

Forum:[Hack Forums](#)

Thread: [\[TUT\] How to Crack Programs For Beginners with IDA Pro](#)

Zer0Memo0ry - January 4, 2017, 7:13 am

In this simple tutorial you'll learn the basics of statically patching a program.

We will be using IDA Pro to crack an easy crackme, you'll need:

- IDA Pro
- Basic Knowledge of the x86/x86_64 architecture, assembly and intel syntax.
- Patience

Alright, so let's start with this one:

It's only 12kb, so it will be likely really, really easy to analyze, reverse and crack.

When ran, the crackme asks for a password:

Alright, so let's open up IDA Pro. Drop the target executable to IDA and you'll be prompted by this window:

Usually you never have to change anything there, IDA automatically recognizes the best default options for each executable. If the file is packed then you might need to alter the settings. Just press OK to continue.

this is my default IDA layout, in the center you can see the disassembly of our target program's main function
which what we'll be focusing on first.

If you press F5 on the disassembly window you can see the pseudo C code generated by IDA of the disassembly.

It can be really useful since in some scenarios it can really be close to the original code, although nearly always it will not resemble the original code. Pseudo code gives you a good idea of the function. Sometimes it might be full of junk or something else that makes the function hard to understand, in such case it will be easier to examine the disassembly first and try to understand it.

Now we can start renaming some of the functions of the program to make it easier to understand in the first place;

As you can see, this function 'sub_401740' is called with arguments "std::cout, "string"" in every case.

From this we can see that the function is actually the STL's "cout <<" operation. So let's rename it to "std::cout":

'sub_4019B0' is STL's cin operator and so on.

Now that the pseudocode is a bit easier to read, we can see that this is where the algorithm compares our inputkey and the correct key.

As you can see it's not a direct string comparison, the algorithm works differently.

Probably by calculating each character's hex value together and then comparing the sum to the input key or something similar to that.

So let's switch to the disasm view and try to find the point where the program does this step.

Now let's switch to graph view:

Here we can see the flow of the main function, which contains the input and key comparison.

Here we can see the key validation, as you can see there are three jumps; jnz, jb and jnz. to the "wrongkey" location:

Now we can just patch the program to jump to "rightkey" location instead of jumping to "wrongkey".

Now the program jumps to the "rightkey" location, no matter what we type as the inputkey.

If we check the pseudocode of the main function again, we can see that the if-else comparison statements have disappeared completely, and the program directly outputs "Well done, you gave the right key.".

Now just apply patches to the input file and see how it works:

As you can see, the program does not care what the input key is this time, and will let us in regardless.

SIM-1534439 - January 4, 2017, 7:17 am

Really nice tutorial man!

Good job

Zer0Memo0ry - January 4, 2017, 7:22 am

original crackme, cracked crackme and IDA pro database of the file:

<https://github.com/Zer0Mem0ry/CrackMe>

(Jan 4, 2017 08:17 AM)Trukfit Wrote: [\[CITE\]](#) Really nice tutorial man!

Good job

Thanks, be sure to +rep ;)

tinal1 - *August 8, 2018, 6:09 am*

Very good tut for beginner like me ..thankyou