126 2x: Reverse Engineering with IDA Pro Freeware (10-40 pts.)

What you need:

- A Windows computer (real or virtual) with an Internet connection

Purpose

You will use IDA Pro Free to disassemble and analyze Windows executable files.

Downloading an EXE to Examine

Create a working directory C:\IDA.

Download this file and move it to C:\IDA

- crackme-121-1.exe

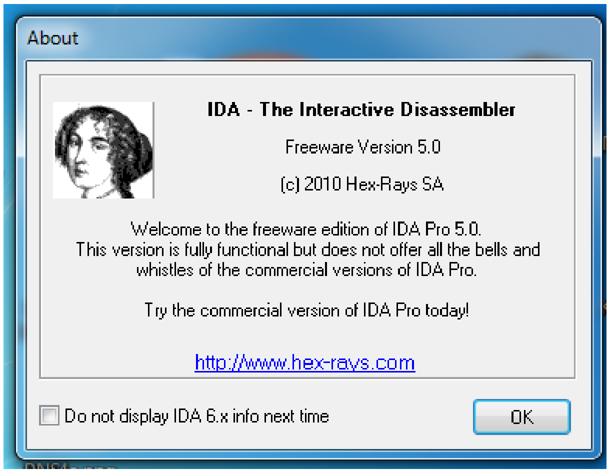
Downloading IDA Pro Free

Open a Web browser and go to http://www.hex-rays.com/products/ida/support/download_freeware.shtml

At the bottom of the page, click the "IDA Freeware (16mb)" link.

Install the software with the default options. I saw an error message saying something about a single-quote directory not found, but just closed it and it seemed not to matter.

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



IDA - The Interactive Disassembler

Freeware Version 5.0

(c) 2010 Hex-Rays SA

Welcome to the freeware edition of IDA Pro 5.0.
This version is fully functional but does not offer all the bells and whistles of the commercial versions of IDA Pro.

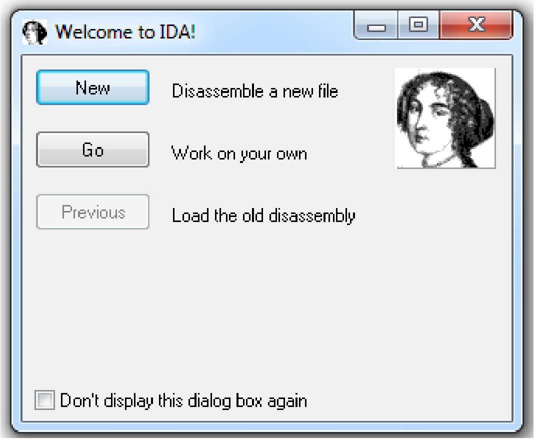
Try the commercial version of IDA Pro today!

http://www.hex-rays.com

□ Do not display IDA 6.x info next time

Click "**I Agree**".

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



New — Disassemble a new file

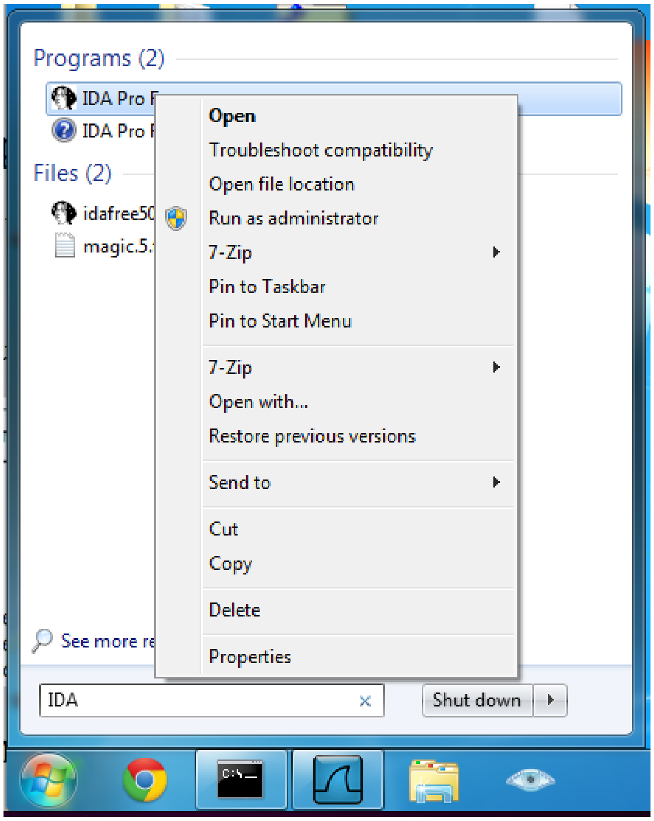
Go — Work on your own

Previous — Load the old disassembly

□ Don't display this dialog box again

If you are using Windows 7, IDA crashes. It needs Administrator privileges.

