Don't trust the return address

devblogs.microsoft.com/oldnewthing/20040101-00

January 1, 2004

Raymond Chen

Sometimes people ask, “So I know how to get my return address [use the _ReturnAddress() intrinsic]; how do I figure out what DLL that return address belongs to?”

Beware.

Even if you figure out which DLL the return address belongs to [use GetModuleHandle-Ex(GET_MODULE_HANDLE_EX_FLAG_FROM_ADDRESS)], that doesn’t mean that that is actually the DLL that called you.

A common trick is to search through a “trusted” DLL for some code bytes that coincidentally match ones you (the attacker) want to execute. This can be something as simple as a “retd” instruction, which are quite abundant. The attacker then builds a stack frame that looks like this, for, say, a function that takes two parameters.

```
trusted_retd
hacked parameter 1
hacked parameter 2
hacker_code_addr
```

After building this stack frame, the attacker then jumps to the start of the function being attacked.

The function being attacked looks at the return address and sees `trusted_retd`, which resides in a trusted DLL. It then foolishly trusts the caller and allows some unsafe operation to occur, using hacked parameters 1 and 2. The function being attacked then does a “retd 8” to return and clean the parameters. This transfers control to the `trusted_retd`, which performs a simple `retd`, which now gives control to the `hacker_code_addr`, and the hacker can use the result to continue his nefarious work.

This is why you should be concerned if somebody says, “This code verifies that its caller is trusted...” How do they know who the caller really is?

Raymond Chen

Follow