Beware of non-null-terminated registry strings

devblogs.microsoft.com/oldnewthing/20040824-00

August 24, 2004

Raymond Chen

Even though a value is stored in the registry as REG_SZ, this doesn't mean that the value actually ends with a proper null terminator. At the bottom, the registry is just a hierarchically-organized name/value database.

And you can lie and get away with it.

Lots of people lie about their registry data. You'll find lots of things that should be REG_DWORD stored as a four-byte REG_BINARY. (This is in part a holdover from Windows 95's registry, which didn't support REG_DWORD.)

One of the most insidious lies is to lie about the length of a string you're writing to the registry. Consider the following program:

```
#include <windows.h>
#include <stdio.h>
int __cdecl main(int argc, char **argv)
{
    RegSetValueExW(HKEY_CURRENT_USER, L"Scratch",
                   0, REG_SZ, (BYTE*)L"12", 2);
    DWORD cb = 0;
    RegQueryValueExW(HKEY_CURRENT_USER, L"Scratch",
                     NULL, NULL, NULL, &cb);
    printf("Size is %d bytes\n", cb);
    WCHAR sz[2];
    sz[0] = 0xFFFF;
    sz[1] = 0xFFFF;
    cb = sizeof(sz[0]);
    DWORD dwRc = RegQueryValueExW(HKEY_CURRENT_USER, L"Scratch",
                                  NULL, NULL, (BYTE*)sz, &cb);
    printf("RegQueryValueExW requesting %d bytes => %d\n",
           sizeof(sz), dwRc);
    printf("%d bytes required\n", cb);
    if (dwRc == ERROR_SUCCESS) {
        printf("sz[0] = %d\n", sz[0]);
        printf("sz[1] = %d\n", sz[1]);
    }
    RegDeleteValueW(HKEY_CURRENT_USER, L"Scratch");
    return 0;
}
```

If you run this program, you get this:

```
Size is 2 bytes
RegQueryValueExW requesting 4 bytes => 0
2 bytes required
sz[0] = 49
sz[1] = 65535
```

What happened?

First, observe that the call to RegSetValueExW lies about the length of the string, claiming that it is two bytes long when in fact it is six! (Two WCHARs plus a terminator.)

The registry dutifully records this lie and reports it back to subsequent callers.

The first call to RegQueryValueExW asks how big the string is, and the registry reports the value 2, since that's the value it was given when the value was originally stored.

To show that there really is no null terminator, we ask the registry to read those two bytes of data into our buffer, pre-filling the buffer with sentinel values so we can see what got updated and what didn't.

Lo and behold, the values were read from the registry and only two bytes were read. sz[0] contains the character '1', and sz[1] remains uninitialized.

This has security implications.

If your program assumes that strings in the registry are always null-terminated, then you can be tricked into a buffer overflow if you happen across a non-null-terminated string. (For example, if you use `strcpy` to copy it around.)

(Note: I'm not going to get into whether it should have been possible to get into this state in the first place. I didn't design the registry. Arguing about the past isn't going to change the present, and the present is that this is how it works so you'd better be ready for it.)

Exercise: Change the last parameter of RegSetValueExW to 3 and run the program again. Explain the results and discuss its consequences.

Raymond Chen

Follow