Click **Start**, type **IDA**, right-click "**IDA Pro Free**", and click "**Run as Administrator**", as shown below:



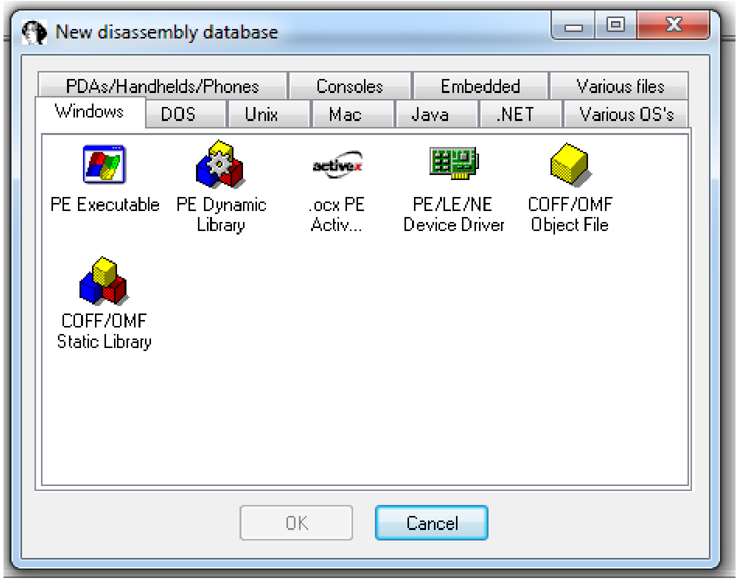
If a "User Account Control" box pops up, click **Yes**.

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

Loading the EXE File

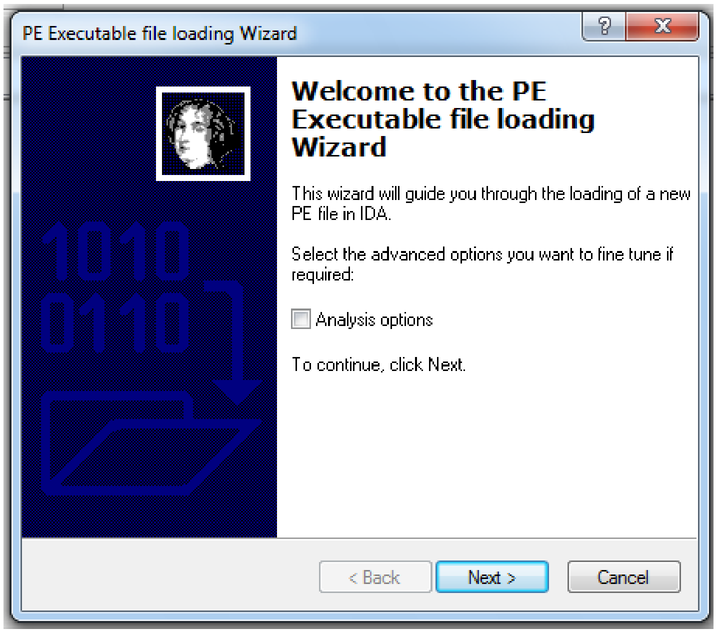
In the "Welcome to IDA" box, click the **New** button.

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 the **crackme-121-1.exe** file you saved earlier in the C:\IDA directory and double-click it.

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



PE Executable file loading Wizard

Welcome to the PE Executable file loading Wizard

This wizard will guide you through the loading of a new PE file in IDA.

Select the advanced options you want to fine tune if required:

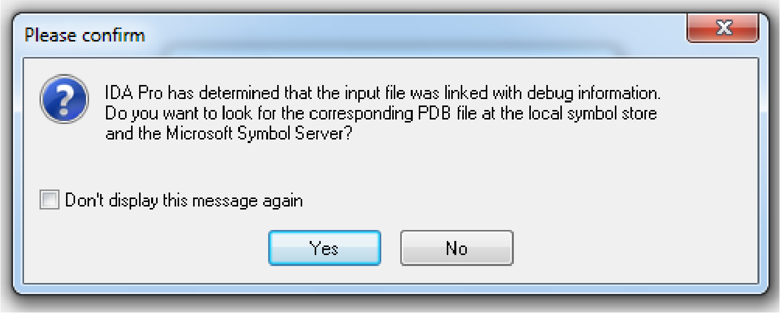
□ Analysis options

To continue, click Next.

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.



