Lab 9: Working with IDA

Solutions

Overview

Today we will practice advanced static analysis using IDA. The goal is to gain familiarity with IDA and advanced static analysis, not necessarily to perform a full analysis using IDA.

Part 1: Crack the file(s) using the graph view!

1. Download the "IDA_practice_crackme.7z" archive from Canvas, move it to your Windows VM, and unzip it using "cse434" as the password.

2. For each of the files:

- Run the program in the command-line. Explain what the program does.
- Use IDA to analyze the executable and crack the password. (Note: you will need to download an older version of IDA to run on our Windows 7 VM. The older version (version 5.0) is available as a link from this post: https://www.scummvm.org/news/20180331/ .)
- Write abstract pseudocode to describe the program flow you identified.

---

* crackme-1.exe

1. It looks like a program with a password crack challenge.

2. The password is 'topsecret'

3. pseudocode

```
Main() {
        If there is no input parameter, then print 'Usage: crackme-123-1 password'
        Check a user input param with the 'topsecret'
```

```
    If succ, then print ‘You found the password!  Congratulations!’
    Else, then print ‘Fail!’
}
```

* crackme-2.exe

1. It looks like a program with a password crack challenge.
2. The password is ‘alligator’
3. pseudocode

```
Main() {
    If there is no input parameter, then print ‘Usage: crackme-123-2 password’
    Check a user input param with the ‘alligator’
    If succ, then print ‘You found the password!  Congratulations!’
    Else, then print ‘Fail!’
}
```

* crackme-3.exe

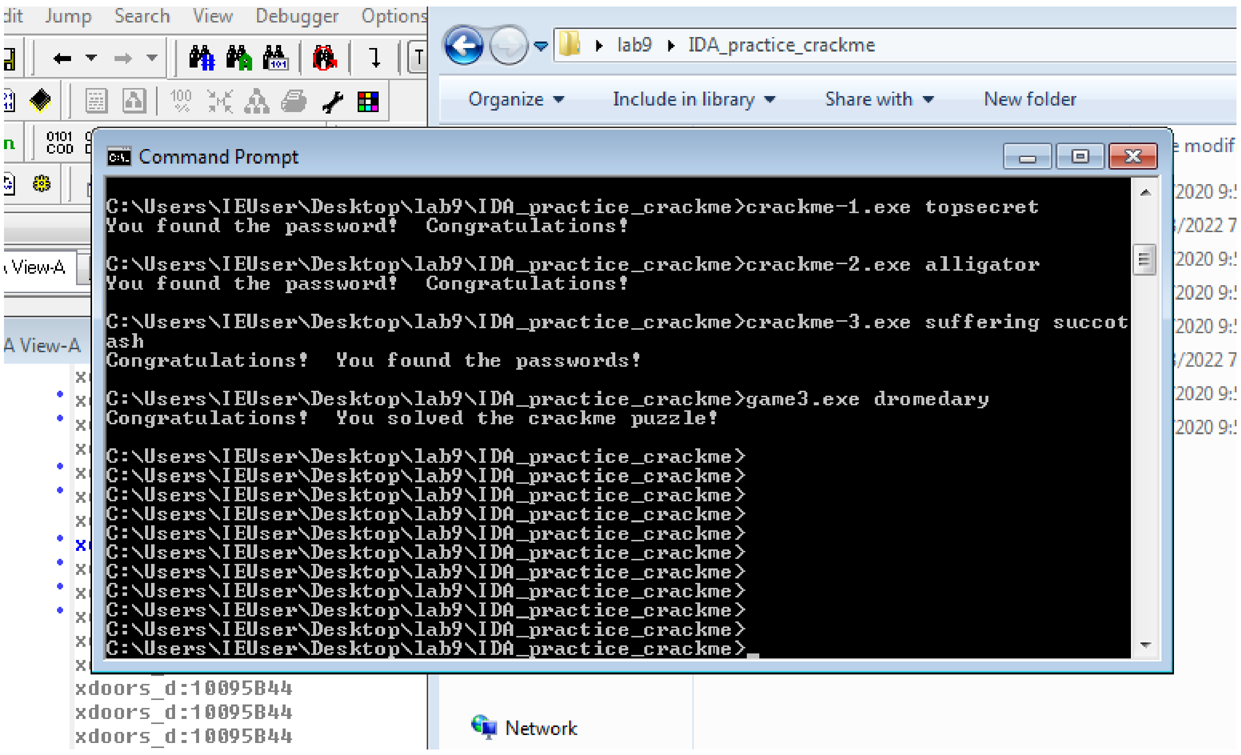
1. It looks like a program with a password crack challenge.
2. You need to give two passwords in order. ‘suffering’ and ‘succotash’. You need to give them in order.
3. pseudocode

```
Main() {
    If there are not 2 input parameters, then print ‘Usage: crackme-123-3 password1 password2’
    Check user’s first input param with the ‘suffering’
    If first input param doesn’t match, then print ’ Fail!  First word was wrong!’
    If second input param doesn’t match, then print ‘Fail!  Second word was wrong!’
    If two parameters are correct, then print ‘Congratulations!  You found the passwords!’
}
```

* crackme-4.exe

1. It looks like a program with a password crack challenge.
2. The password is ‘dromedary’. But before giving the password, you need to change the file name from ‘crackme-4.exe’ to ‘game3.exe’
3. pseudocode

```
Main() {
        If there is no input parameters, then print 'Usage: game3.exe password'
        Check execution file name and user's first input param.
        If execution file name is 'game3.exe' and the first parameter is 'dromedary',
then print ' Congratulations!  You solved the crackme puzzle!'
        Else print 'Fail!'
}
```



```
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>crackme-1.exe topsecret
You found the password!  Congratulations!

C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>crackme-2.exe alligator
You found the password!  Congratulations!

C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>crackme-3.exe suffering succot
ash
Congratulations!  You found the passwords!

C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>game3.exe dromedary
Congratulations!  You solved the crackme puzzle!

C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
C:\Users\IEUser\Desktop\lab9\IDA_practice_crackme>
```

```
xdoors_d:10095B44
xdoors_d:10095B44
xdoors_d:10095B44
```

