CreateRemoteThread 32->64 and/or 64->32

Asked 15 years, 10 months ago Modified 2 years, 8 months ago

Viewed 6k times

4

I need a way to CreateRemoteThread in x64 windows into both 64 and 32 bit processes. I've worked out how to find instruction set of target process, how to allocate memory in the target process for the assembly sled, and I've almost worked out what to do about address space randomization.

I don't know how to actually start the thread on the remote process when it is of the wrong instruction set.

Notice: I don't care which of the two problems you solve. My own exe can be either 32 or 64 bits (but I really do have to choose before I know the number of bits of the target process).

Before somebody complains that I really shouldn't have to do this, ask Microsoft why I have to set `FILE_SHARE_DELETE` on all open handles before I can delete a file that is in use. No, there's no way around needing to delete files that other process have open either.

multithreading winapi

Share

Improve this question

Follow

edited Jun 21, 2009 at 22:01

Shog9
159k ● 36 ● 235 ● 240

asked Jan 30, 2009 at 2:22

Joshua
43.1k ● 9 ● 78 ● 148

This really did diserve insane. – Joshua May 12, 2009 at 21:15

You ever get CreateRemoteThread 32 -> 64 to work? – QAZ Jan 19, 2010 at 18:21

3 Answers

Sorted by: Highest score (default)

6

CreateRemoteThread 32->64 doesn't work.

CreateRemoteThread 64->32 works.

Share Improve this answer

Follow

answered Feb 4, 2009 at 20:17

Joshua
43.1k ● 9 ● 78 ● 148

I was kind of hoping for self-learner badge here but I think I'm not going to get it. – Joshua Feb 5, 2009 at 3:49

6

The following source code, which performs normal as well as X86->X64 and X64->X86 injection, has all the details you need: https://github.com/OpenWireSec/metasploit/blob/master/external/source/meterpreter/source/common/arch/win/i386/base_inject.c

The short story is that it involves lots of architecture-specific "undocumented" functionality, since you must execute 64-bit code in a 32-bit WoW process to perform X86-X64.

But that code has been working well for many versions of Windows.

Share Improve this answer Follow

edited Apr 9, 2022 at 13:55

Paul
6,801 ● 8 ● 50 ● 75

answered May 28, 2012 at 20:39

Matt
993 ● 7 ● 2

1 Working URL: github.com/OpenWireSec/metasploit/blob/master/external/source/… – Paul Mar 23, 2015 at 15:37

2

As a work-around for your stated problem, you could and probably should use the deferred delete on reboot functionality provided by the operating system if you need to delete files in use by other processes, and the other processes have not opened them for FILE_SHARE_DELETE. It's likely much less potentially dangerous than modifying existing file handles in foreign processes with thread injection, not to mention potentially requiring less privileges. Just a thought.

If you're set on remote thread injection, see the workarounds on the MSDN page; maybe there's some inspiration there. You could also consider just brute-force killing the other processes (nicely first, forcefully as necessary), since you're goona need admin access anyway, and mucking with their internal handles might not leave them in a good state. That's what installers do (or ask the user to do) when they need to replace open files without a reboot.

Share Improve this answer Follow

edited Jan 30, 2009 at 7:26

answered Jan 30, 2009 at 2:42

Nick

6,846 ● 1 ● 24 ● 34

Nope, DeleteOnReboot() requires admin rights. I know exactly what processes are holding the file open and don't even need to scan for them and furthermore know they are by the same user. I have considered other mechanisms and none show promise. – Joshua Jan 30, 2009 at 3:43

Thread injection cross-process requires debugging rights iirc, which is equivalent to admin. You're right, though... deleting on reboot requires equivalent rights. – Nick Jan 30, 2009 at 7:21

Nick, Thread injection for process created by same user does not require admin rights. – Joshua Feb 4, 2009 at 20:16

One more thing, all my injected thread does is add FILE_SHARE_DELETE to the open attributes. I have yet to see a program misbehave because of that. – Joshua Apr 14, 2009 at 4:14