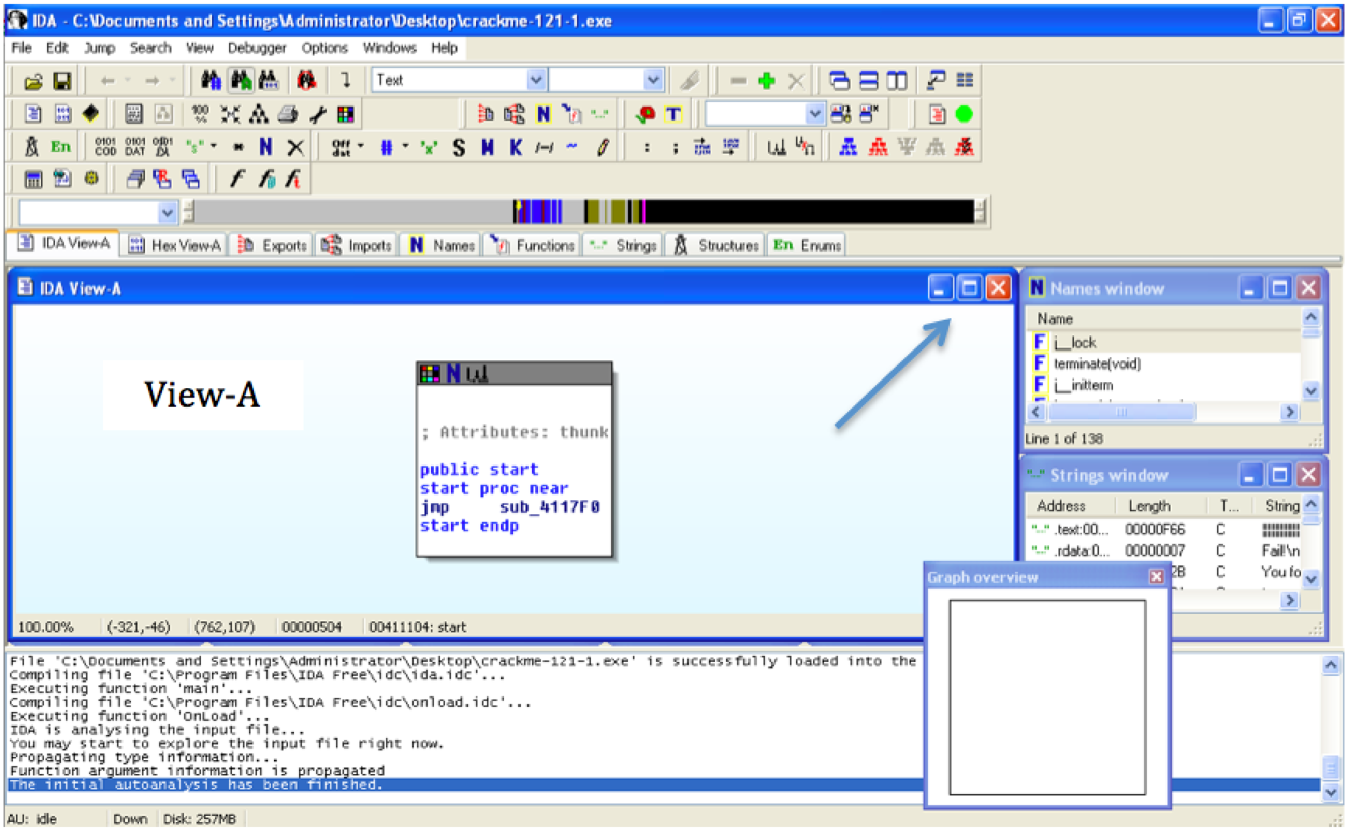
Please confirm

IDA Pro has determined that the input file was linked with debug information. Do you want to look for the corresponding PDB file at the local symbol store and the Microsoft Symbol Server?

□ Don't display this message again

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:



IDA - C:\Documents and Settings\Administrator\Desktop\crackme-121-1.exe
File Edit Jump Search View Debugger Options Windows Help
IDA View-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | Enums
IDA View-A
View-A
Names window
Strings window
Graph overview

```
; Attributes: thunk

public start
start proc near
jmp     sub_4117F0
start endp
```

100.00% (-321,-46) (762,107) 00000504 00411104: start

Name
j__lock
terminate(void)
j__initterm

Line 1 of 138

Address	Length	T...	String
.text:00...	00000F66	C	##########
.rdata:0...	00000007	C	Fail!\n
	...2B	C	You fo

```
File 'C:\Documents and Settings\Administrator\Desktop\crackme-121-1.exe' is successfully loaded into the
Compiling file 'C:\Program Files\IDA Free\idc\ida.idc'...
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.
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.
```

AU: idle | Down | Disk: 257MB

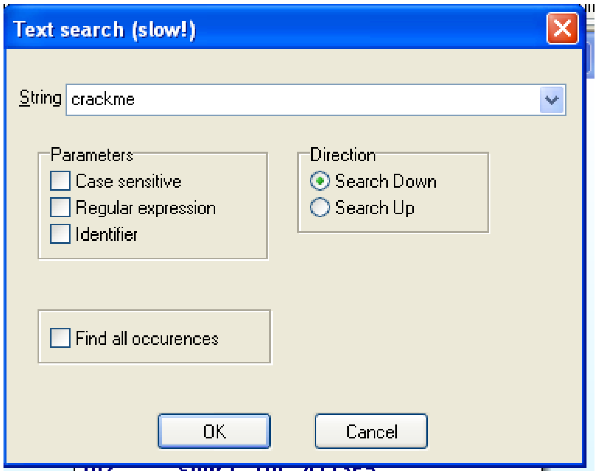
Close the "Graph Overview" box in the lower right corner.

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**.