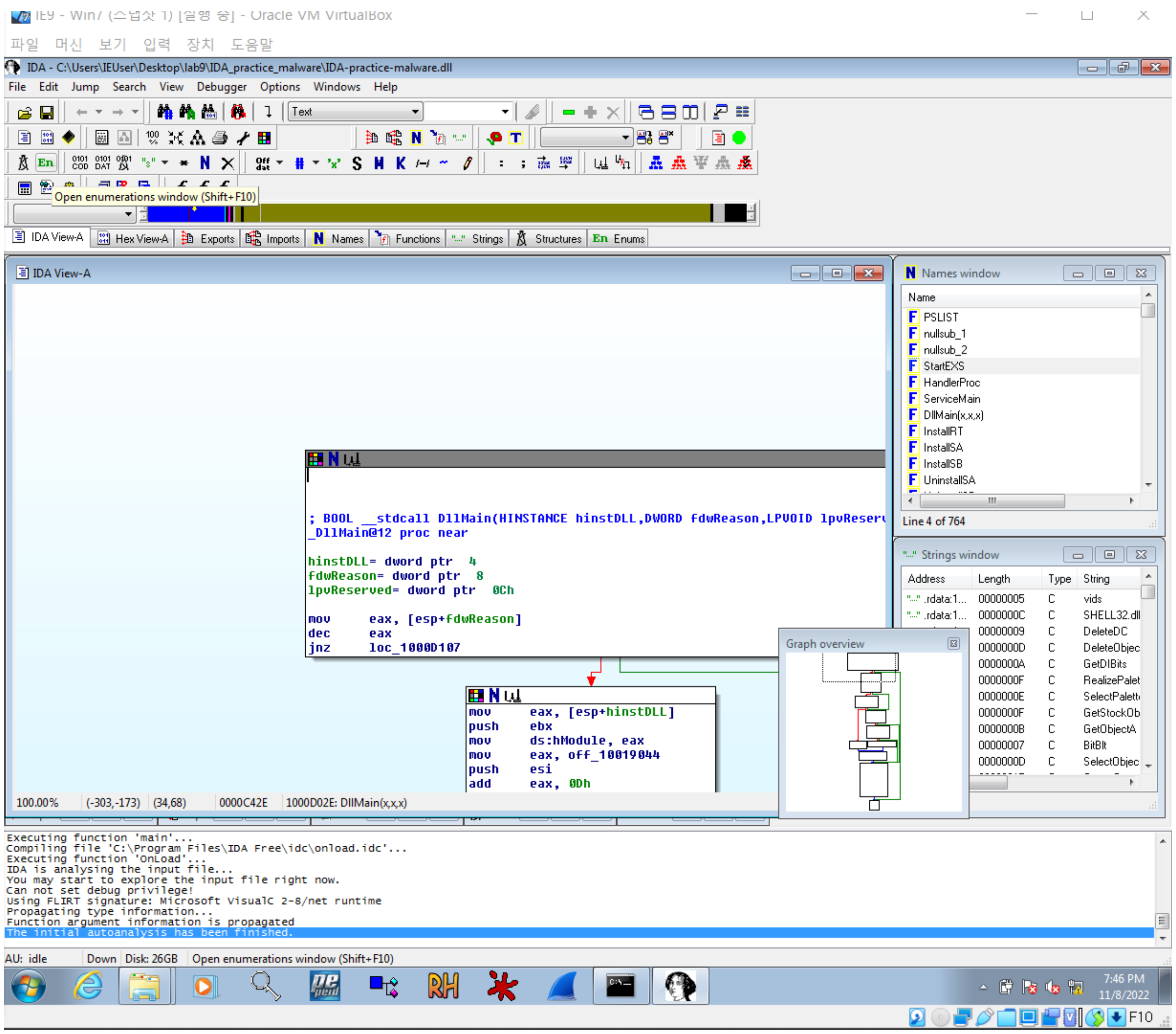
Part 2: More IDA functionality

Download the "IDA_practice_malware.7z" archive from Canvas, move it to your Windows VM, and unzip it using "infected" as the password. This part of the lab is based on chapter 5 of "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software."

Launch IDA and open 'IDA_practice_malware.dll'. Make sure you're viewing the "IDA View-A" window. Press the spacebar to see the code.

Switch back to the "graph mode", and repeat the same steps to display this additional information in "graph mode".

Take a screenshot of what you see:


IE9 - Win7 (스냅샷 1) [실행 중] - Oracle VM VirtualBox
파일 머신 보기 입력 장치 도움말
IDA - C:\Users\IEUser\Desktop\lab9\IDA_practice_malware\IDA-practice-malware.dll
File Edit Jump Search View Debugger Options Windows Help
Open enumerations window (Shift+F10)
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Strings
Structures
Enums
Names window
Graph overview
Strings window

```
; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpReserv
_DllMain@12 proc near

hinstDLL= dword ptr  4
fdwReason= dword ptr  8
lpvReserved= dword ptr  0Ch

mov     eax, [esp+fdwReason]
dec     eax
jnz     loc_1000D107
```

```
mov     eax, [esp+hinstDLL]
push    ebx
mov     ds:hModule, eax
mov     eax, off_10019044
push    esi
add     eax, 0Dh
```

100.00% (-303,-173) (34,68) 0000C42E 1000D02E: DllMain(x,x,x)

Name
F PSLIST
F nullsub_1
F nullsub_2
F StartEXS
F HandlerProc
F ServiceMain
F DllMain(x,x,x)
F InstallRT
F InstallSA
F InstallSB
F UninstallSA

