Lab 5: IDA Pro

What you need:

- A Windows machine, real or virtual, such as the Windows 2008 Server VM we've been using
- The textbook: "Practical Malware Analysis"

Purpose

You will practice using IDA Pro.

Installing IDA Pro Free

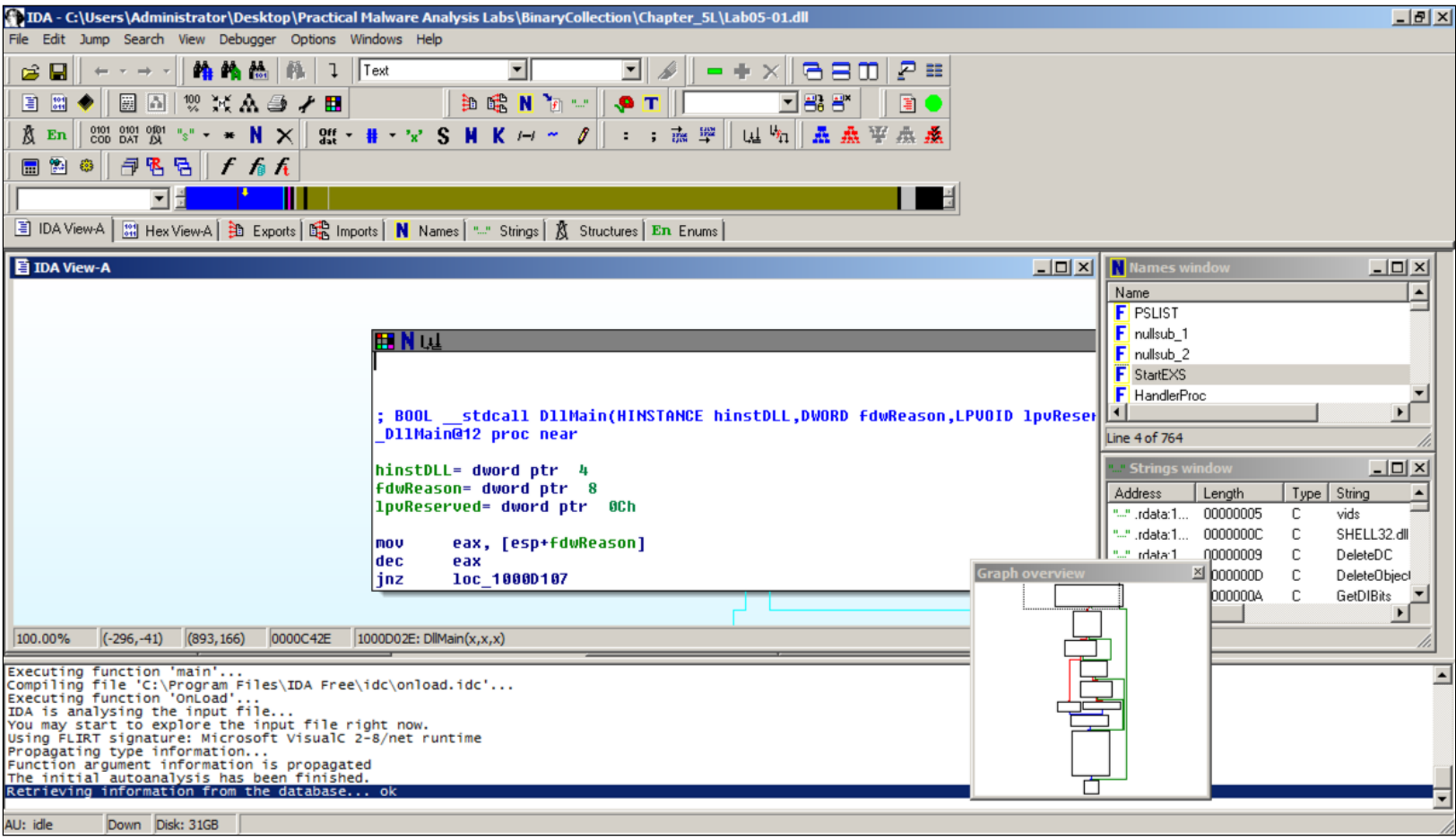
If you don't have it, download IDA Pro Free here:

https://www.hex-rays.com/products/ida/support/download_freeware.shtml Install the Windows version with the default options.

Opening Lab05-01.dll in IDA Pro

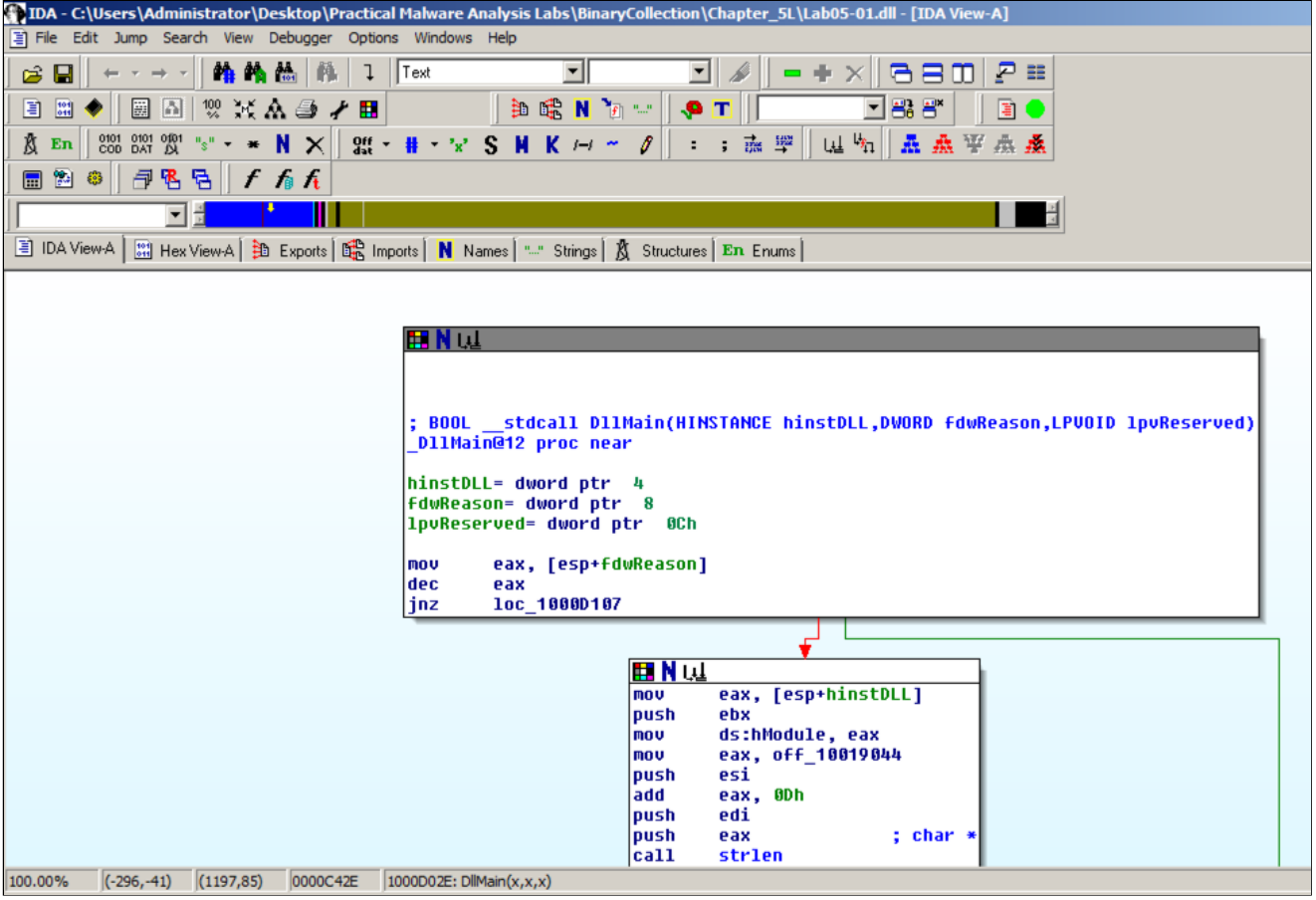
Launch IDA Pro Free. Click **OK**. Click **New**. Click the "**PE Dynamic Library**" icon and click **OK**. Navigate to Lab05-01.dll and open it.