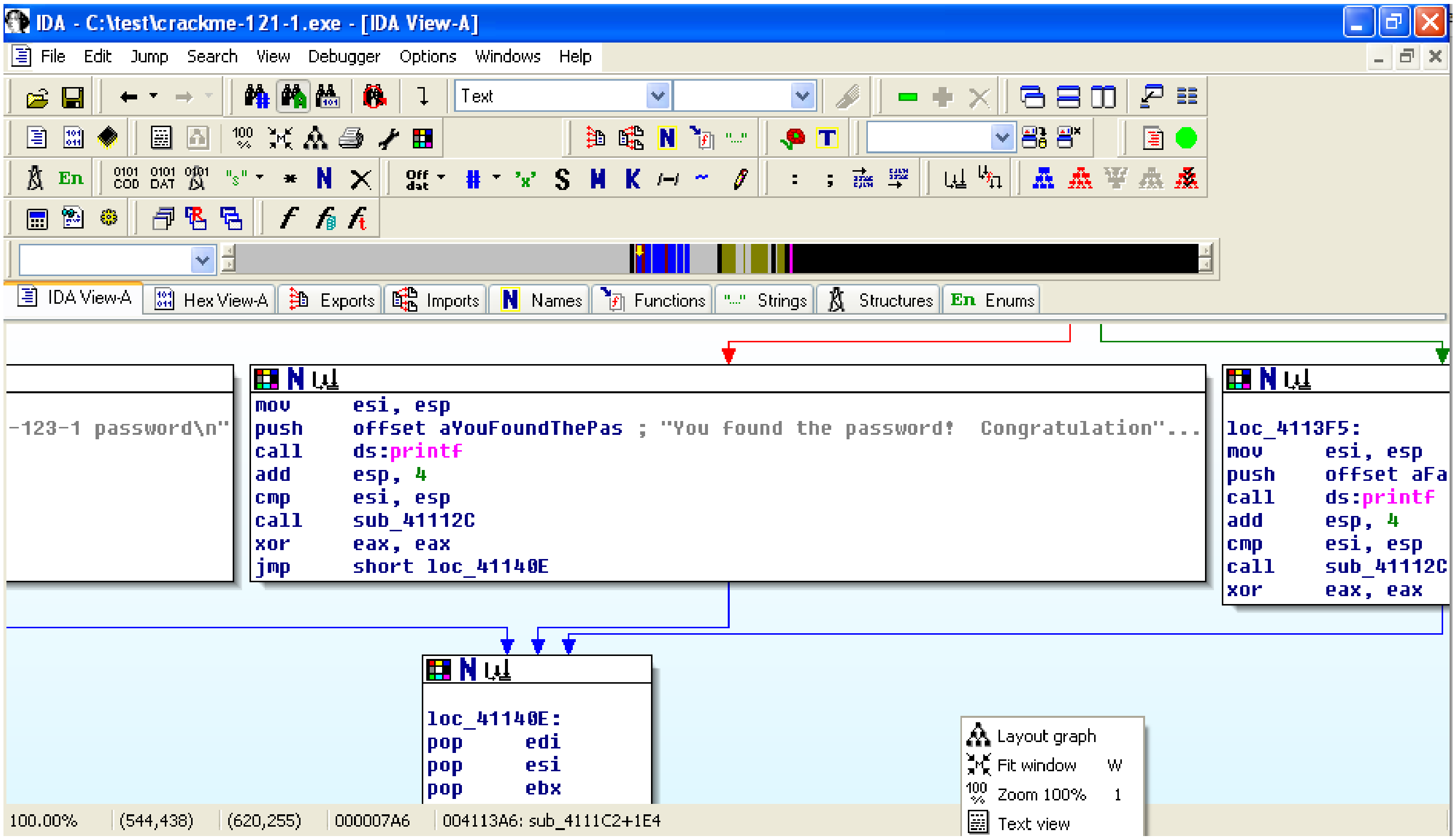
String: crackme

Parameters
□ Case sensitive
□ Regular expression
□ Identifier

Direction
● Search Down
○ Search Up

□ Find all occurences

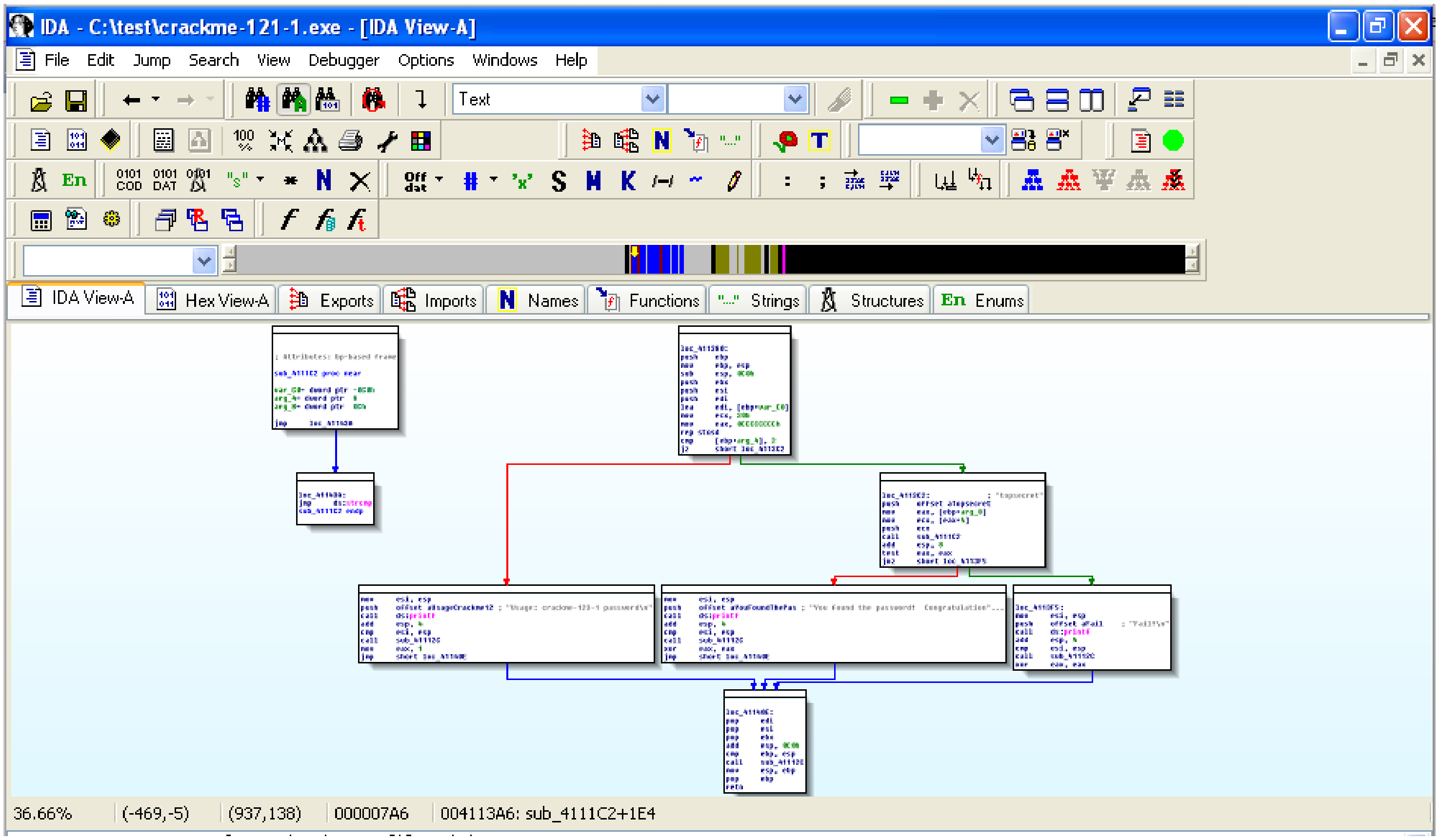
Right-click in the "View-A" box and click "**Fit window**", as shown below:



```
-123-1 password\n"

mov     esi, esp
push    offset aYouFoundThePas ; "You found the password!  Congratulation"...
call    ds:printf
add     esp, 4
cmp     esi, esp
call    sub_41112C
xor     eax, eax
jmp     short loc_41140E

loc_4113F5:
mov     esi, esp
push    offset aFa
