CNIT 58100 CFM: CYBERFORENSICS OF MALWARE – LAB 15

Ibrahim Waziri Jr
PhD in Information Security (CERIAS)
Lab 15

Instructor: **Associate Prof Sam Liles**
Purdue University
2014

Abstract

This lab covers the skills discussed in chapter 15 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.
This paper provides answers to Chapter 15 labs. The lab uses 3 different files which are: *Lab10-01.exe, Lab10-02.exe and Lab10-03.exe.* These files are malwares are therefore could be harmful if used for non-training purposes.

Lab 15-1

1. As shown in the figure below: We load the program Lab15-01 into IDAPro, and we can that the anti-disassembly technique used in the binary by this malware are false conditional branches: an xor eax and jz as shown below:

```
.text:00401000 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401000 _main:                                  ; CODE XREF: start+DE↓p
.text:00401000                 push    ebp
.text:00401001                 mov     ebp, esp
.text:00401003                 push    ebx
.text:00401004                 push    esi
.text:00401005                 push    edi
.text:00401006                 cmp     dword ptr [ebp+8], 2
.text:0040100A                 jnz     short loc_40105E
.text:0040100C                 xor     eax, eax
.text:0040100E                 jz      short near ptr loc_401010+1
.text:00401010
```

2. The malware tricks the disassembler into disassembling the code by using the first of the 5-byte call instruction, which is immediately followed by the anti-disassembly technique jz as shown below:

```
.text:0040100C                 xor     eax, eax
.text:0040100E                 jz      short near ptr loc_401010+1
.text:00401010
.text:00401010 loc_401010:                             ; CODE XREF: .text:0040100E↑j
.text:00401010                 call    near ptr 8B4C55A0h
.text:00401015                 dec     eax
.text:00401016                 add     al, 0Fh
.text:00401018                 mov     esi, 70FA8311h
.text:0040101D                 jnz     short loc_40105E
```

3. With reference to the first question, we can see that the technique is used 5 times.

4. From the command instructions shown below we can see that the command-line argument pdq will cause the program to print “Good Job!” as shown below:

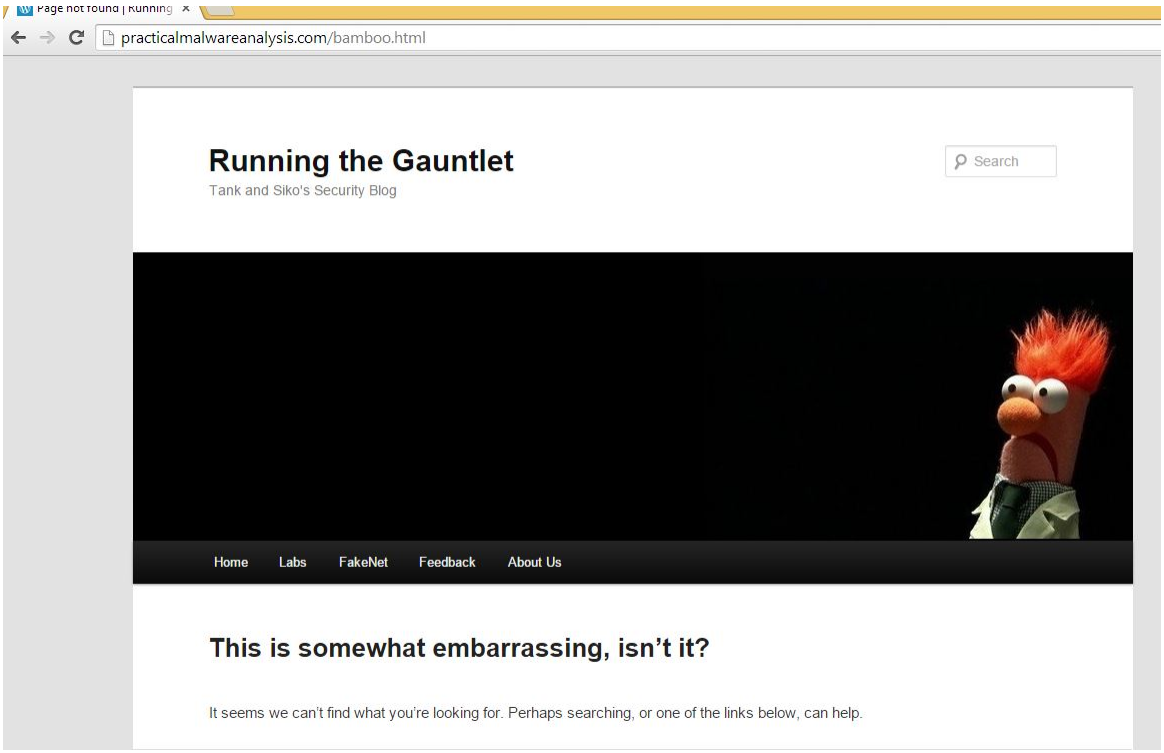
```
.text:0040104B loc_40104B:                             ; CODE XREF: .text:00401049↑j
.text:0040104B                 call    near ptr 40702088h
.text:00401050                 add     bh, bh
.text:00401052                 adc     eax, offset printf
.text:00401057                 add     esp, 4
.text:0040105A                 xor     eax, eax
.text:0040105C                 jmp     short loc_401073
.text:0040105E ; ---------------------------------------------------------------------------
.text:0040105E
.text:0040105E loc_40105E:                             ; CODE XREF: .text:0040100A↑j
.text:0040105E                                         ; .text:0040101D↑j ...
.text:0040105E                 xor     eax, eax
.text:00401060                 jz      short near ptr loc_401062+1
.text:00401062
.text:00401062 loc_401062:                             ; CODE XREF: .text:00401060↑j
.text:00401062                 call    near ptr 40702CCFh
.text:00401067                 add     bh, bh
.text:00401069                 adc     eax, offset printf
.text:0040106E                 add     esp, 4
.text:00401071                 xor     eax, eax
.text:00401073
.text:00401073 loc_401073:                             ; CODE XREF: .text:0040105C↑j
.text:00401073                 pop     edi
.text:00401074                 pop     esi
```