In the "Welcome to the PE Dynamic Library file loading Wizard" box, click **Next**, **Next**, **Finish**. IDA opens the file, as shown below:



Adjusting Graph Mode Options

The screen is cluttered. Close the little "Graph overview" box and maximize the "IDA View-A" window. Drag the message area to the bottom of the screen to minimize it, as shown below:

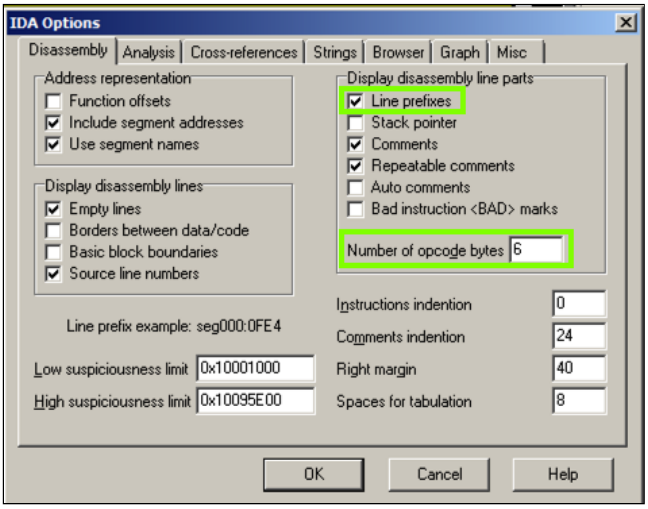


The code doesn't show line numbers or hexadecimal instructions. To fix that, click **Options**, **General**.

In the IDA Options box, in the top right, in the '"Display disassembly line parts" pane, make these changes, as shown below:

- Check "**Line prefixes**"
- Change "**Number of opcode bytes**" to **6**

Click **OK**.



The "graph mode" display is more informative now, as shown below.

```
1000D02E
1000D02E
1000D02E
1000D02E                ; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
1000D02E                _DllMain@12 proc near
1000D02E
1000D02E                hinstDLL= dword ptr  4
1000D02E                fdwReason= dword ptr  8
1000D02E                lpvReserved= dword ptr  0Ch
1000D02E
1000D02E 8B 44 24 08    mov     eax, [esp+fdwReason]
1000D032 48             dec     eax
1000D033 0F 85 CE 00 00 00 jnz  loc_1000D107
```

```
1000D039 8B 44 24 04    mov     eax, [esp+hinstDLL]
1000D03D 53             push    ebx
1000D03E A3 00 30 09 10 mov     ds:hModule, eax
1000D043 A1 44 90 01 10 mov     eax, off_10019044
1000D048 56             push    esi
1000D049 83 C0 0D       add     eax, 0Dh
1000D04C 57             push    edi
1000D04D 50             push    eax             ; char *
1000D04E E8 F9 7E 00 00 call    strlen
1000D053 8B 1D 08 62 01 10 mov  ebx, ds:CreateThread
1000D059 8B 35 C0 62 01 10 mov  esi, ds:_strnicmp
```

Text Mode

Click in the Graph Mode window and press the **SPACEBAR**.

IDA shows the assembly code in a text-only view, as shown below.

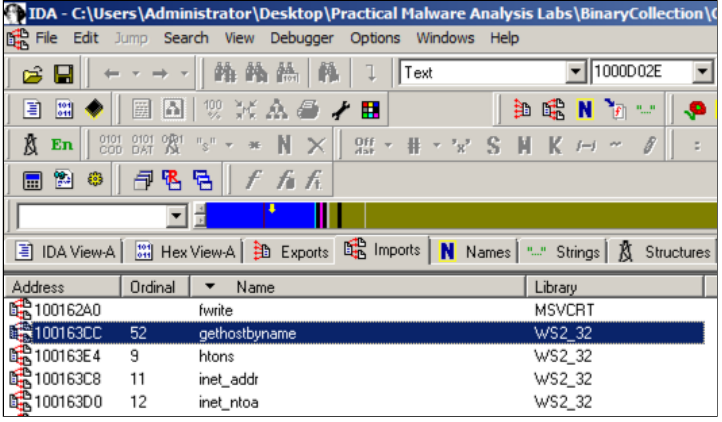
```
.text:1000D02E
.text:1000D02E hinstDLL        = dword ptr  4
.text:1000D02E fdwReason       = dword ptr  8
.text:1000D02E lpvReserved     = dword ptr  0Ch
.text:1000D02E
.text:1000D02E                 mov     eax, [esp+fdwReason]
.text:1000D032                 dec     eax
.text:1000D033                 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:////"
.text:1000D075                 push    eax             ; char *
```

Press the **SPACEBAR** again to return to Graph Mode.