call    ds:printf
add     esp, 4
cmp     esi, esp
call    sub_41112C
xor     eax, eax

loc_41140E:
pop     edi
pop     esi
pop     ebx
```

```
100.00%   (544,438)   (620,255)   000007A6   004113A6: sub_4111C2+1E4
```

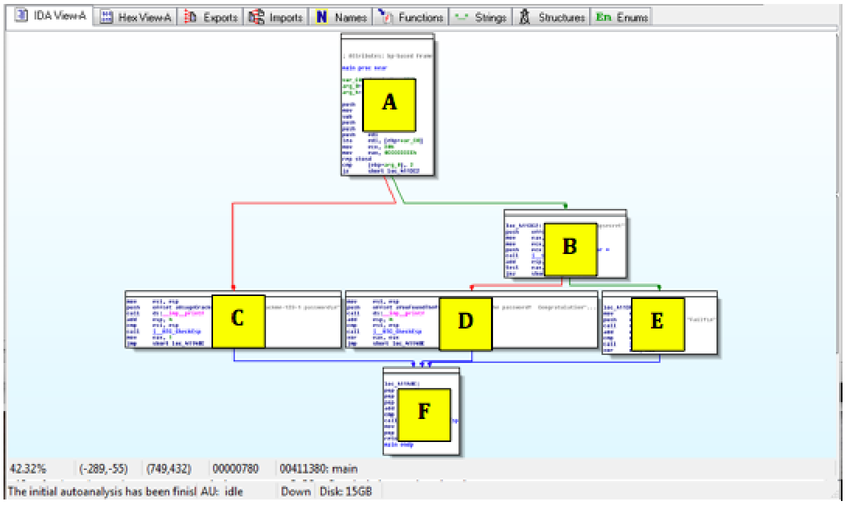
You should now see the entire program shown as six boxes connected by lines, as shown below. (Ignore the two extra boxes at the upper left):


IDA - C:\test\crackme-121-1.exe - [IDA View-A]
File Edit Jump Search View Debugger Options Windows Help
Text
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Strings
Structures
Enums

