

Forum:[Hack Forums](#)

Thread: [\[TUTORIAL\] Basic WinDbg + SOS .NET Runtime cracking WITHOUT deobfuscator](#)

hgdfhsdgdgdfh - February 4, 2012, 3:09 am

I assume you have installed WinDbg 32-bit version (<http://msdn.microsoft.com/en-us/windows/hardware/gg463009> , <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=8279>) and set-up symbols etc (<http://msdn.microsoft.com/en-us/windows/hardware/gg462988>).

First few basic WinDbg commands:G - Run

P - Step

A <address> - Assemble mode, modify asm code at address

r @eax/@ebx... - Dump register content

r @eax/@ebx... = value - Set register content

Ok, lets start cracking. I'm using PolarX crackme (<https://hackforums.net/showthread.php?tid=2176139>) as an example. Credits for him.

Start up WinDbg and go to File -> Open Executable... and open crackme exe from there. Something like

[img]http://img638.imageshack.us/img638/6514/tmpd1cf.png[/img]

should pop up.

Thats the initial pause when starting executable, no code has been run yet. Now you must type either

Code sxe ld:mscorwksor

Code sxe ld:clrdepending on .NET version. CLR for .NET 4, mscorwks for older. If you aren't sure you can type both. These set breakpoints to module load (clr.dll and mscorwks.dll). After you have typed either one or both, type G to continue run.

This pops up

[img]http://img842.imageshack.us/img842/5214/tmpd401.png[/img]

It means that CLR.dll is being loaded. Now you can type

Code .loadby SOS clr/mscorwks (depending on which module got loaded, clr or mscorwks)This loads the SOS plugin from same place as clr/mscorwks. Now WinDbg is set up for debugging .NET applications.

Now the cracking phase. You can see the crackme will show a message box when you enter incorrect password. Thats why we are going to breakpoint messagebox API call. Type

Code bp MessageBoxWto set the breakpoint. If you get ERRORS about symbols, they arent fatal and in most cases you should be able to continue. You can confirm that breakpoint was set by typing

Code blto list all breakpoints.

Here you can see breakpoint added

[img]http://img814.imageshack.us/img814/7119/tmp9e03.png[/img]

Now when MessageBoxW is breakpointed, type G to run the program. Window pops up, enter something random to password and click the button. You wont see the incorrect password window appear yet. Instead on WinDbg window you will see something like

[img]http://img706.imageshack.us/img706/2319/tmp4d64.png[/img]

That means message box was just about to be called by crackme code.

Type

Code !clrstack to see .NET internal call stack

[img]http://img838.imageshack.us/img838/7574/tmpd8a2.png[/img]

Its long output but you can see only line with obfuscated looking name with weird characters. Thats the crackme function.

Code 0040e8dc 00244b86 . . (System.Object, System.EventArgs)

00244b86 is the instruction address in that function.

Type

Code !ip2md <instruction addr> (in my case 00244b86 but it changes on each run)and you will get something similar to

[img]http://img718.imageshack.us/img718/7405/tmpf17f.png[/img]

Here MethodDesc is the thing we are interested about.

If you type

Code !dumpil <methoddesc> (in my case 001d802c but again its not same always)you get the function code MSIL:

[img]http://img213.imageshack.us/img213/4981/tmp6bad.png[/img]

It may look confusing but after bit of visual analyzing you can see that on

[img]http://img62.imageshack.us/img62/7104/tmp39d5.png[/img]

It jumps to IL_25 after calling CompareString. This must be the password comparison. On IL_25 you can see something usually used for comparison, brfalse/brtrue

[img]http://img198.imageshack.us/img198/1251/tmp5c81.png[/img]

This must be the check location.

To modify it, type

Code !u <methoddesc> (same as used on !dumpil)What you see is the JIT compiled native code of crackme method. This can be modified to bypass things.

After bit of scrolling through the code you can see that only conditional jump (jump depending on condition) is at

[img]http://img195.imageshack.us/img195/2291/tmpecbf.png[/img]

Pretty beginning. There is JNE (jump if not equals).

In order to crack this crackme, we have to swap the opcode to JE (jump if equals).

You see on line

Code 00244b3e 751b jne 00244b5bbeginning 00244b3e is address of that instruction.

So you have to type:

Code A <instruction addr> (00244b3e for me)and then you can input new instruction which will replace the original on address 00244b5b.

Type

Code JE 0x<jump location>Jump location is taken from original instruction, but you have to add the 0x hexadecimal prefix.

After you have typed that, press enter again to exit from assembly mode.

Type G to run the application and enter something into password box and click the button. You will probably get MessageBox breakpoint again but if you do, you can just type G to continue directly.

Message box will show you that password was valid :P

[img]http://img855.imageshack.us/img855/8082/tmp384f.png[/img]

The main advantage in using windbg IMO is that you dont have to deobfuscate the application and its usually pretty easy to find the check location. Bad thing is that everything happens runtime,

AFAIK you cant modify the IL code and dump that to file. And if you did, you still would have to deal with anti-tamper.

Credits:
CrackMe - PolarX

ubbelol - February 4, 2012, 3:44 am

Amazing tutorial! Was waiting for a good Windbg tutorial! :D

Callum. - February 4, 2012, 3:46 am

VERY HQ Tutorial man, i love Tutorials like these, gives me hope in HF :)

Stenography - February 4, 2012, 4:39 am

Thanks for giving me credits,

Anyone who manage to crack it with 0x's tutorial and post a proof will be added to the crackers list.

@OP, I think you should show how to get the original password too !

Deceptive - February 4, 2012, 9:00 am

Thank you was looking for WinDbg tutorials, always general and never really this specific.

SpK - February 9, 2012, 8:02 am

One of the best tutorials out there, and deserves to get more attention.

Abuser™ - February 9, 2012, 6:00 pm

Great TuT!

Keep it on the top of this subforum!