Finding the Import for gethostbyname

"Gethostbyname" is a Windows API function that can preform a DNS lookup.

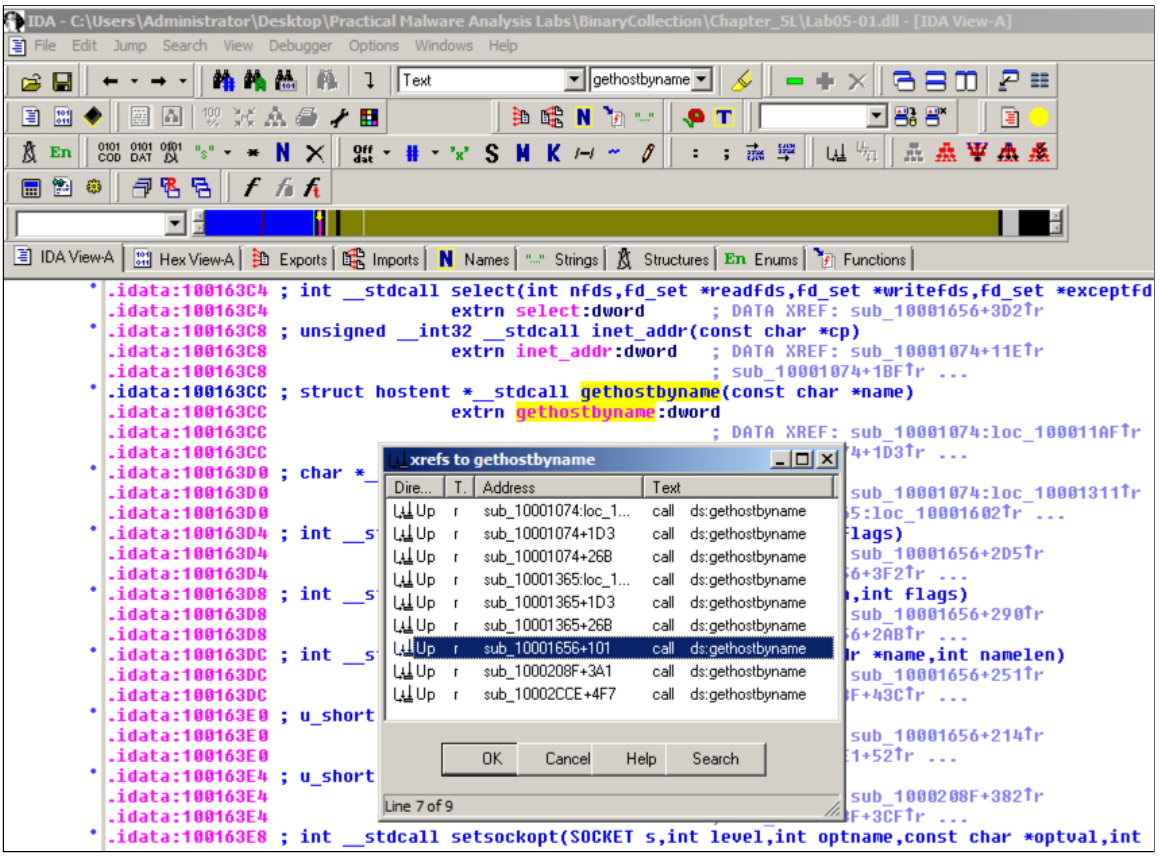
In IDA Pro, click **View**, "**Open subview**", **Imports**. Click the **Name** header to sort by name. Find "gethostbyname", as shown below. (Note that capital letters and lowercase letters sort into separate groups.)



Double-click **gethostbyname**.

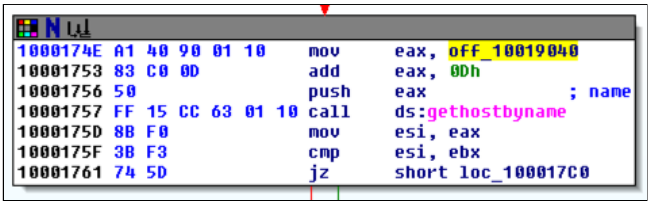
The code for the function opens in Text mode, as shown below.

Click **gethostbyname**. Yellow highlights appear on both occurrences of that name, as shown below. Press **Ctrl+x** to open the "xrefs to gethostbyname" box shown below.



There are nine calls to gethostbyname in five different functions. Double-click the third one from the bottom, at an address of **1001656+101**, as highlighted in the image above.

The function appears, as shown below. It loads an address named **off_10019040** into register eax, adds 13 to it (0d in hexadecimal), pushes that address onto the stack, and calls gethostbyname.



Double-click **off_10019040**.

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

```
.data:10019038                                          ; sub_10001656+182↑r ...
.data:10019038                                          ; "[This is RPO]80"
.data:1001903C off_1001903C     dd offset aThisIsRip    ; DATA XREF: sub_10001656:loc_100017C0↑r
.data:1001903C                                          ; sub_10001656+1BB↑r ...
.data:1001903C                                          ; "[This is RIP]                          "...
.data:10019040 off_10019040     dd offset aThisIsRdoPics_
.data:10019040                                          ; DATA XREF: sub_10001656:loc_10001722↑r
.data:10019040                                          ; sub_10001656+F8↑r ...
.data:10019040                                          ; "[This is RDO][redacted].praticalmalwareanalys"...
.data:10019044 off_10019044     dd offset aThisIsRur    ; DATA XREF: sub_10001074+59↑r
.data:10019044                                          ; sub_10001365+59↑r ...
```

Flag PMA 303.1: Domain Name (10 pts)

For a clearer view, click the "**Hex View-A**" tab.

The four bytes starting at 10019040 contain a 32-bit address in little-endian order, as highlighted in blue in the figure below. That address is 10019194. There's a series of ASCII values at that address, outlined in green in the figure below. Skipping the first 13 bytes leaves a string ending in

.praticalpalwareanalysis.com

as shown below. This is the domain that will be resolved by calling gethostbyname.