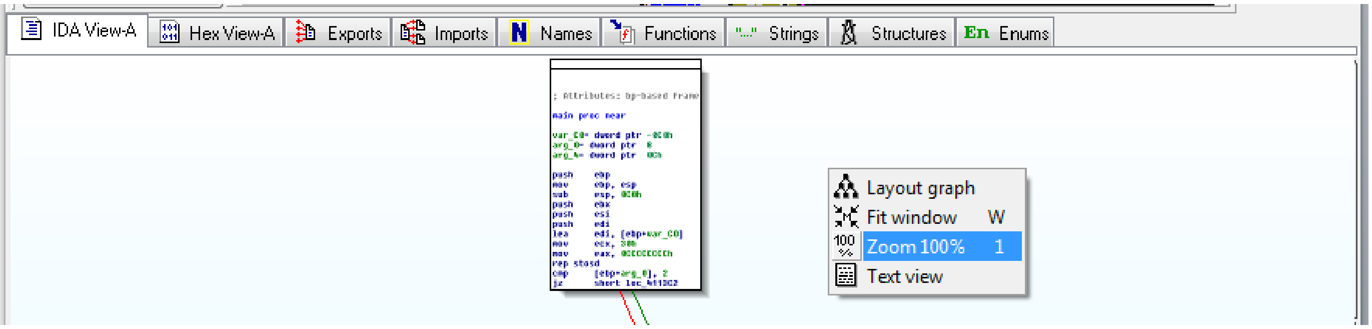
```
; Attributes: bp-based frame

sub_4111C2 proc near

var_C0= dword ptr -0C0h
arg_4= dword ptr  8
arg_8= dword ptr  0Ch

jmp     loc_411320

loc_4114D6:
jmp     ds:strcmp
sub_4111C2 endp

loc_41138C:
push    ebp
mov     ebp, esp
sub     esp, 0C0h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_C0]
mov     ecx, 30h
mov     eax, 0CCCCCCCCh
rep stosd
cmp     [ebp+arg_4], 2
jz      short loc_4113C3

loc_4113C3:                     ; "topsecret"
push    offset aTopsecret
mov     eax, [ebp+arg_8]
mov     ecx, [eax+4]
push    ecx
call    sub_4111C2
add     esp, 8
test    eax, eax
jnz     short loc_4113F5

mov     esi, esp
push    offset aUsageCrackme12 ; "Usage: crackme-123-1 password\n"
call    ds:printf
add     esp, 4
cmp     esi, esp
call    sub_41112C
mov     eax, 1
jmp     short loc_41140E

mov     esi, esp
push    offset aYouFoundThePas ; "You found the password!  Congratulation"...
call    ds:printf
add     esp, 4
cmp     esi, esp
call    sub_41112C
xor     eax, eax
jmp     short loc_41140E

loc_4113F5:
mov     esi, esp
push    offset aFail    ; "Fail!\n"
call    ds:printf
add     esp, 4
cmp     esi, esp
call    sub_41112C
xor     eax, eax

loc_41140E:
pop     edi
pop     esi
pop     ebx
add     esp, 0C0h
cmp     ebp, esp
call    sub_41112C
mov     esp, ebp
pop     ebp
retn
```

```
36.66%   (-469,-5)   (937,138)   000007A6   004113A6: sub_4111C2+1E4
```

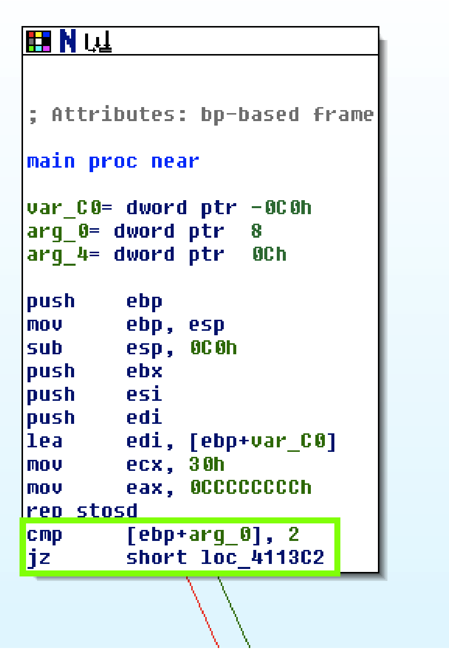
For this project, I have labelled the modules with letters as shown below:


IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Strings
Structures
Enums
A
B
C
D
E
F
42.32% (-289,-55) (749,432) 00000780 00411380: main
The initial autoanalysis has been finisl AU: idle Down Disk: 15GB

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:



```
; Attributes: bp-based frame

main proc near

var_C0= dword ptr -0C0h
arg_0= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 0C0h
push    ebx
push    esi
push    edi
lea     edi, [ebp+var_C0]
mov     ecx, 30h
mov     eax, 0CCCCCCCCh
rep stosd
cmp     [ebp+arg_0], 2
jz      short loc_4113C2
```

