If you pass invalid parameters, then all bets are off

devblogs.microsoft.com/oldnewthing/20080505-00

May 5, 2008

Raymond Chen

Alun Williams pointed out that if you pass invalid parameters to `DeferWindowPos`, it does not destroy the `HDWP`. Well, yeah, because if you pass invalid parameters, then all bets are off.

Different functions perform different degrees of parameter validation; the degree to which this is done is typically guided by security concerns. Information that crosses security boundaries must be fully-validated, whereas a call to an in-process function has very little in the way of security obligations with respect to invalid parameters, since a bad caller could just mess with the in-process function directly; no need to try to "trick" it with invalid parameters.

In practice, most functions that perform parameter validation go something like this:

```
SomeFunction(...)
{
 if (any parameter is invalid) {
   signal invalid parameter error in an appropriate manner
 } else {
   actually do something
 }
}
```

(In some cases, the validation code is not even written by a human being. Instead, there's a script that parses the header files and autogenerates the validation code.)

If there is an invalid parameter, the entire operation is typically abandoned. Because, after all, how can you expect a function even to get off the ground when it doesn't have all its parameters? I mean, how can the `DeferWindowPos` destroy the `HDWP` when it fails to validate its parameters, if the invalid parameter might be the `HDWP` ?

Regardless of the degree to which parameter validation occurs, if you pass invalid parameters, then (generally speaking) there are no guarantees. Passing valid parameters is part of the basic ground rules for programming. If you break your end of the deal, then the function is under no obligation to hold up its end.

Raymond Chen

Follow