The flag is covered by a green box in the image below.

```
.data:10019010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019020  AC 92 01 10 98 92 01 10  84 92 01 10 70 92 01 10  ¼Æ ┼ÿÆ ┼äÆ ┼pÆ
.data:10019030  5C 92 01 10 48 92 01 10  34 92 01 10 E4 91 01 10  \Æ ┼HÆ ┼4Æ ┼Sæ
.data:10019040  94 91 01 10 44 91 01 10  F4 90 01 10 A4 90 01 10  öæ ┼Dæ ┼(É ┼ñÉ
.data:10019050  54 90 01 10 5B 54 68 69  73 20 69 73 20 50 57 44  TÉ ┼[This is PWD
.data:10019060  5D 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  ]
.data:10019070  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
.data:10019080  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
.data:10019090  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
.data:100190A0  20 20 00 00 5B 54 68 69  73 20 69 73 20 52 47 50    ..[This is RGP
.data:100190B0  5D 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  ]
.data:100190C0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
.data:100190D0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
.data:100190E0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
.data:100190F0  20 20 00 00 5B 54 68 69  73 20 69 73 20 52 4E 41    ..[This is RNA
.data:10019100  5D 70 69 63 73 00 00 00  00 00 00 00 00 00 00 00  ]pics...........
.data:10019110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019140  00 20 00 00 5B 54 68 69  73 20 69 73 20 52 55 52  . ..[This is RUR
.data:10019150  5D 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ]...............
.data:10019160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
.data:10019190  00 20 00 00 5B 54 68 69  73 20 69 73 20 52 44 4F  . ..[This is RDO
.data:100191A0  5D [redacted] 2E 70 72  61 74 69 63 61 6C 6D 61  ][redacted].praticalma
.data:100191B0  6C 77 61 72 65 61 6E 61  6C 79 73 69 73 2E 63 6F  lwareanalysis.co
.data:100191C0  6D 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  m...............
```

Examinining the Code that References "\cmd.exe /c"

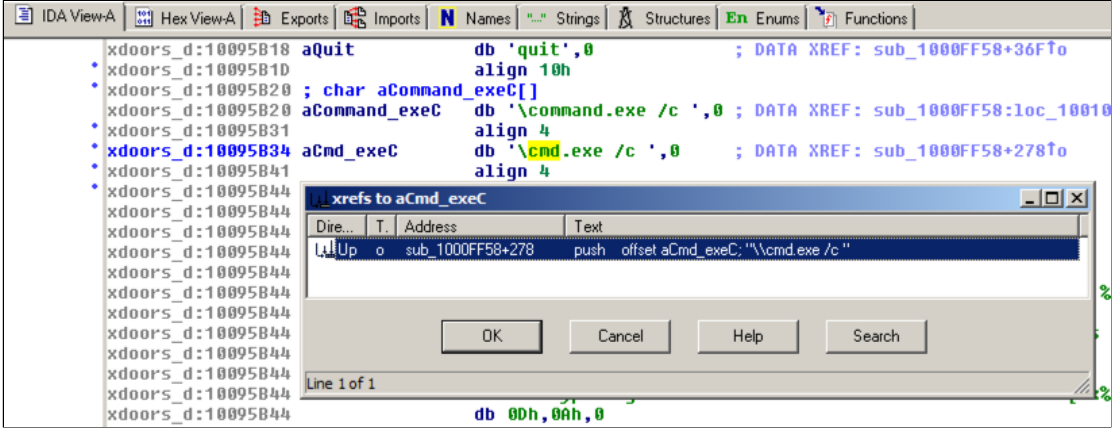
In IDA Pro, click the **Strings** tab. Click the gray **String** column header to sort the data.

Scroll down about 3/4 of the way, and find the String "**\\cmd.exe /c**", as highlighted in the image below.

Address	Length	Type	String
.data:100190F4	00000012	C	[This is RNA]pics
.data:10019234	00000010	C	[This is RPO]80
.data:10019144	0000000E	C	[This is RUR]
.data:1001925C	00000014	C	[This is SS2]
.data:10019270	00000014	C	[This is SSD]
xdoors_d:100939A0	0000000F	C	\\Device\\Video0
xdoors_d:100954B0	0000000C	C	\\Parameters
xdoors_d:10095B34	0000000D	C	\\cmd.exe /c
xdoors_d:10095B20	00000011	C	\\command.exe /c
xdoors_d:10093844	0000000B	C	\n\n\n[%s %s]
xdoors_d:100943C4	0000000F	C	\r\n%-16d%-20s%d
xdoors_d:10093D50	00000023	C	\r\n(1) Enter Current Directory '%s'
xdoors_d:10093A98	00000034	C	\r\n(1) Enter Current Directory Error Update Failed\r\n

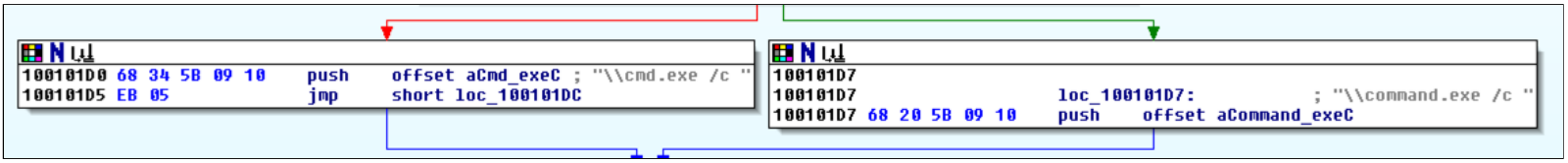
Double-click "**\\cmd.exe /c**". Click the "b>IDA View-A" tab.

The string appears in text mode, as shown below. Click in the word **cmd** so it's highlighted and press **Ctrl+x**. A "xrefs to aCmd_exeC" box appears, as shown below.