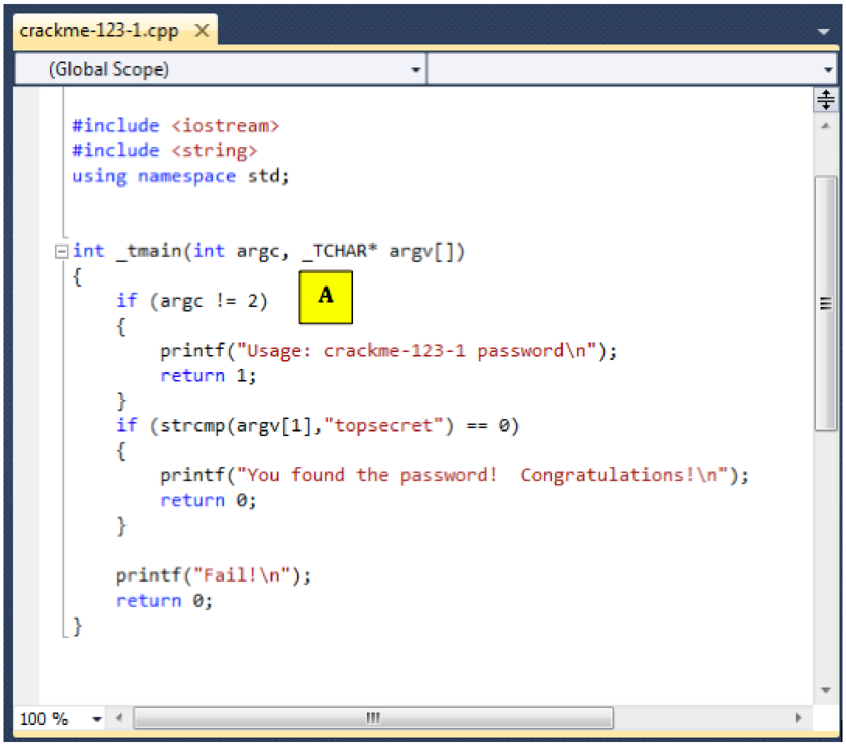
The assembly code is hard to read, but you don't need to understand it all. Focus on the last two instructions:

```
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.

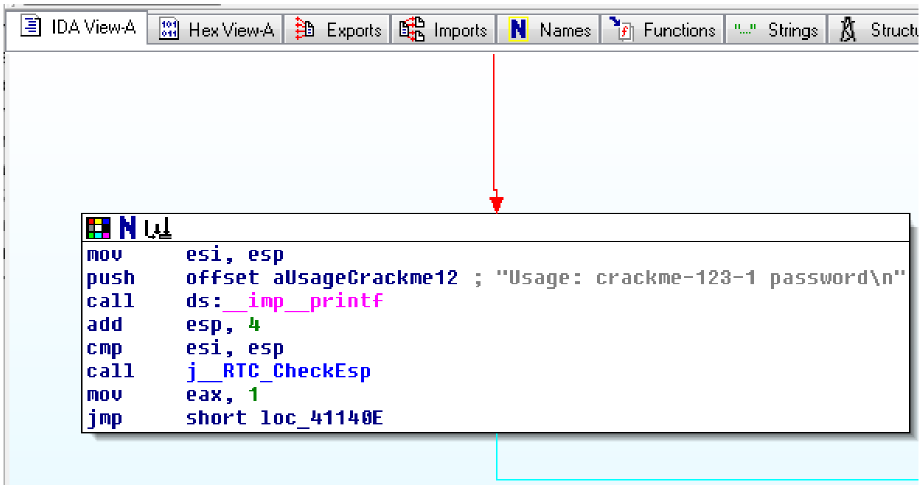
C Source Code

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:



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

int _tmain(int argc, _TCHAR* argv[])
{
    if (argc != 2)
    {
        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:

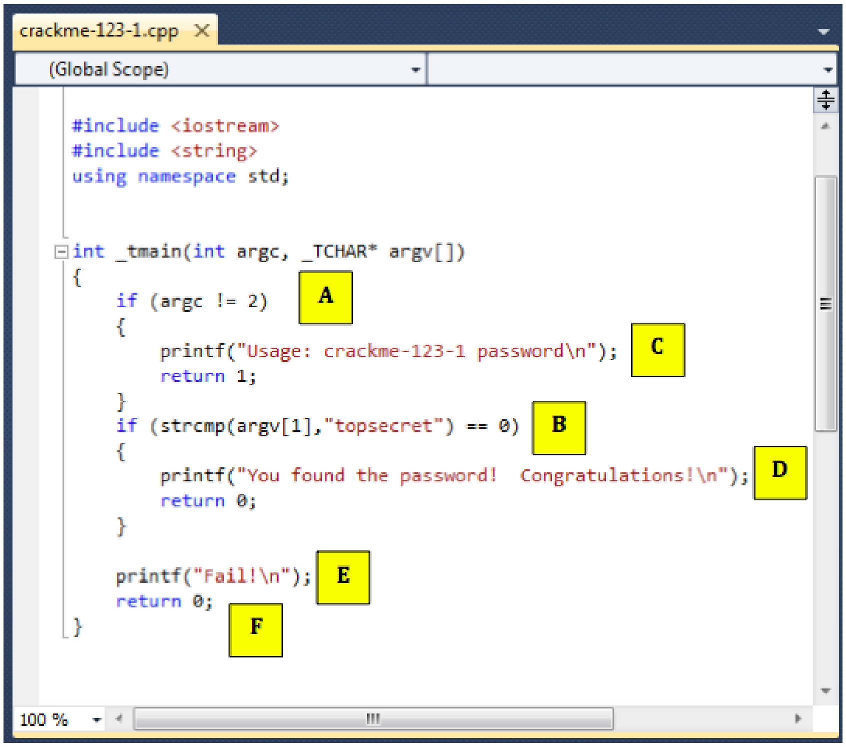