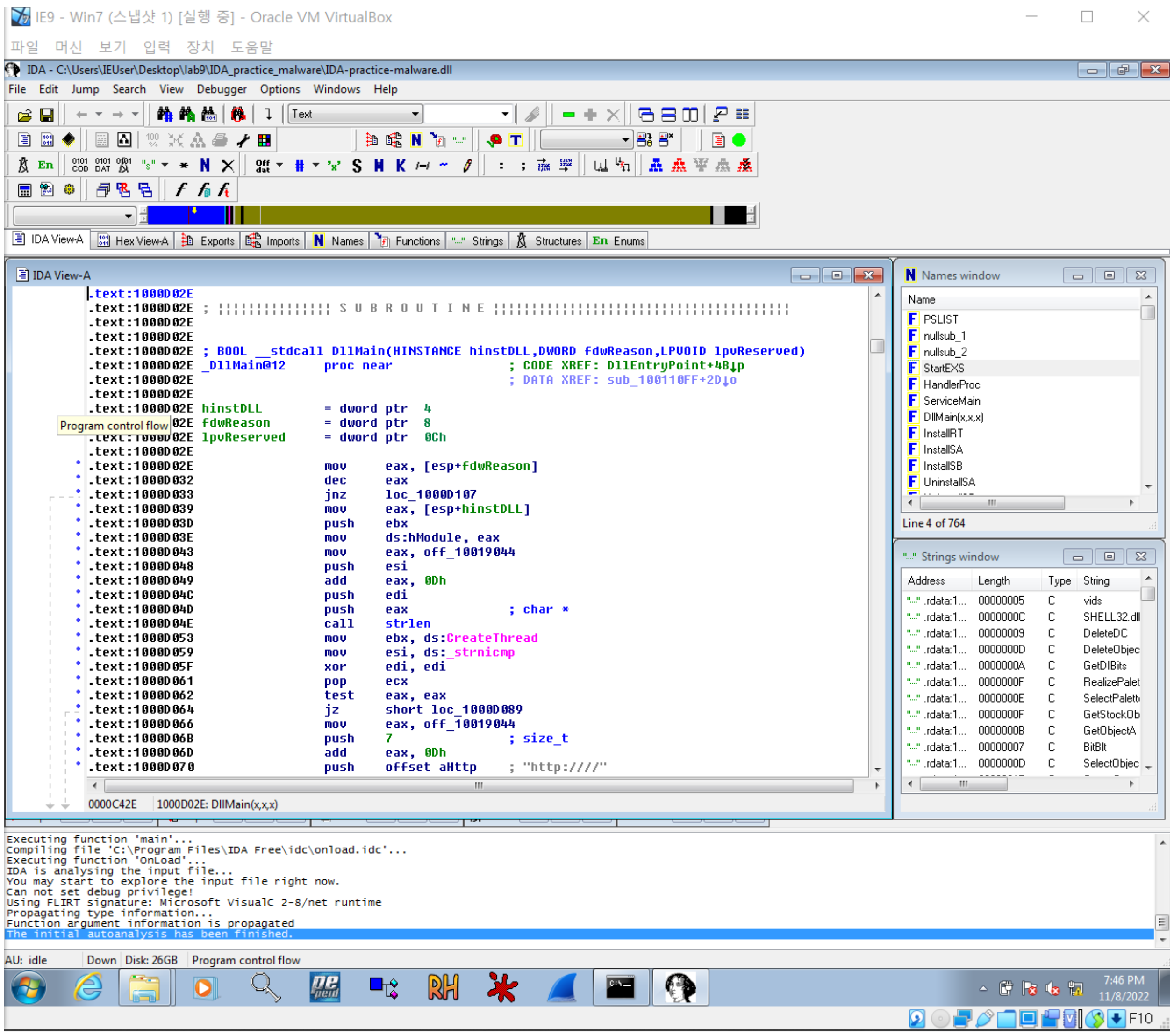
Line 4 of 764

Address	Length	Type	String
"..." .rdata:1...	00000005	C	vids
"..." .rdata:1...	0000000C	C	SHELL32.dll
	00000009	C	DeleteDC
	0000000D	C	DeleteObjec
	0000000A	C	GetDIBits
	0000000F	C	RealizePalet
	0000000E	C	SelectPalett
	0000000F	C	GetStockOb
	0000000B	C	GetObjectA
	00000007	C	BitBlt
	0000000D	C	SelectObjec

```
Executing function 'main'...
Compiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Can not set debug privilege!
Using FLIRT signature: Microsoft VisualC 2-8/net runtime
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.
```

AU: idle | Down | Disk: 26GB | Open enumerations window (Shift+F10)

7:46 PM
11/8/2022
F10


IE9 - Win7 (스냅샷 1) [실행 중] - Oracle VM VirtualBox
파일 머신 보기 입력 장치 도움말
IDA - C:\Users\IEUser\Desktop\lab9\IDA_practice_malware\IDA-practice-malware.dll
File Edit Jump Search View Debugger Options Windows Help
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Strings
Structures
Enums
Program control flow
Names window
Strings window

```
.text:1000D02E
.text:1000D02E ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
.text:1000D02E
.text:1000D02E
.text:1000D02E ; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
.text:1000D02E _DllMain@12     proc near               ; CODE XREF: DllEntryPoint+4B↓p
.text:1000D02E                                         ; DATA XREF: sub_100110FF+2D↓o
.text:1000D02E
.text:1000D02E hinstDLL        = dword ptr  4
.text:1000D02E fdwReason       = dword ptr  8
.text:1000D02E lpvReserved     = dword ptr  0Ch
.text:1000D02E
.text:1000D02E                 mov     eax, [esp+fdwReason]
.text:1000D032                 dec     eax
.text:1000D033                 jnz     loc_1000D107
.text:1000D039                 mov     eax, [esp+hinstDLL]
.text:1000D03D                 push    ebx
.text:1000D03E                 mov     ds:hModule, eax
.text:1000D043                 mov     eax, off_10019044
.text:1000D048                 push    esi
.text:1000D049                 add     eax, 0Dh
.text:1000D04C                 push    edi
.text:1000D04D                 push    eax             ; char *
.text:1000D04E                 call    strlen
.text:1000D053                 mov     ebx, ds:CreateThread
.text:1000D059                 mov     esi, ds:_strnicmp
.text:1000D05F                 xor     edi, edi
.text:1000D061                 pop     ecx
.text:1000D062                 test    eax, eax
.text:1000D064                 jz      short loc_1000D089
.text:1000D066                 mov     eax, off_10019044
.text:1000D06B                 push    7               ; size_t
.text:1000D06D                 add     eax, 0Dh
.text:1000D070                 push    offset aHttp    ; "http:////"
```

0000C42E 1000D02E: DllMain(x,x,x)

Name
PSLIST
nullsub_1
nullsub_2
StartEXS
HandlerProc
ServiceMain
DllMain(x,x,x)
InstallRT
InstallSA
InstallSB
UninstallSA

Line 4 of 764

Address	Length	Type	String
.rdata:1...	00000005	C	vids
.rdata:1...	0000000C	C	SHELL32.dll
.rdata:1...	00000009	C	DeleteDC
.rdata:1...	0000000D	C	DeleteObjec
.rdata:1...	0000000A	C	GetDIBits
.rdata:1...	0000000F	C	RealizePalet
.rdata:1...	0000000E	C	SelectPalett
.rdata:1...	0000000F	C	GetStockOb
.rdata:1...	0000000B	C	GetObjectA
.rdata:1...	00000007	C	BitBlt
.rdata:1...	0000000D	C	SelectObjec

```
Executing function 'main'...
Compiling file 'C:\Program Files\IDA Free\idc\onload.idc'...
Executing function 'OnLoad'...
IDA is analysing the input file...
You may start to explore the input file right now.
Can not set debug privilege!
Using FLIRT signature: Microsoft VisualC 2-8/net runtime
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.
```