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



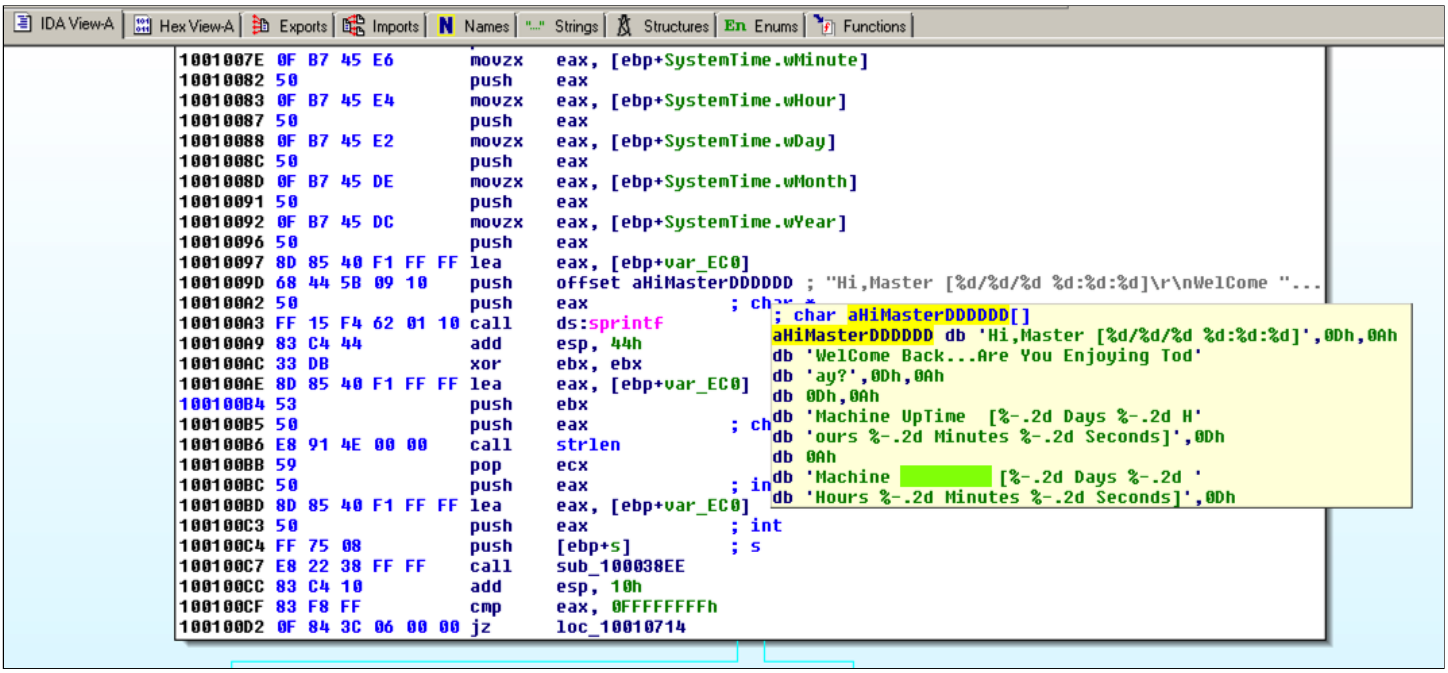
Flag PMA 303.2: Message (5 pts)

Drag the code boxes down to see the module containing "**Hi, Master**", as shown below.

Hover the mouse over **aHiMasterDDDDDD** to see more of the referenced strings, as shown below.

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

The flag is covered by a green box in the image below.



Challenges with IDA

Downloading the Files to Examine

If you are using the VM handed out by your instructor, the files you need are already on the disk in the **C:\IDA** folder.

Otherwise, download these files into the C:\IDA folder.

- crackme-121-1.exe
- crackme-121-1.exe
- crackme-121-1.exe
- crackme-121-1.exe
- msvcr100d.dll

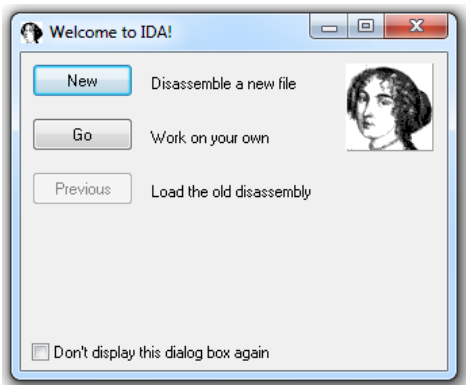
Launching IDA Pro Free

Start IDA Pro Free.

When you see the IDA window shown below, click the **OK** button.

Click "**I Agree**".

In the "Welcome to IDA!" box, as shown below, click the **New** button.

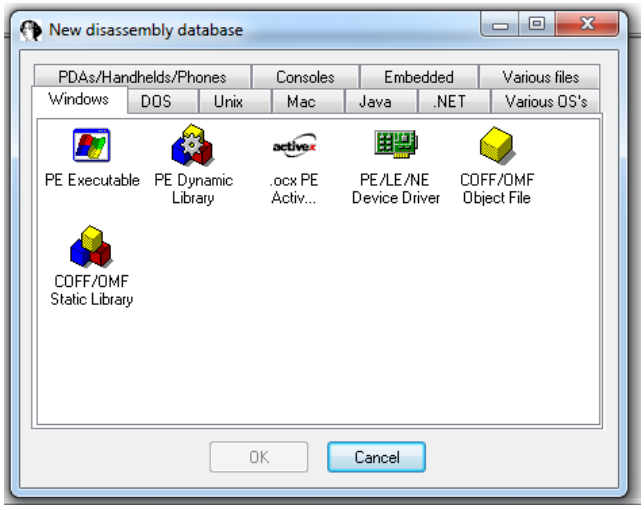


In the "About" box, click the **OK** button.

Loading the EXE File

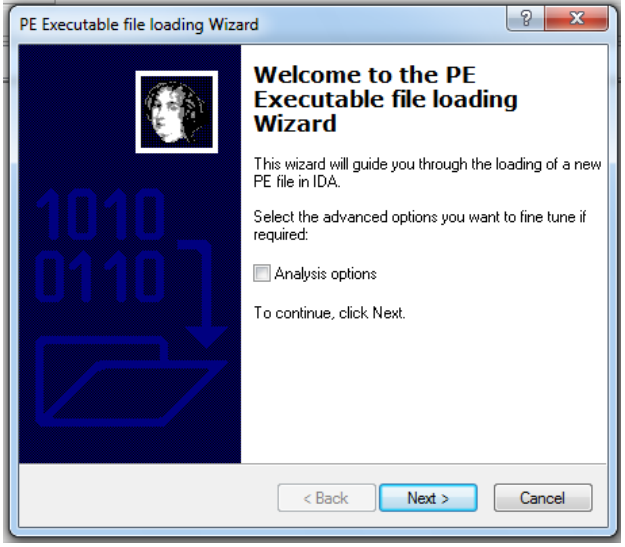
In the "Welcome to IDA" box, click the **New** button.

In the "New disassembly database" box, click "**PE Executable**", and then click **OK**, as shown below:



In the "Select PE Executable to disassemble" box, navigate to **C:\IDA\crackme-121-1.exe** and double-click it.

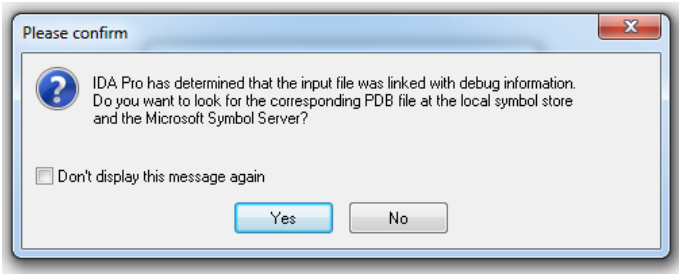
In the "Welcome to the PE Executable file loading Wizard" box, click the **Next** button, as shown below:



In the "Segment Creation" box, click **Next**.