```
mov     esi, esp
push    offset aUsageCrackme12 ; "Usage: crackme-123-1 password\n"
call    ds:__imp__printf
add     esp, 4
cmp     esi, esp
call    j__RTC_CheckEsp
mov     eax, 1
jmp     short loc_41140E
```

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:

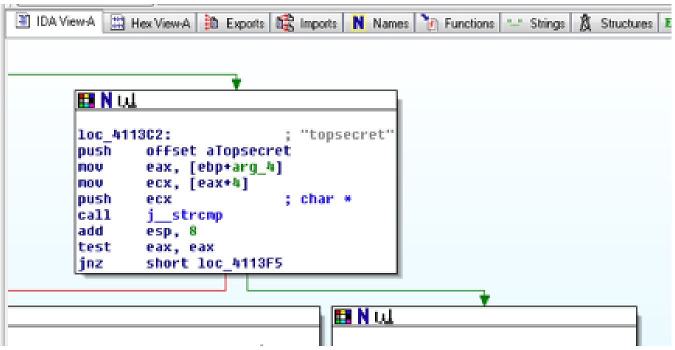


```
#include <iostream>
#include <string>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    if (argc != 2)
    {
        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;
}
```

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.

Saving the Image

Drag the "View-A" screen to show module "B", as shown below:



```
loc_4113C2:                     ; "topsecret"
push    offset aTopsecret
mov     eax, [ebp+arg_4]
mov     ecx, [eax+4]
push    ecx                     ; char *
call    j__strcmp
add     esp, 8
test    eax, eax
jnz     short loc_4113F5
```

Make sure the gray "topsecret" text is visible.

Save this image with the filename **Proj 2xa from YOUR NAME**

Running the Executable

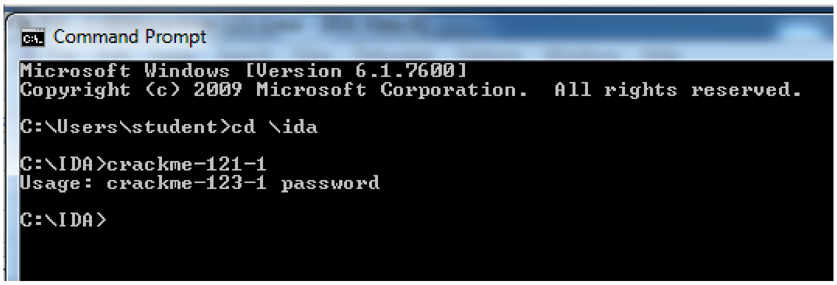
Click **Start**, type in **CMD**, and press Enter to open a Command Prompt window.

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:



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\student>cd \ida

C:\IDA>crackme-121-1
Usage: crackme-123-1 password

C:\IDA>
```

If you see a message saying "This application has failed to start because MSVCR100D.dll was not found", download that file here, and put it in the same folder as the .exe file:

msvcr100d.dll

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:

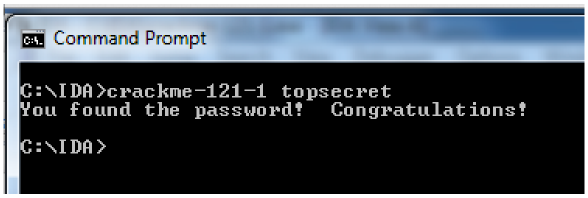
```
crackme-121-1 wrongpassword
```

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:



```
C:\IDA>crackme-121-1 topsecret
You found the password!  Congratulations!

C:\IDA>
```

Saving the Image

Make sure the "You found the password!" text is visible.

Save this image with the filename **Proj 2xb from YOUR NAME**

Point Value

Those two images are worth a total of ten points. You can now earn more points by using the same technique to crack more files, as explained below.

crackme-121-2 (10 points)

Download this file:

crackme-121-2.exe

It is very similar to crackme-121-1. Perform these steps:

1. Load the executable in IDA Pro
2. Find the module containing the password, and save a screen capture of it
3. Run the program at a command prompt and save an image of it congratulating you for finding the password.

crackme-121-3 (10 points)

This one is a little more complicated, with two passwords instead of just one.

Download this file:

crackme-121-3.exe

Perform these steps:

1. Load the executable in IDA Pro
2. Find the modules containing the passwords, and save a screen capture of them
3. Run the program at a command prompt and save an image of it congratulating you for finding the passwords.

crackme-121-4 (10 points)

This one is a little more complicated--you need to do more than just provide a password.

Download this file:

crackme-121-4.exe

Perform these steps:

1. Load the executable in IDA Pro
2. Find the modules that perform string comparisons (strcmp) and try to guess what they are referring to.
3. Run the program at a command prompt and save an image of it congratulating you for solving the puzzle.

Turning in your Project

Email the images to cnit.126sam@gmail.com with the subject line: **Proj 2x from YOUR NAME**

Credits

This is based on a class I took at the HoneyNet conference, from Felix Leder.

Last modified 7-22-14