Fumbling around in the dark and stumbling across the wrong solution

devblogs.microsoft.com/oldnewthing/20060613-05

June 13, 2006

Raymond Chen

I don't mean to pick on this series of entries, but it illustrates an interesting pattern of stumbling across the wrong "solution".

The series begins by attempting to trigger the system's monitor blank timeout by posting a message to the desktop window. As we saw earlier, the desktop window is a very special window and as a rule should be avoided, since it won't behave like windows created by applications. In particular, the author tried to post a message to the desktop window. This used to work in the historically open world of the window manager, but security and robustness concerns have come to take priority over compatibility. In Windows XP SP2, the desktop window resists being disabled because programs were doing it inadvertently, and it appears that the desktop also resists having messages posted to it. My guess is that this was done as a way to strengthen protection against shatter attacks. This did improve robustness and stability, but it also broke the article's dubious `PostMessage` hack.

Enter round three, wherein the author fumbled around for other windows the monitor blank timeout message could be posted to, and eventually the author found that posting the message to the mysterious window `HWND_TOPMOST = -1` seemed to do the trick.

I knew in the back of my mind that people developed software this way, but the hopeful part of my brain continued to wish that it was merely taking place in a fantasy world. Making up intentionally invalid parameters and seeing what happens falls into the category of malicious goofing around, not in the realm of software engineering and design. Even if you find something that seems to work, you certainly wouldn't design a product around it!

(Similarly, I've seen people ask questions like "What does message 49251 mean?" This is the reverse case: Seeing a made-up number and attempting to assign meaning to it. Message numbers starting at `0xC000` (decimal 49152) are messages registered via `RegisterWindowMessage`. The numerical value of the message associated with a registered window message is unpredictable and varies from desktop to desktop. The only guarantee is that it will remain consistent within a single desktop.)

If you look more carefully at what the author stumbled across, you'll see that the "solution" is actually another bug. It so happens that the numerical value `-1` for a window handle is suspiciously close to the value of `HWND_BROADCAST`:

```
#define HWND_BROADCAST  ((HWND)0xffff)
```

It so happens that internally, the window manager supports `(HWND)-1` as an alternative value for `HWND_BROADCAST`. (I leave you to speculate why.) As a result, what the author actually is doing is broadcasting the monitor power-off message to all top-level windows! As we saw before, broadcasting messages is a very dangerous business, and in this case, the author is just lucky that all the windows on the desktop interpret the message the same way, that it is safe to process the message multiple times, and none of the windows perform any special filtering for that message. (Another author stumbled across the same incorrect "solution" but didn't provide any insight into the process by which the result was arrived at. Yet another author sort of found some issues but didn't quite put them all together.)

For example, a presentation program might want to suppress monitor power-off when it is the foreground window by trapping the message and turning the monitor back on. If such a program happens to be running, broadcasting the power-off message to all top-level windows would turn off the monitor for all the windows that deferred to system default behavior, but when that presentation program received the message, it would turn the monitor back on. Now you're at the mercy of the order in which the windows process that broadcast message. When the presentation program processes the message, the monitor will turn back on, and if that program happens to be the last one to process the message (say, it got paged out and was slow to page back in), then the monitor will merely blink off and back on.

The correct solution is not to post messages to random windows. If you want the message to go through window message default processing, create a window and process it yourself. Don't try to trick some other window (or in this case, hundreds of other windows simultaneously) into doing it for you.

Raymond Chen

Follow