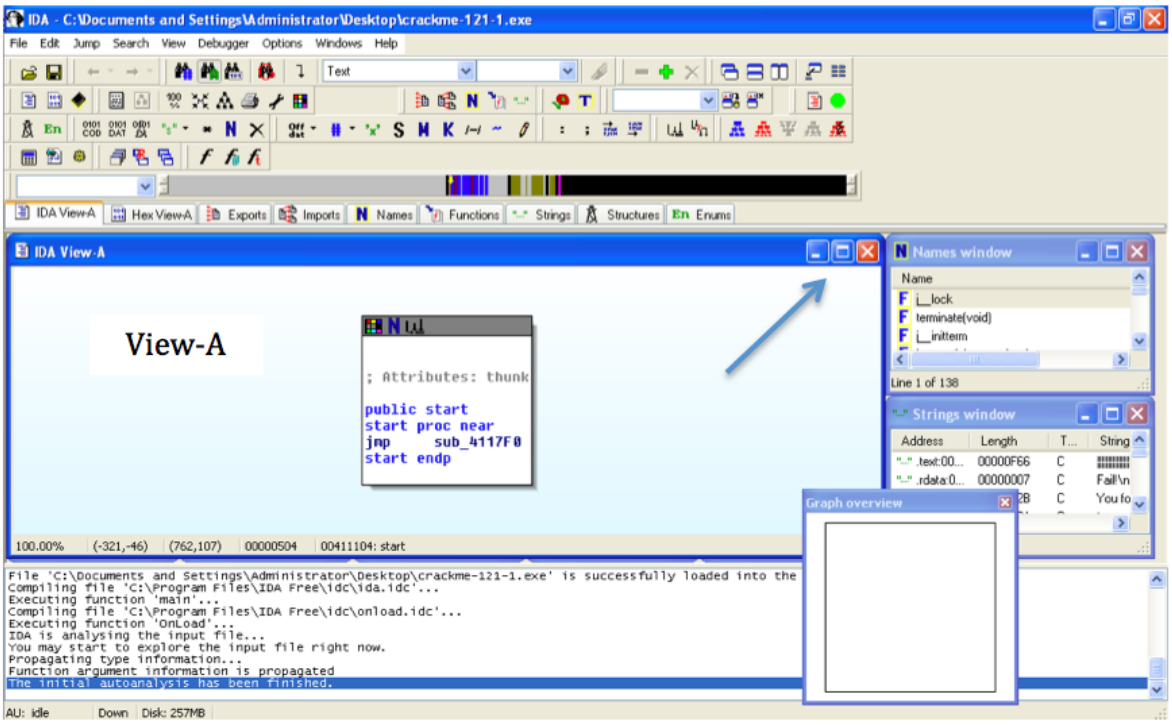
In the "File loading" box, click **Finish**.

A box pops up saying "...the input file was linked with debug information...", as shown below. Click the **Yes** button.



Viewing Disassembled Code

In IDA Pro, find the "View-A" pane, which shows boxes containing code linked to other boxes in a flowchart style. Maximize this pane, by clicking the button indicated by the arrow in the figure below:



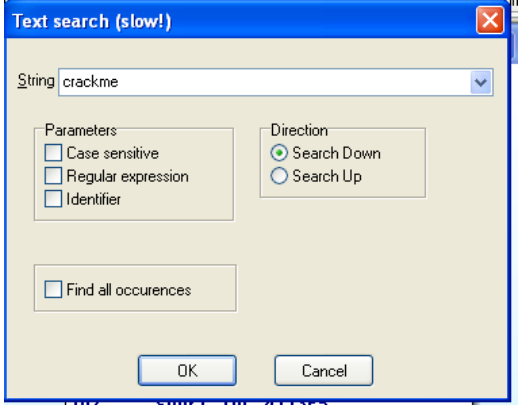
Close the "Graph Overview" box in the lower right corner.

Drag the lower border of the "View-A" pane down, to make as large a viewable area as possible.

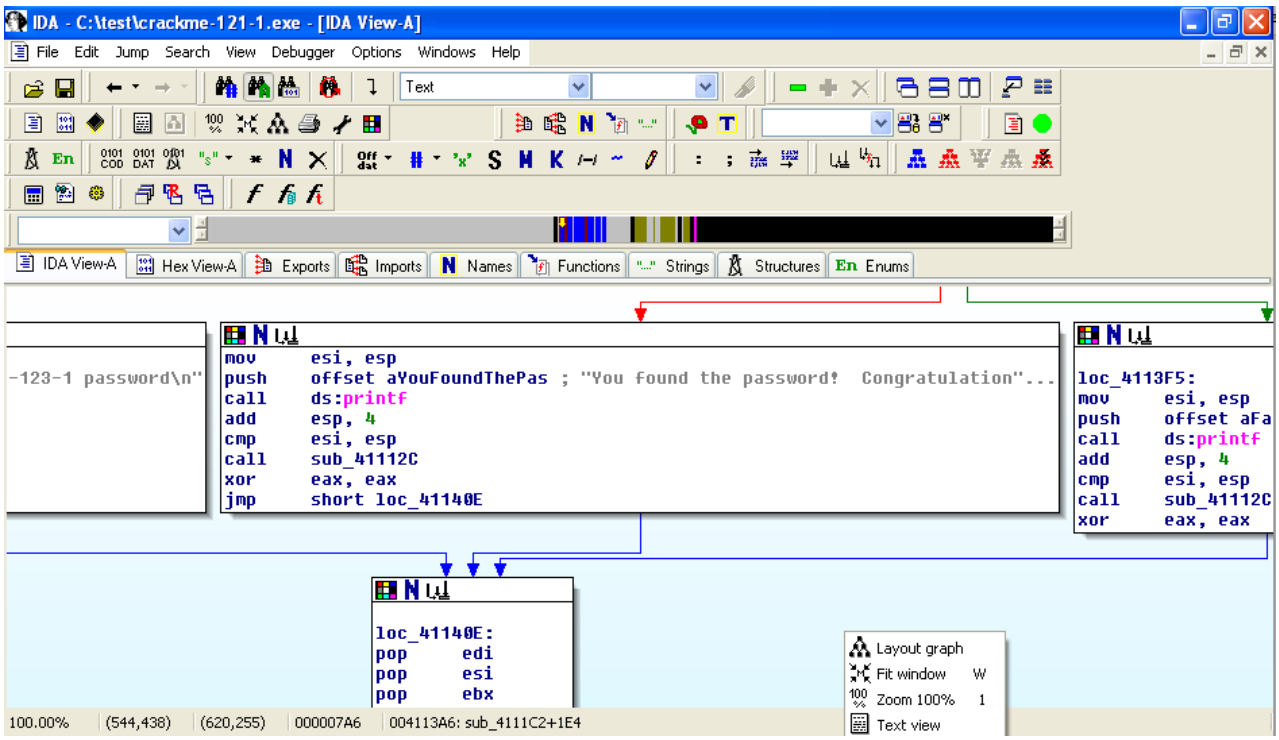
From the IDA menu bar, click **Search**, **Text**.

