On enabling NX and ASLR for a module after the fact

devblogs.microsoft.com/oldnewthing/20170607-00

June 7, 2017

Raymond Chen

A customer wanted to enable NX (also known as Data Execution Prevention, or DEP) and ASLR for some executables and DLLs. There are two ways of doing this:

- Enable the options at link time by passing the linker command line options `/NX-COMPAT` and `/DYNAMICBASE`, and for good measure `/HIGHENTROPYVA`.
- Build the modules the usual way, and then use the `EDITBIN` program with those same command line options to enable the features on the files in question.

For reasons the customer didn't provide (but which I can guess),[1] there are a handful of files that they cannot relink, so they are forced to use the `EDITBIN` approach for those files.

What the customer found was that both the linker and the `EDITBIN` approaches seemed to be fine with `/NXCOMPAT`. But the story was different with the `/DYNAMICBASE` flag.

Specifically, the linker approach was creating a larger file due to a new `.reloc` section. The file produced by the `EDITBIN` approach didn't have a `.reloc` section, and dumping the header reports **Relocations stripped**. Furthermore, when running `EDITBIN`, it generated the ominous warning message, "Warning LNK4259: '/DYNAMICBASE' is not compatible with '/FIXED'; image may not run." It appears that linking with the `/DYNAMICBASE` flag implicitly sets `/FIXED:NO`, but the `EDITBIN` command doesn't apply the same behavior, and it doesn't support the `/FIXED:NO` command line option.

The customer had the following questions:

> Are the linker approach and the the `EDITBIN` approach for enabling `/NXCOMPAT` equivalent?

Yes, enabling NX via the linker `/NXCOMPAT` flag is equivalent to enabling it with `EDITBIN`. In both cases, they set the `IMAGE_ DLLCHARACTERISTICS_ NX_ COMPAT` bit in the `IMAGE_ OPTIONAL_ HEADER`'s `DllCharacteristics`.

> Are the linker approach and the the `EDITBIN` approach for enabling `/DYNAMICBASE` equivalent?

The answer depends on what color glasses you're wearing. If you're wearing kernel-colored glasses, then yes, the two approaches are the same because both set the `IMAGE_DLL-CHARACTERISTICS_DYNAMIC_BASE` bit in the `IMAGE_OPTIONAL_HEADER`'s `Dll-Characteristics`. But in reality, they are not the same, because of the behavior noted above with respect to relocations. If you ask the linker for `/DYNAMICBASE`, it will default to `/FIXED:NO` because `/DYNAMICBASE` means "My base address can be moved around", which is the opposite of `/FIXED`, which means "My base address cannot change."

If you say `/FIXED`, then the linker does not generate "relocations", which are the bits of information that describe what adjustments need to be made to your DLL in order to make it happy at its new location. Trying to turn on `/DYNAMICBASE` with `EDITBIN` on a binary that is `/FIXED` doesn't work because `EDITBIN` doesn't know how to regenerate the `.reloc` section.

If you want to enable `/DYNAMICBASE`, then you cannot link with `/FIXED`.

> Are there any adverse consequences of mixing ASLR-enabled DLLs and non-ASLR-enabled DLLs in the same process?

There are no consequences beyond those already stated on the tin. The ASLR-enabled binaries will be subject to ASLR, and the non-ASLR-enabled binaries will not. You can mix and match freely within a process. Just be aware that the non-ASLR-enabled binaries will not be randomly relocated, which means that those binaries will not benefit from the security protections provided by ASLR.

> Are there any adverse consequences of mixing NX-enabled DLLs and non-NX-enabled DLLs in the same process?

The NX setting is process-wide, and the process takes its NX state from the `/NXENABLED` state of the executable, not from any DLLs. It's yet another one of the module flags that are meaningless for DLLs. So mix it up in the DLLs all you want. Nobody will care, because the flag is ignored for DLLs anyway.

> Are there any adverse consequences of mixing EXEs and DLLs which use different approaches for enabling NX and/or ASLR? For example, is there an unexpected interaction between an EXE which enabled ASLR with `EDITBIN` and a DLL that enabled ASLR with a linker switch? How can I tell whether an EXE or DLL has had its NX or ASLR bit set via the linker as opposed to the `EDITBIN` program?

There is no way to tell how the NX or ASLR attributes were enabled. Using the linker or `EDITBIN` both lead to the same binary. There's nothing that records who set the bit. As a result, it doesn't matter how you enabled NX and ASLR. The system behaves the same regardless of how you enabled them.

Bonus chatter: There is some text on MSDN (which got copied into a glossary on TechNet) which says

> For a component to support ASLR, all components that it loads must also support ASLR. For example, if A.exe consumes B.dll and C.dll, all three must support ASLR.

Nobody is quite sure how that text got into the documentation, since it's not true. One of the original authors of that article surmises that perhaps what they were trying to say was "In order for a process to take full advantage of ASLR, the executable and all DLLs must support ASLR," but somehow the message got garbled during editing.

Given what we already know about how the loader shares pages in the face of ASLR, having a DLL be subject to ASLR in some processes but not others would create extra work for the loader, because it now has to be able to keep up to two copies of every DLL in memory: A randomly-located version for ASLR, and a non-relocated version for non-ASLR. It's extra work for no real benefit, so I can understand why they don't do it.

[1] I can think of a few reasons why the customer cannot relink all of the files. One possibility is that they don't have the source code any more. Another is that they have the source code, but they don't have the build tools any more. (For example, it may be built with a very old compiler that doesn't work any more.) Or they have the source code, and they have the tools, but they simply don't want to take the risk that relinking the file might result in an unexpected change to the program. (For example, if it was linked with an older version of the linker, and they have since upgraded to a newer version.)

Raymond Chen

Follow