The history of the EncodePointer function for obfuscating pointers

devblogs.microsoft.com/oldnewthing/20201113-00

November 13, 2020

Raymond Chen

Windows XP introduced the function `EncodePointer` whose mission was to take a pointer value and obfuscate it in a way that is different for each process. The purpose is to add a layer of defense in depth: If an attacker is able to extract the encoded pointer value, they nevertheless don't get very much information about the address space layout of the target process. This makes it harder to attack the target with, say, write-what-where attack to overwrite a function pointer: Before they can overwrite the pointer, they need to be able to encode it in the specific way that is custom to the target, so that when the target decodes it, the result is the value the attacker desires.

Windows XP took the extra step of taking the information required to encode and decode pointers, and moving some of it out of the process address space. So even if somebody got free access to the entire process address space, they *still* couldn't figure out how to encode and decode pointers.

Some of the critical information was kept in kernel mode, and each time the target process wanted to encode or decode a pointer, it called into the kernel to say, "Please do your extra secret magic sauce."

In Windows 10 version 1809, the secret magic sauce was brought into the process to avoid a kernel transition every time a pointer needed to be encoded or decoded. This was good news, bad news, and mitigated news.

The good news is that this makes encoding and decoding pointers much, much faster.

The bad news is that this makes all of the information needed to encode and decode pointers visible to user-mode.

The mitigated news is that in the time since pointer encoding was introduced, there have been a lot of changes which mitigate the security impact. For example, address space layout randomization (ASLR) makes it harder to predict where that last piece of information went.

And control flow guard (CFG) makes it harder to get control to jump through a function pointer to an address of your choosing.

Raymond Chen

Follow