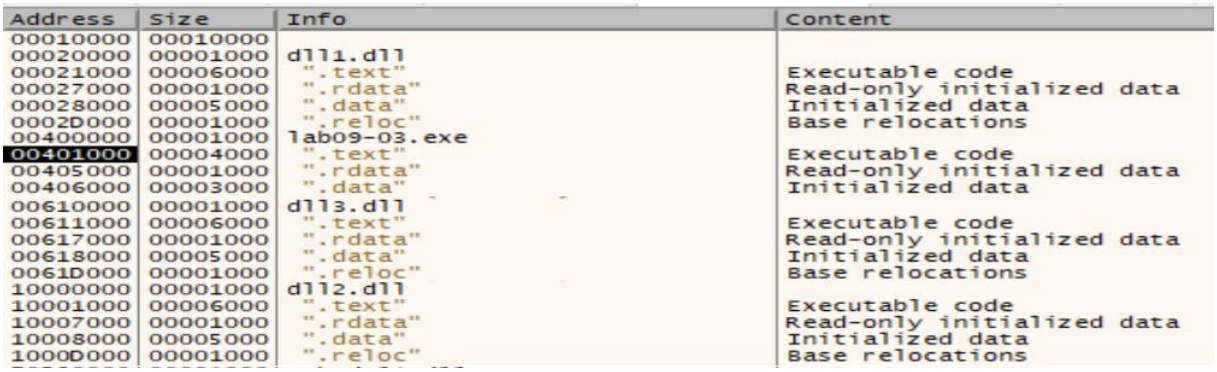
- KERNEL32.dll
- NETAPI32.dll
- DLL1.dll
- DLL2.dll
- DLL3.dll (dynamically by LoadLibraryA)

2. What is the base address requested by DLL1.dll, DLL2.dll, and DLL3.dll?

The base addresses for all those three dll's are the same - 0x10000000.

3. When you use OllyDbg to debug Lab09-03.exe, what is the assigned based address for: DLL1.dll, DLL2.dll, and DLL3.dll?

Well, the base address of the dll's differ each time the application runs.



Address	Size	Info	Content
00010000	00010000		
00020000	00001000	dll1.dll	
00021000	00006000	".text"	Executable code
00027000	00001000	".rdata"	Read-only initialized data
00028000	00005000	".data"	Initialized data
0002D000	00001000	".reloc"	Base relocations
00400000	00001000	lab09-03.exe	
00401000	00004000	".text"	Executable code
00405000	00001000	".rdata"	Read-only initialized data
00406000	00003000	".data"	Initialized data
00610000	00001000	dll3.dll	
00611000	00006000	".text"	Executable code
00617000	00001000	".rdata"	Read-only initialized data
00618000	00005000	".data"	Initialized data
0061D000	00001000	".reloc"	Base relocations
10000000	00001000	dll2.dll	
10001000	00006000	".text"	Executable code
10007000	00001000	".rdata"	Read-only initialized data
10008000	00005000	".data"	Initialized data
1000D000	00001000	".reloc"	Base relocations

4. When Lab09-03.exe calls an import function from DLL1.dll, what does this import function do?

DLL1 entry gets an ID of current running process and stores it in a variable dword_10008030.

```
.text:10001000 ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:10001000 _DllMain@12      proc near               ; CODE XREF: DllEntryPoint+4B↓p
.text:10001000
.text:10001000 hinstDLL         = dword ptr  8
.text:10001000 fdwReason        = dword ptr  0Ch
.text:10001000 lpvReserved      = dword ptr  10h
.text:10001000
.text:10001000                  push    ebp
.text:10001001                  mov     ebp, esp
.text:10001003                  call    ds:GetCurrentProcessId
.text:10001009                  mov     dword_10008030, eax
.text:1000100E                  mov     al, 1
.text:10001010                  pop     ebp
.text:10001011                  retn    0Ch
.text:10001011 _DllMain@12      endp
```

DLL1Print prints the text into console “DLL 1 mystery data %d\n” where %d is a current process id that was saved at the dll entry preload.

5. When Lab09-03.exe calls WriteFile, what is the filename it writes to?

It writes data into file called “temp.txt” that’s located under currently running malware’s directory.

6. When Lab09-03.exe creates a job using NetScheduleJobAdd, where does it get the data for the second parameter?

It gets data from the buffer that is filled from DLL3Structure call.

7. While running or debugging the program, you will see that it prints out three pieces of mystery data. What are the following: DLL 1 mystery data 1, DLL 2 mystery data 2, and DLL 3 mystery data 3?

DLL1Print prints the text “DLL 1 mystery data %d\n” where %d is a current process id that was saved at the dll entry preload.

DLL2Print prints the text “DLL 2 mystery data %d\n” where %d is a handle to a created file “temp.txt”.

DLL3Print prints the text “DLL 3 mystery data %d\n” where %d is a decimal value of a memory address of a string containing data “ping malwareanalysisbook.com”.

```
DLL 1 mystery data 5976
DLL 2 mystery data 228
DLL 3 mystery data 6074560
```

8. How can you load DLL2.dll into IDA Pro so that it matches the load address used by OllyDbg?

First of all, load the library in a debugger and find the base memory address of the DLL2.dll, then load DLL2.dll in IDA with selection of Manual Load, and write the gathered from the debugger base memory address.