AU: idle Down Disk: 26GB Program control flow

7:46 PM 11/8/2022

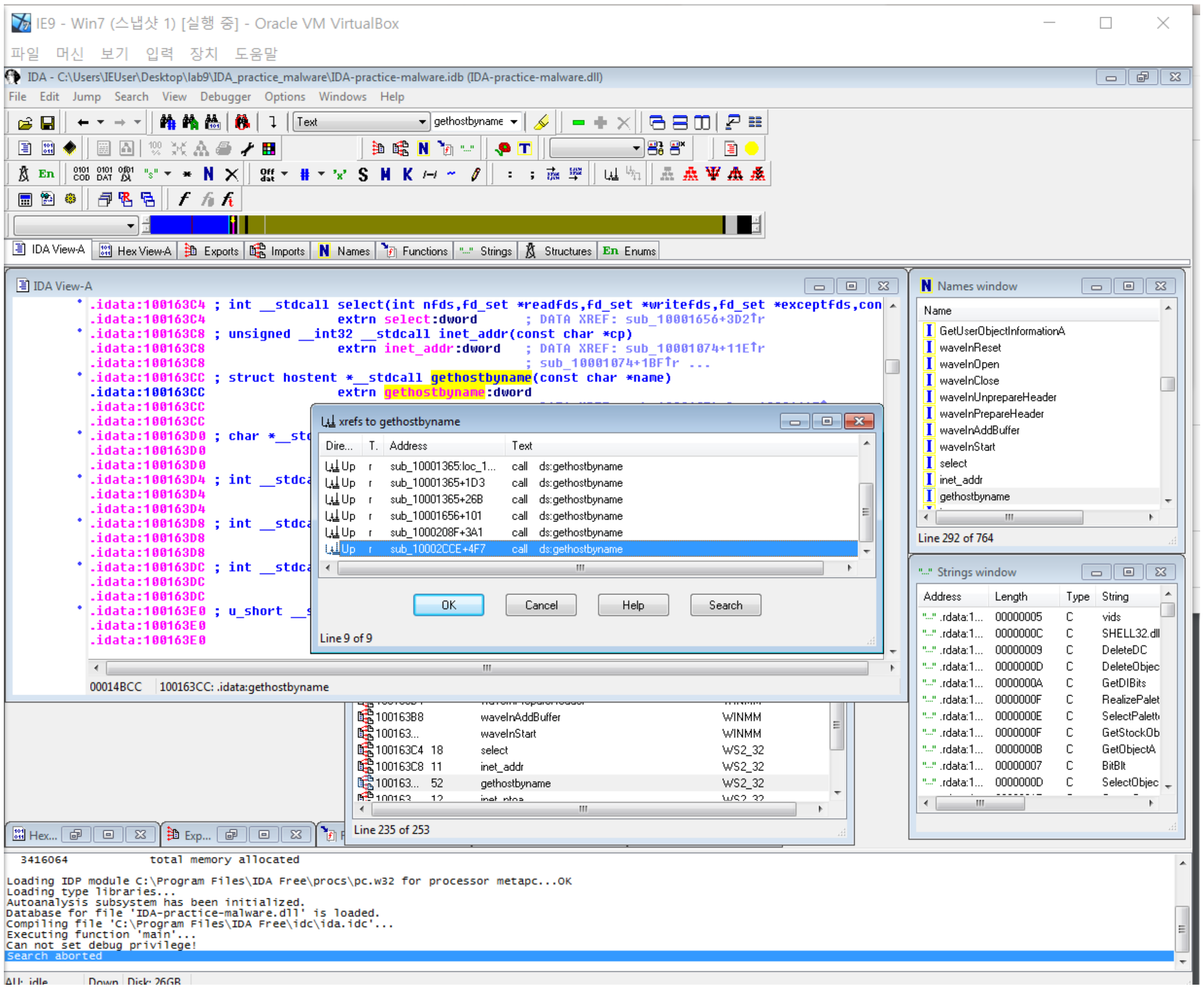
"Gethostbyname" is a Windows API function that can perform a DNS lookup. Switch to the “Imports” tab. Click the Name header to sort by name and find "gethostbyname". (Note that capital letters and lowercase letters sort into separate groups.)

Double-click gethostbyname.

The code for the function opens in Text mode. Click gethostbyname. Yellow highlights appear on both occurrences of that name. Can you determine how many times this program is being called?

The ‘gethostbyname’ is called 9 times.

How did you find your answer? If you didn’t use xrefs (by pressing ‘x’), try it now. Take a screenshot of the window.



IE9 - Win7 (스냅샷 1) [실행 중] - Oracle VM VirtualBox

IDA - C:\Users\IEUser\Desktop\lab9\IDA_practice_malware\IDA-practice-malware.idb (IDA-practice-malware.dll)

```
.idata:100163C4 ; int __stdcall select(int nfds,fd_set *readfds,fd_set *writefds,fd_set *exceptfds,con
.idata:100163C4                 extrn select:dword      ; DATA XREF: sub_10001656+3D2↑r
.idata:100163C8 ; unsigned __int32 __stdcall inet_addr(const char *cp)
.idata:100163C8                 extrn inet_addr:dword   ; DATA XREF: sub_10001074+11E↑r
.idata:100163C8                                         ; sub_10001074+1BF↑r ...
.idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC                 extrn gethostbyname:dword
.idata:100163CC
.idata:100163CC
.idata:100163D0 ; char *__st
.idata:100163D0
.idata:100163D0
.idata:100163D4 ; int __stdca
.idata:100163D4
.idata:100163D4
.idata:100163D8 ; int __stdca
.idata:100163D8
.idata:100163D8
.idata:100163DC ; int __stdca
.idata:100163DC
.idata:100163DC
.idata:100163E0 ; u_short __s
.idata:100163E0
.idata:100163E0
```

00014BCC 100163CC: .idata:gethostbyname

xrefs to gethostbyname

Dire...	T.	Address	Text
Up	r	sub_10001365:loc_1...	call ds:gethostbyname
Up	r	sub_10001365+1D3	call ds:gethostbyname
Up	r	sub_10001365+26B	call ds:gethostbyname
Up	r	sub_10001656+101	call ds:gethostbyname
Up	r	sub_1000208F+3A1	call ds:gethostbyname
Up	r	sub_10002CCE+4F7	call ds:gethostbyname

Line 9 of 9

100163B8		waveInAddBuffer	WINMM
100163...		waveInStart	WINMM
100163C4	18	select	WS2_32
100163C8	11	inet_addr	WS2_32
100163...	52	gethostbyname	WS2_32

Line 235 of 253

Names window

Name
GetUserObjectInformationA
waveInReset
waveInOpen
waveInClose
waveInUnprepareHeader
waveInPrepareHeader
waveInAddBuffer
waveInStart
select
inet_addr
gethostbyname

Line 292 of 764

Strings window

Address	Length	Type	String
.rdata:1...	00000005	C	vids
.rdata:1...	0000000C	C	SHELL32.dll
.rdata:1...	00000009	C	DeleteDC
.rdata:1...	0000000D	C	DeleteObjec
.rdata:1...	0000000A	C	GetDIBits
.rdata:1...	0000000F	C	RealizePalet
.rdata:1...	0000000E	C	SelectPalett
.rdata:1...	0000000F	C	GetStockOb
.rdata:1...	0000000B	C	GetObjectA
.rdata:1...	00000007	C	BitBlt
.rdata:1...	0000000D	C	SelectObjec