Lab 15-2

1. From the figure below we can see a call to InternetOpenUrlA, following that call URL initially requested opens http://www.practicalmalwareanalysis.com/bamboo.html

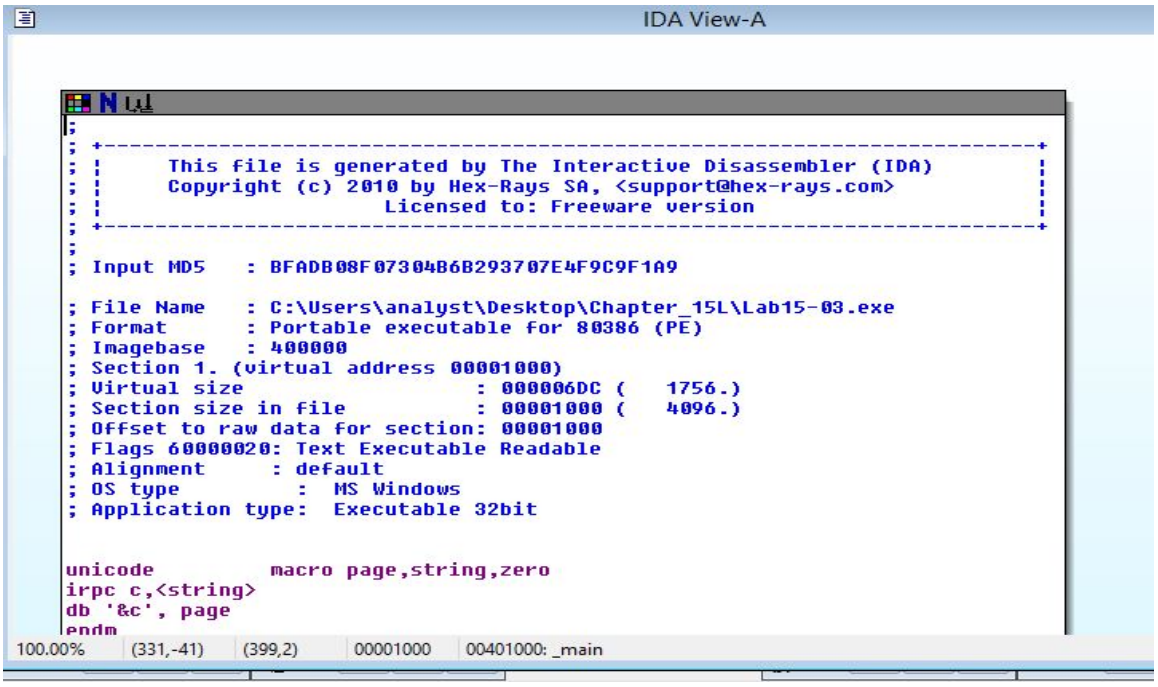
```
.text:00401173                 push    eax
.text:00401174                 call    ds:InternetOpenUrlA
.text:0040117A                 mov     [ebp-29Ch], eax
.text:00401180                 cmp     dword ptr [ebp-29Ch], 0
```

2. The User-Agent is generated by adding 1 to each letter and number in the hostname.
3. The program looks for the string Bamboo in the page it requested, even though the page appears to be offline.
4. The page is offline, and can therefore when the program runs it doesn't extract anything, because the page has no content as shown below.