Search for **crackme** as shown below.

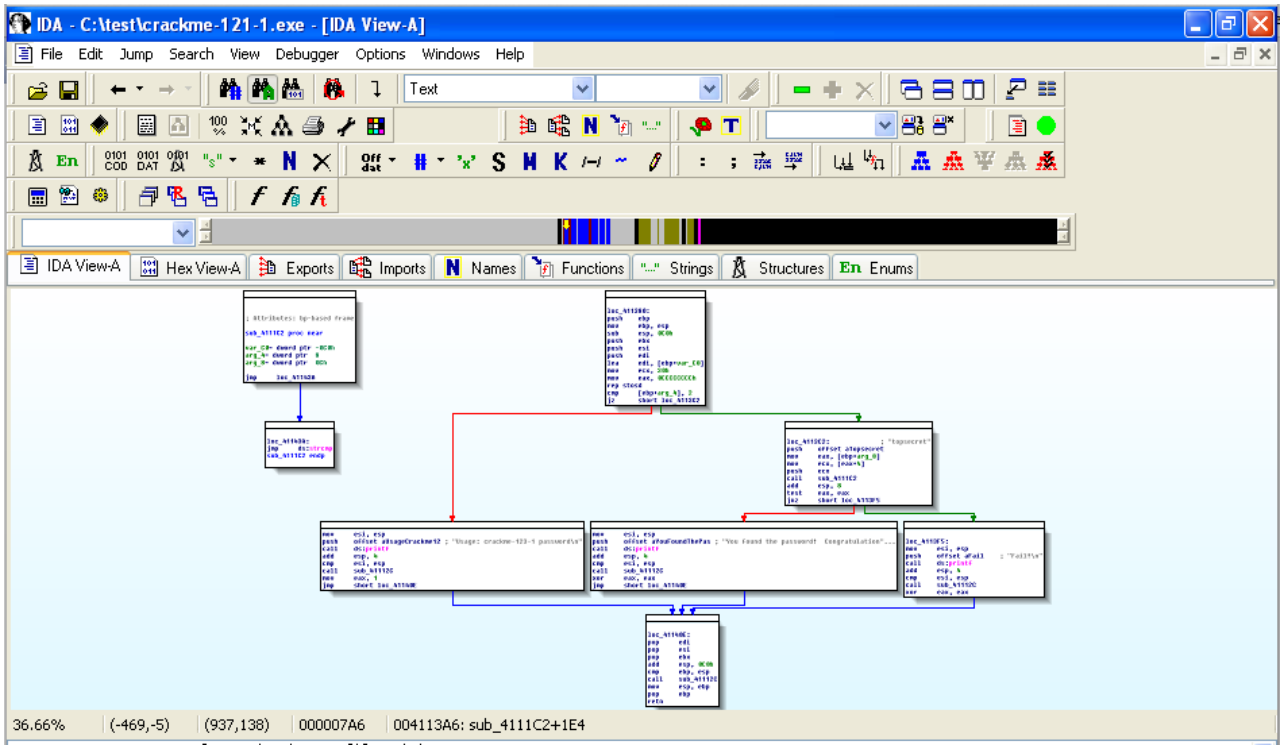
Click **OK**.



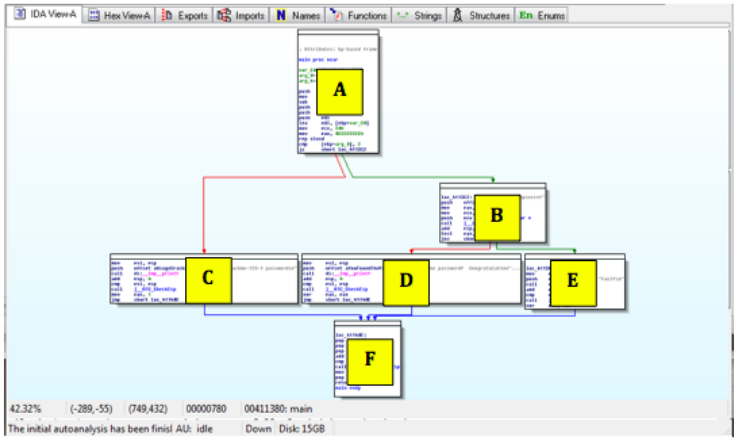
Right-click in the "View-A" box and click "**Fit window**", as shown below:



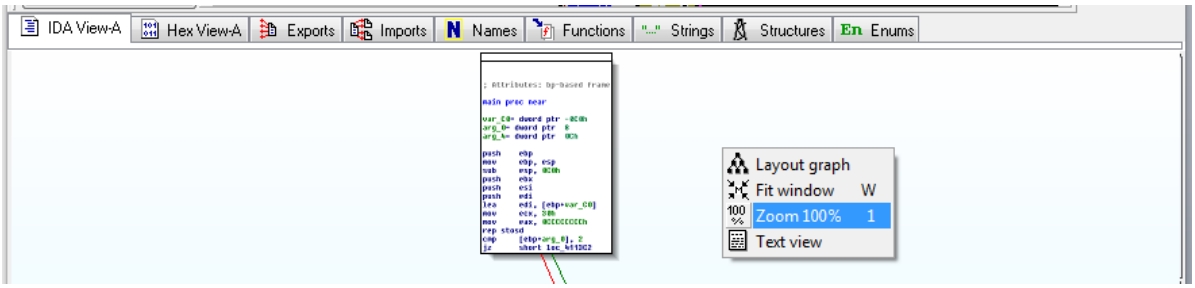
You should now see the entire program shown as six boxes connected by lines, as shown below. (Ignore the two extra boxes at the upper left):