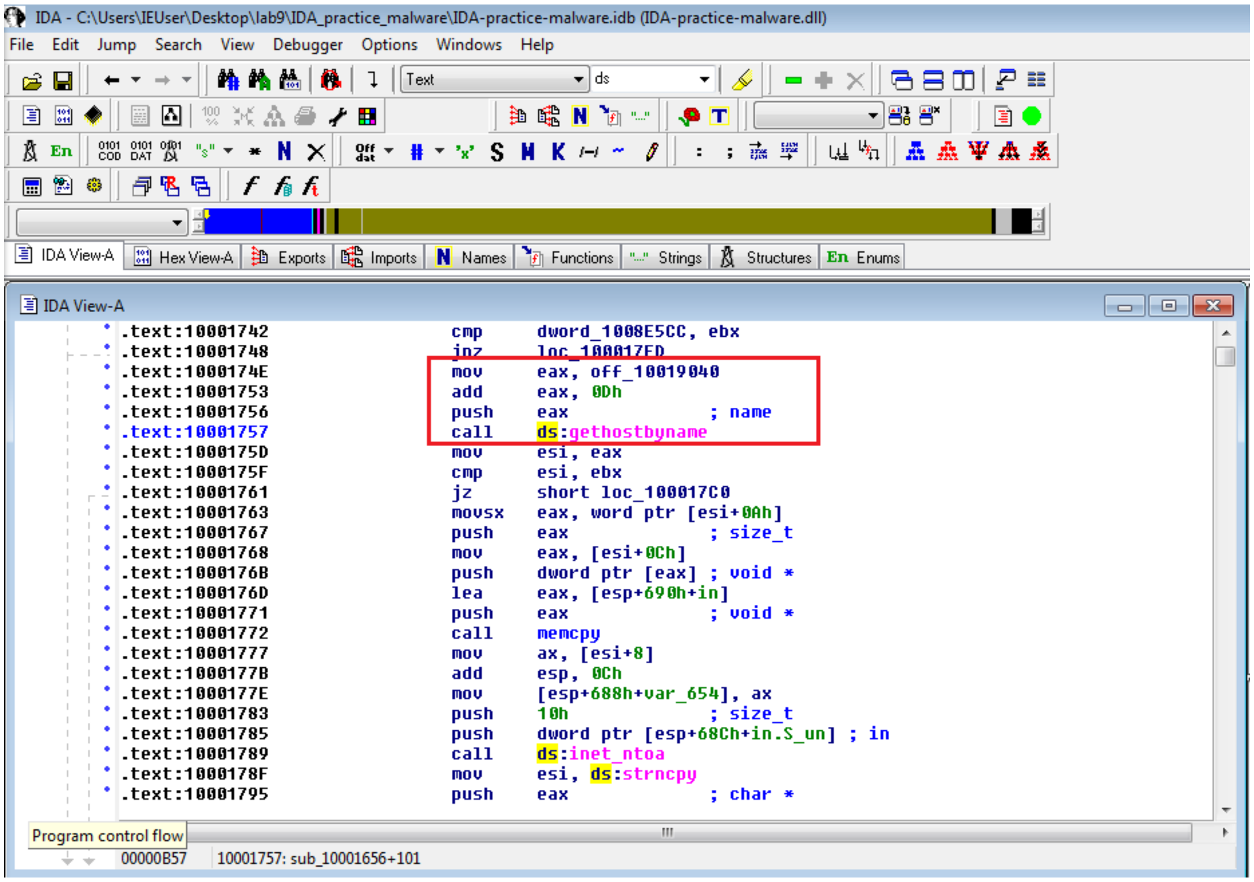
```
 3416064          total memory allocated

Loading IDP module C:\Program Files\IDA Free\procs\pc.w32 for processor metapc...OK
Loading type libraries...
Autoanalysis subsystem has been initialized.
Database for file 'IDA-practice-malware.dll' is loaded.
Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
Executing function 'main'...
Can not set debug privilege!
Search aborted
```

AU: idle   Down   Disk: 26GB

Double-click the line that has 1001656+101.

Now you are looking at a function. You can see that it loads an address named off_10019040 into register eax, adds 13 to it, pushes that address onto the stack, and calls gethostbyname.

Take a screenshot of the function and make sure that the described operation is shown in your screenshot.



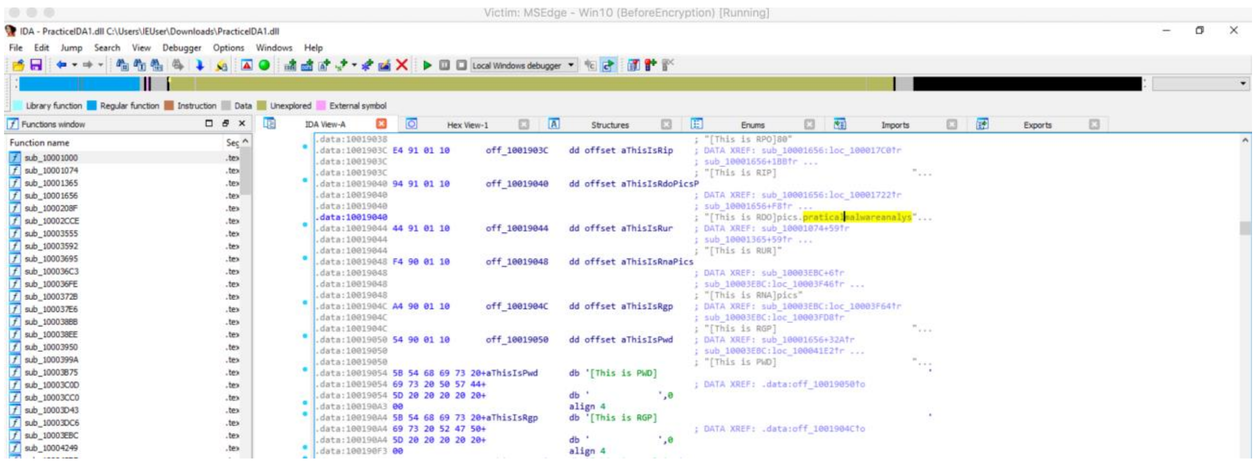
IDA - C:\Users\IEUser\Desktop\lab9\IDA_practice_malware\IDA-practice-malware.idb (IDA-practice-malware.dll)

```
.text:10001742                 cmp     dword_1008E5CC, ebx
.text:10001748                 jnz     loc_100017ED
.text:1000174E                 mov     eax, off_10019040
.text:10001753                 add     eax, 0Dh
.text:10001756                 push    eax             ; name
.text:10001757                 call    ds:gethostbyname
.text:1000175D                 mov     esi, eax
.text:1000175F                 cmp     esi, ebx
.text:10001761                 jz      short loc_100017C0
.text:10001763                 movsx   eax, word ptr [esi+0Ah]
.text:10001767                 push    eax             ; size_t
.text:10001768                 mov     eax, [esi+0Ch]
.text:1000176B                 push    dword ptr [eax] ; void *
.text:1000176D                 lea     eax, [esp+690h+in]
.text:10001771                 push    eax             ; void *
.text:10001772                 call    memcpy
.text:10001777                 mov     ax, [esi+8]
.text:1000177B                 add     esp, 0Ch
.text:1000177E                 mov     [esp+688h+var_654], ax
.text:10001783                 push    10h             ; size_t
.text:10001785                 push    dword ptr [esp+68Ch+in.S_un] ; in
.text:10001789                 call    ds:inet_ntoa
.text:1000178F                 mov     esi, ds:strncpy
.text:10001795                 push    eax             ; char *
```

00000B57 10001757: sub_10001656+101

Double-click off_10019040.

The Text view shows that this location contains a pointer to a string containing "praticalmalwareanalys", as shown below.