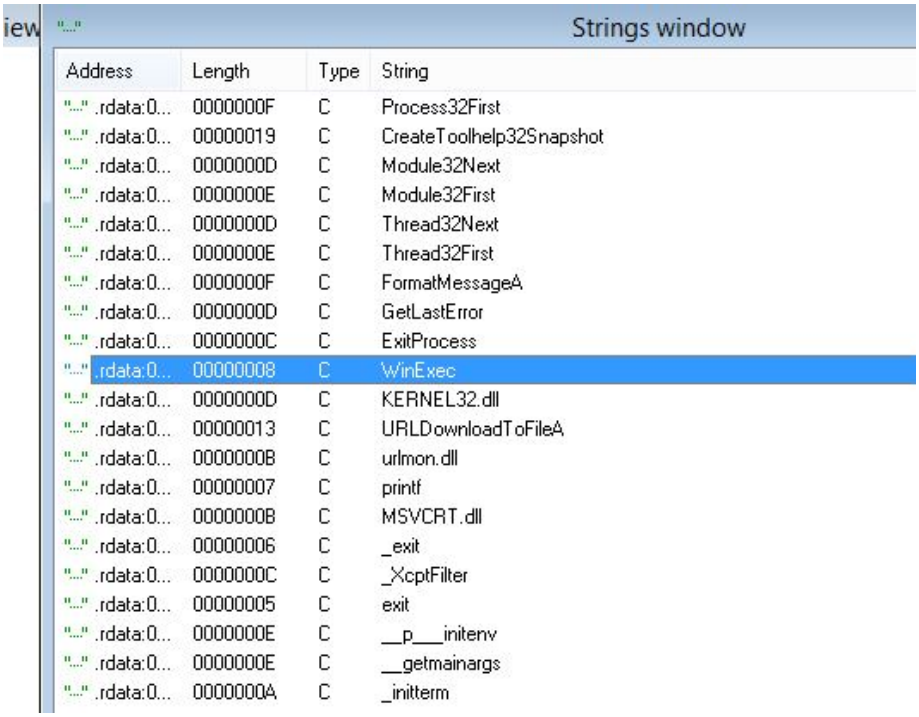


Lab 15-3

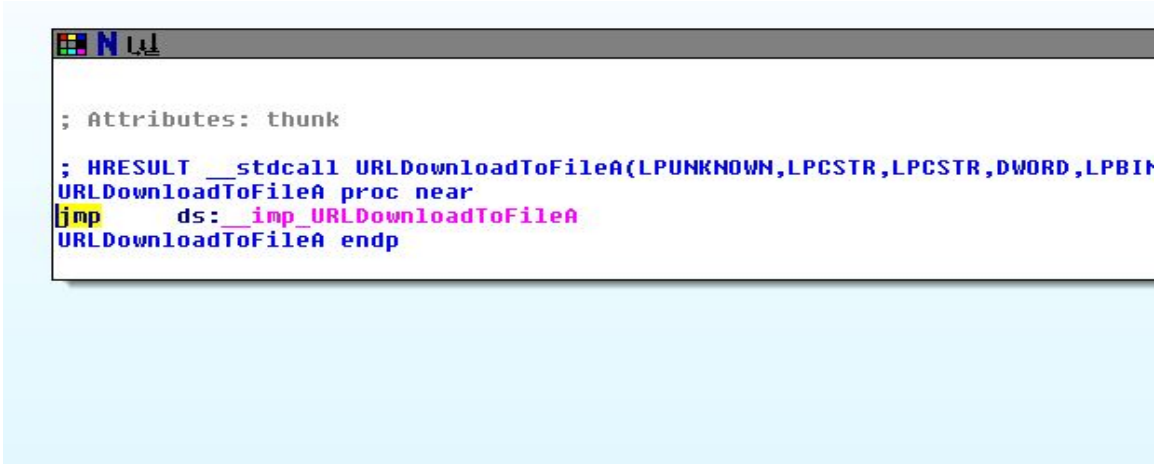
1. The malicious code is initially called by overwriting the return pointer from the main function.



2. From the strings, we can see that the malicious code downloads a file from a URL and launches it with WinExec as as shown below:



3. We see the string URLDownloadToFileA, and following that address shows us the URL used by the program, which is http://www.practicalmalwareanalysis.com/tt.html as shown below.



4. The filename the malware uses is spoolsrv.exe. We couldn't capture the name, because the cmd prompt page runs so fast, and ends quickly whenever we run the program.

Conclusion:

When malware authors create malwares, they manually use disassembly crafted codes in the malware to cause disassembly tools to produce an incorrect program listing. In these labs we see how this disassembly codes are in a malware and how we can bypass it.