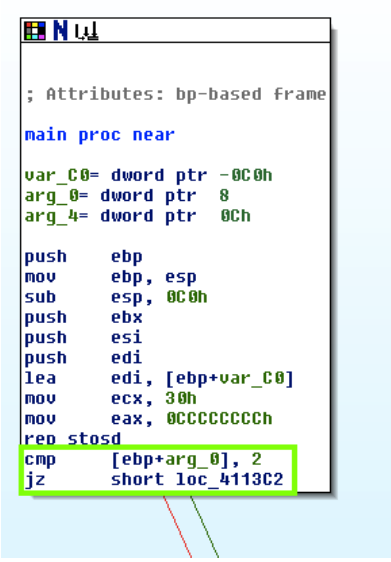
For this project, I have labelled the modules with letters as shown below:



Right-click in the "View-A" box and click "**Zoom 100%**", as shown below:



Click and drag the "View-A" display as needed to make module A visible, as shown below:



The assembly code is hard to read, but you don't need to understand it all. Focus on the last two instructions:

```
cmp     [ebp+arg_0], 2
jz      short loc_4113C2
```

This compares some number to 2 with the **cmp** (Compare) operation, and jumps to a different module if it is 2, using the **jz** (Jump if Zero) operation.

C Source Code

Here is the actual C source code for the file you are disassembling. Module A is the assembly code for the first "if" statement, labelled with the yellow "A" box below:

```
crackme-123-1.cpp
(Global Scope)

#include <iostream>
#include <string>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    if (argc != 2)        A
    {
        printf("Usage: crackme-123-1 password\n");
        return 1;
    }
    if (strcmp(argv[1],"topsecret") == 0)
    {
        printf("You found the password!  Congratulations!\n");
        return 0;
    }

    printf("Fail!\n");
    return 0;
}
```

Drag the "View-A" display to make Module C visible, as show below:

```
IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structu

mov     esi, esp
push    offset aUsageCrackme12 ; "Usage: crackme-123-1 password\n"
call    ds:__imp__printf
add     esp, 4
cmp     esi, esp
call    j__RTC_CheckEsp
mov     eax, 1
jmp     short loc_41140E
```

Notice the gray readable text on the right side, saying "Usage: crackme-121-1 password".

This module pushes those characters onto the stack with a **push** command, and then calls the printf function with the **call ds:_imp_printf** command.

The figure below shows the C statements that comple to the "C" module:

```
crackme-123-1.cpp
(Global Scope)

#include <iostream>
#include <string>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    if (argc != 2)        A
    {
        printf("Usage: crackme-123-1 password\n");   C
        return 1;
    }
    if (strcmp(argv[1],"topsecret") == 0)   B
    {
        printf("You found the password!  Congratulations!\n");   D
        return 0;
    }

    printf("Fail!\n");   E
    return 0;
}   F
```

Follow along in IDA Pro and make sure you see what each of the six modules do, and how they correspond to the C source code.

Adjusting Graph Mode Options

If the code doesn't show line numbers or hexadecimal instructions, click **Options**, **General**.

In the IDA Options box, make these changes:

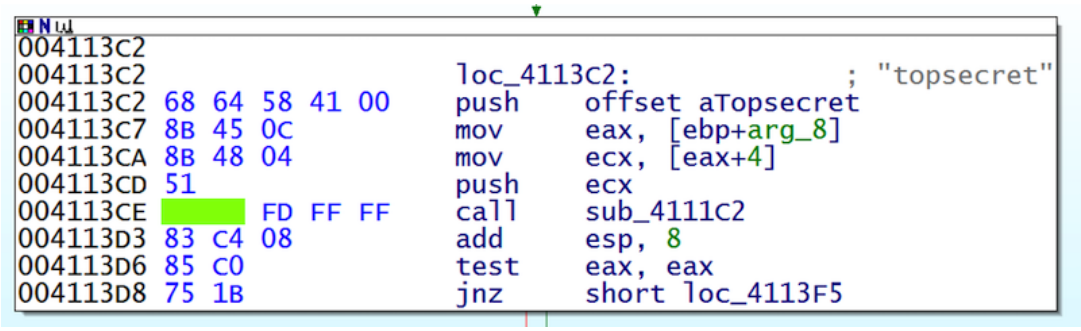
- Check "**Line prefixes**"
- Change "**Number of opcode bytes**" to **6**

Click **OK**.

Flag PMA 303.3: Finding the Password (5 pts)

Drag the "View-A" screen to show module "B", as shown below. The password "topsecret" is visible.

The flag is covered by a green box in the image below.



Running the Executable

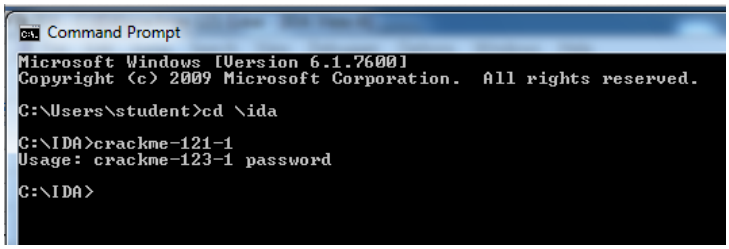
Click **Start**, type in **CMD**, and press Enter to open a Command Prompt window.

In the Command Prompt window, execute these commands:

```
cd \IDA

crackme-121-1
```

You should see the message "Usage: crackme-121-1 password", as shown below:



This message is telling you that you need to add a password after the "crackme-121-1".

In the Command Prompt window, execute this command:

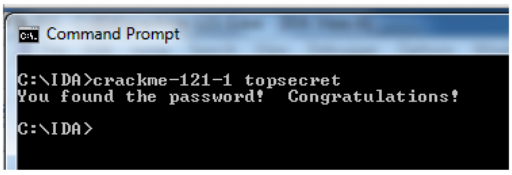
```
crackme-121-1 wrongpassword
```

You should see the message "Fail!".

In the Command Prompt window, execute this command:

```
crackme-121-1 topsecret
```

You should see the message "You found the password!", as shown below:



Flag PMA 303.4: crackme-121-2

Analyze crackme-121-2 in IDA. Find the password. Run the program in a Command Prompt with the correct password and verify that it produces the "Congratulations" message.

The password is the flag.

Flag PMA 303.5: crackme-121-3

Analyze crackme-121-3 in IDA. Find the password. Run the program in a Command Prompt with the correct password and verify that it produces the "Congratulations" message.

The password is the flag.

Flag PMA 303.6: crackme-121-4

Analyze crackme-121-4 in IDA. This one is different. Find the complete command line required to see the "Congratulations" message.

The flag is that complete command line, like this:

```
notepad.exe topsecret
```

Modified for WCIL 5-21-19
Renumbered and flags changed for CCSF use 9-3-19