Victim: MSEdge - Win10 (BeforeEncryption) [Running]
IDA - PracticeIDA1.dll C:\Users\IEUser\Downloads\PracticeIDA1.dll
File Edit Jump Search View Debugger Options Windows Help
Local Windows debugger
Library function Regular function Instruction Data Unexplored External symbol
Functions window
IDA View-A
Hex View-1
Structures
Enums
Imports
Exports

Function name	Seg
sub_10001000	.tex
sub_10001074	.tex
sub_10001365	.tex
sub_10001656	.tex
sub_1000208F	.tex
sub_10002CCE	.tex
sub_10003555	.tex
sub_10003592	.tex
sub_10003695	.tex
sub_100036C3	.tex
sub_100036FE	.tex
sub_1000372B	.tex
sub_100037E6	.tex
sub_100038BB	.tex
sub_100038EE	.tex
sub_10003950	.tex
sub_1000399A	.tex
sub_10003B75	.tex
sub_10003C0D	.tex
sub_10003CC0	.tex
sub_10003D43	.tex
sub_10003DC6	.tex
sub_10003EBC	.tex
sub_10004249	.tex

```
.data:10019038                                                   ; "[This is RPO]80"
.data:1001903C E4 91 01 10      off_1001903C    dd offset aThisIsRip    ; DATA XREF: sub_10001656:loc_100017C0↑r
.data:1001903C                                                   ; sub_10001656+1B8↑r ...
.data:1001903C                                                   ; "[This is RIP]                    "...
.data:10019040 94 91 01 10      off_10019040    dd offset aThisIsRdoPicsP
.data:10019040                                                   ; DATA XREF: sub_10001656:loc_10001722↑r
.data:10019040                                                   ; sub_10001656+F8↑r ...
.data:10019040                                                   ; "[This is RDO]pics.praticalmalwareanalys"...
.data:10019044 44 91 01 10      off_10019044    dd offset aThisIsRur    ; DATA XREF: sub_10001074+59↑r
.data:10019044                                                   ; sub_10001365+59↑r ...
.data:10019044                                                   ; "[This is RUR]"
.data:10019048 F4 90 01 10      off_10019048    dd offset aThisIsRnaPics
.data:10019048                                                   ; DATA XREF: sub_10003EBC+6↑r
.data:10019048                                                   ; sub_10003EBC:loc_10003F46↑r ...
.data:10019048                                                   ; "[This is RNA]pics"
.data:1001904C A4 90 01 10      off_1001904C    dd offset aThisIsRgp    ; DATA XREF: sub_10003EBC:loc_10003F64↑r
.data:1001904C                                                   ; sub_10003EBC:loc_10003FD8↑r
.data:1001904C                                                   ; "[This is RGP]                    "...
.data:10019050 54 90 01 10      off_10019050    dd offset aThisIsPwd    ; DATA XREF: sub_10001656+32A↑r
.data:10019050                                                   ; sub_10003EBC:loc_100041E2↑r ...
.data:10019050                                                   ; "[This is PWD]                    "...
.data:10019054 5B 54 68 69 73 20+aThisIsPwd     db '[This is PWD]
.data:10019054 69 73 20 50 57 44+                                ; DATA XREF: .data:off_10019050↑o
.data:10019054 5D 20 20 20 20 20+               db '                 ',0
.data:100190A3 00                               align 4
.data:100190A4 5B 54 68 69 73 20+aThisIsRgp     db '[This is RGP]
.data:100190A4 69 73 20 52 47 50+                                ; DATA XREF: .data:off_1001904C↑o
.data:100190A4 5D 20 20 20 20 20+               db '                 ',0
.data:100190F3 00                               align 4
```

Choose “praticalmalwareanalys” string, and click the "Hex View-A" tab.

The four bytes starting at 10019040 contain a 32-bit address in little-endian order.

That address is 10019194. Find this address in the presented view, and report the domain name you found:

I found ‘pics.practicalmalwareanalysis.com’ at address ‘10019194’

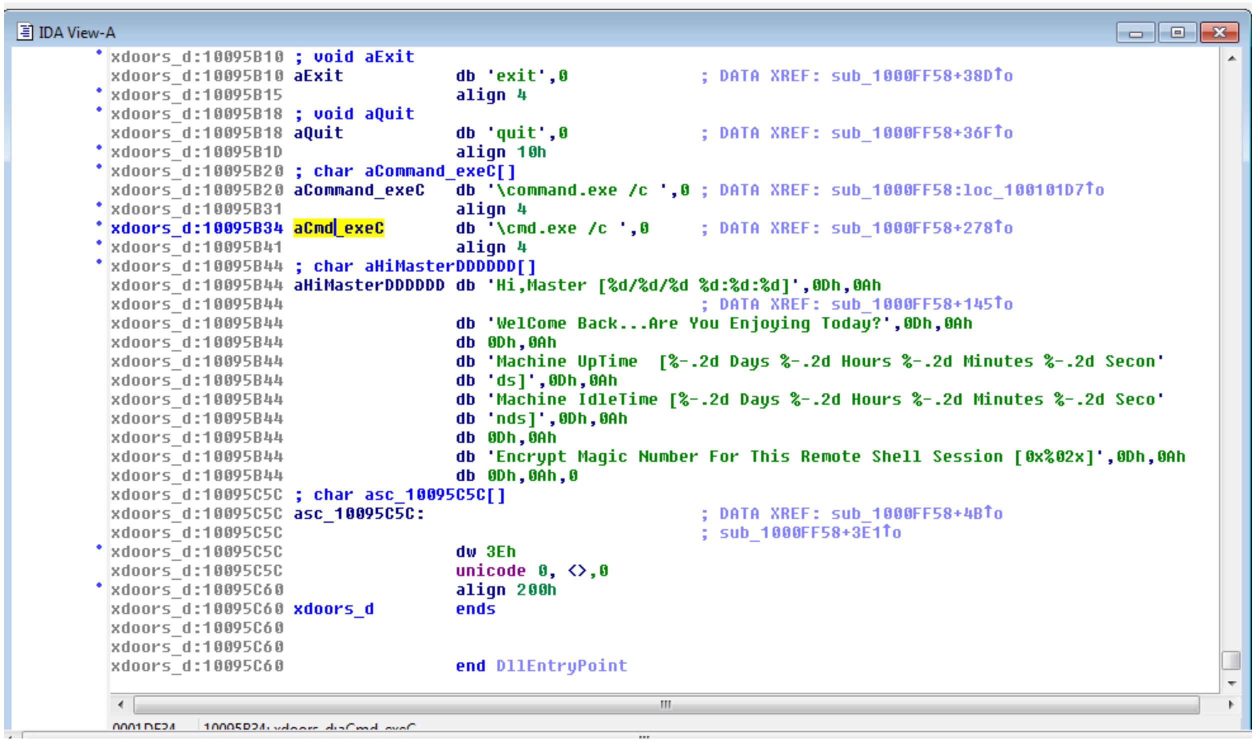
This is the domain that will be resolved by calling gethostbyname. Do you understand why?

Open the Strings tab (View > Open subviews > Strings). Click the gray String column header to sort the data.

Scroll down about 3/4 of the way, and find the String "\\cmd.exe /c",

Double-click "\\cmd.exe /c", and now move to “IDA View-A" tab.

Take a screenshot of what you see.



IDA View-A

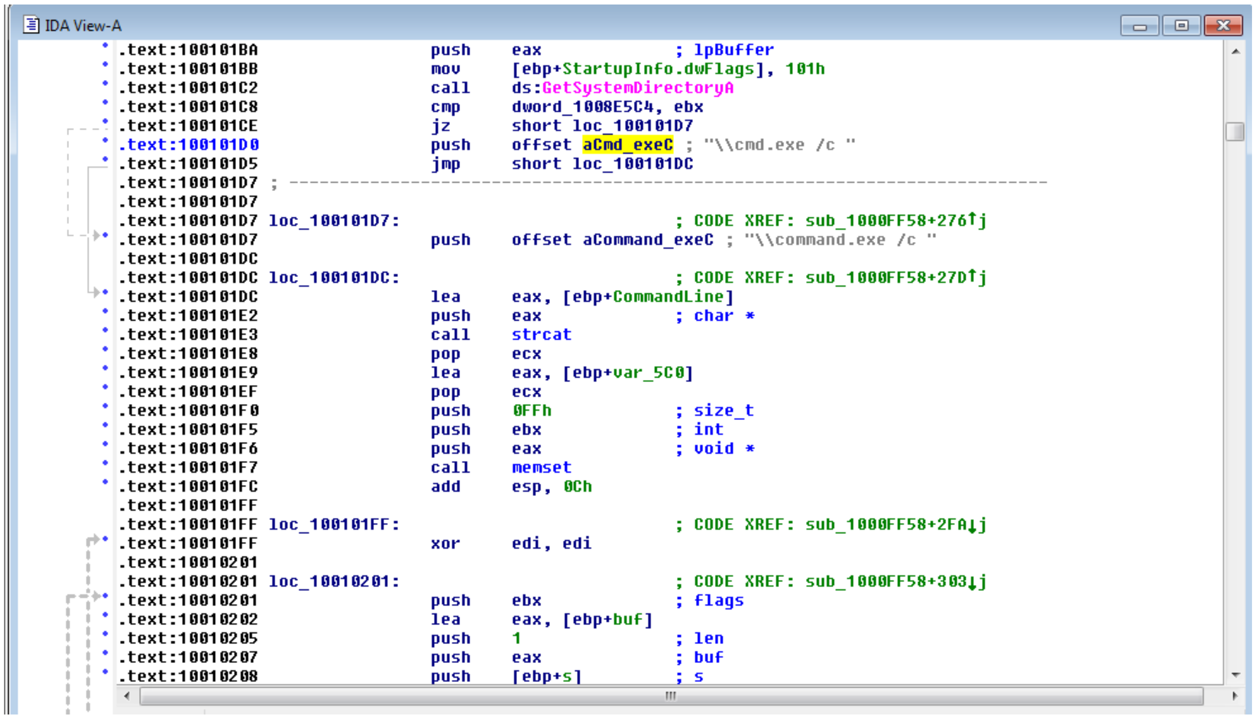
```
xdoors_d:10095B10 ; void aExit
xdoors_d:10095B10 aExit           db 'exit',0             ; DATA XREF: sub_1000FF58+38D↑o
xdoors_d:10095B15                 align 4
xdoors_d:10095B18 ; void aQuit
xdoors_d:10095B18 aQuit           db 'quit',0             ; DATA XREF: sub_1000FF58+36F↑o
xdoors_d:10095B1D                 align 10h
xdoors_d:10095B20 ; char aCommand_exeC[]
xdoors_d:10095B20 aCommand_exeC   db '\command.exe /c ',0 ; DATA XREF: sub_1000FF58:loc_100101D7↑o
xdoors_d:10095B31                 align 4
xdoors_d:10095B34 aCmd_exeC       db '\cmd.exe /c ',0     ; DATA XREF: sub_1000FF58+278↑o
xdoors_d:10095B41                 align 4
xdoors_d:10095B44 ; char aHiMasterDDDDDD[]
xdoors_d:10095B44 aHiMasterDDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
xdoors_d:10095B44                                         ; DATA XREF: sub_1000FF58+145↑o
xdoors_d:10095B44                 db 'WelCome Back...Are You Enjoying Today?',0Dh,0Ah
xdoors_d:10095B44                 db 0Dh,0Ah
xdoors_d:10095B44                 db 'Machine UpTime  [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Secon'
xdoors_d:10095B44                 db 'ds]',0Dh,0Ah
xdoors_d:10095B44                 db 'Machine IdleTime [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Seco'
xdoors_d:10095B44                 db 'nds]',0Dh,0Ah
xdoors_d:10095B44                 db 0Dh,0Ah
xdoors_d:10095B44                 db 'Encrypt Magic Number For This Remote Shell Session [0x%02x]',0Dh,0Ah
xdoors_d:10095B44                 db 0Dh,0Ah,0
xdoors_d:10095C5C ; char asc_10095C5C[]
xdoors_d:10095C5C asc_10095C5C:                           ; DATA XREF: sub_1000FF58+4B↑o
xdoors_d:10095C5C                                         ; sub_1000FF58+3E1↑o
xdoors_d:10095C5C                 dw 3Eh
xdoors_d:10095C5C                 unicode 0, <>,0
xdoors_d:10095C60                 align 200h
xdoors_d:10095C60 xdoors_d        ends
xdoors_d:10095C60
xdoors_d:10095C60
xdoors_d:10095C60                 end DllEntryPoint
```

The string appears in text mode. Click in the word cmd so it's highlighted and press x. A "xrefs to aCmd_exeC" box appears.

In the "xrefs to aCmd_exeC" box, double-click sub_1000FF58+278.

You see the code that uses this string. There are two boxes of code, one that starts a string with "cmd.exe -c" and the other that starts it with "command.exe /c".

Take a screenshot of what you see.



```
.text:100101BA                 push    eax             ; lpBuffer
.text:100101BB                 mov     [ebp+StartupInfo.dwFlags], 101h
.text:100101C2                 call    ds:GetSystemDirectoryA
.text:100101C8                 cmp     dword_1008E5C4, ebx
.text:100101CE                 jz      short loc_100101D7
.text:100101D0                 push    offset aCmd_exeC ; "\\cmd.exe /c "
.text:100101D5                 jmp     short loc_100101DC
.text:100101D7 ; ---------------------------------------------------------------------------
.text:100101D7
.text:100101D7 loc_100101D7:                           ; CODE XREF: sub_1000FF58+276↑j
.text:100101D7                 push    offset aCommand_exeC ; "\\command.exe /c "
.text:100101DC
.text:100101DC loc_100101DC:                           ; CODE XREF: sub_1000FF58+27D↑j
.text:100101DC                 lea     eax, [ebp+CommandLine]
.text:100101E2                 push    eax             ; char *
.text:100101E3                 call    strcat
.text:100101E8                 pop     ecx
.text:100101E9                 lea     eax, [ebp+var_5C0]
.text:100101EF                 pop     ecx
.text:100101F0                 push    0FFh            ; size_t
.text:100101F5                 push    ebx             ; int
.text:100101F6                 push    eax             ; void *
.text:100101F7                 call    memset
.text:100101FC                 add     esp, 0Ch
.text:100101FF
.text:100101FF loc_100101FF:                           ; CODE XREF: sub_1000FF58+2FA↓j
.text:100101FF                 xor     edi, edi
.text:10010201
.text:10010201 loc_10010201:                           ; CODE XREF: sub_1000FF58+303↓j
.text:10010201                 push    ebx             ; flags
.text:10010202                 lea     eax, [ebp+buf]
.text:10010205                 push    1               ; len
.text:10010207                 push    eax             ; buf
.text:10010208                 push    [ebp+s]         ; s
```

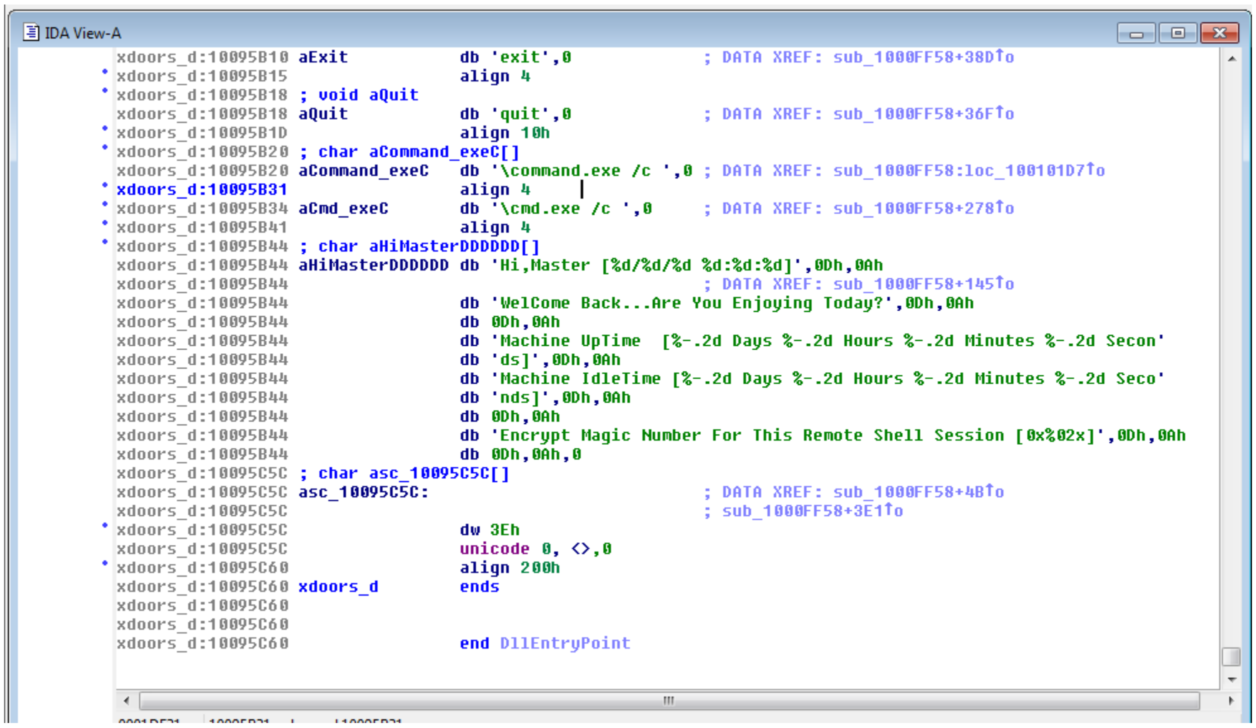
This looks like a remote shell, executing commands from the botmaster for either a 32-bit or 16-bit system.

Drag the code boxes down to see the module containing "Hi, Master".

Hover the mouse over aHiMasterDDDDDD to see more of the referenced strings.

This looks like a message the bot sends to the botmaster, further confirming that this is a RAT (Remote Administration Tool / Remote Access Trojan).

Take a screenshot of what you see.


IDA View-A

```
xdoors_d:10095B10 aExit           db 'exit',0             ; DATA XREF: sub_1000FF58+38D↑o
xdoors_d:10095B15                 align 4
xdoors_d:10095B18 ; void aQuit
xdoors_d:10095B18 aQuit           db 'quit',0             ; DATA XREF: sub_1000FF58+36F↑o
xdoors_d:10095B1D                 align 10h
xdoors_d:10095B20 ; char aCommand_exeC[]
xdoors_d:10095B20 aCommand_exeC   db '\command.exe /c ',0 ; DATA XREF: sub_1000FF58:loc_100101D7↑o
xdoors_d:10095B31                 align 4
xdoors_d:10095B34 aCmd_exeC       db '\cmd.exe /c ',0     ; DATA XREF: sub_1000FF58+278↑o
xdoors_d:10095B41                 align 4
xdoors_d:10095B44 ; char aHiMasterDDDDDD[]
xdoors_d:10095B44 aHiMasterDDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
xdoors_d:10095B44                                         ; DATA XREF: sub_1000FF58+145↑o
xdoors_d:10095B44                 db 'WelCome Back...Are You Enjoying Today?',0Dh,0Ah
xdoors_d:10095B44                 db 0Dh,0Ah
xdoors_d:10095B44                 db 'Machine UpTime  [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Secon'
xdoors_d:10095B44                 db 'ds]',0Dh,0Ah
xdoors_d:10095B44                 db 'Machine IdleTime [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Seco'
xdoors_d:10095B44                 db 'nds]',0Dh,0Ah
xdoors_d:10095B44                 db 0Dh,0Ah
xdoors_d:10095B44                 db 'Encrypt Magic Number For This Remote Shell Session [0x%02x]',0Dh,0Ah
xdoors_d:10095B44                 db 0Dh,0Ah,0
xdoors_d:10095C5C ; char asc_10095C5C[]
xdoors_d:10095C5C asc_10095C5C:                           ; DATA XREF: sub_1000FF58+4B↑o
xdoors_d:10095C5C                                         ; sub_1000FF58+3E1↑o
xdoors_d:10095C5C                 dw 3Eh
xdoors_d:10095C5C                 unicode 0, <>,0
xdoors_d:10095C60                 align 200h
xdoors_d:10095C60 xdoors_d        ends
xdoors_d:10095C60
xdoors_d:10095C60
xdoors_d:10095C60                 end DllEntryPoint
```

Part 3 - Check your understanding

The .dll you analyzed in Part 2 is Lab05-01.dll provided by the book. Go over the Lab questions on page 107, and try to answer them. The book provides answers, so you can verify you can check your answers as needed. This part is optional, and no submission is required.