Why does SetFileValidData fail even though I enabled the SE_MANAGE_VOLUME_NAME privilege?

June 3, 2016

Raymond Chen

A customer reported that their call to `SetFileValidData` was failing with `ERROR_PRIVILEGE_NOT_HELD` despite the fact that they held the privilege whose name is defined by the symbol `SE_MANAGE_VOLUME_NAME`. (Note that the "name" in "manage volume name" doesn't mean that you are managing the name of the volume; rather it means "This is the name of the privilege for managing volumes.")

The customer was kind enough to reduce the problem to a simple program:

```c
#include <windows.h>

int main(int argc, char** argv)
{
 // This succeeds
 HANDLE h = CreateFileW(L"test", GENERIC_WRITE, 0, nullptr,
   CREATE_ALWAYS, FILE_FLAG_NO_BUFFERING | FILE_FLAG_OVERLAPPED,
   nullptr);

 LARGE_INTEGER newSize;
 newSize.QuadPart = 256 * 1024 * 10;
 // This succeeds
 SetFilePointerEx(h, newSize, nullptr, FILE_BEGIN);

 // This succeeds
 SetEndOfFile(h);

 HANDLE hToken;
 // This succeeds
 OpenProcessToken(GetCurrentProcess(),
     TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken);

 // This succeeds
 SetPrivilege(hToken, SE_MANAGE_VOLUME_NAME, TRUE);

 // This succeeds
 CloseHandle(hToken);

 // This fails with ERROR_PRIVILEGE_NOT_HELD
 SetFileValidData(h, newSize.QuadPart);

 CloseHandle(h);

 return 0;
}
```

The answer is hidden in the documentation for the `SetFileValidData` function:

> A caller must have the **SE_MANAGE_VOLUME_NAME** privilege enabled when opening a file initially.

The program didn't enable the manage volume privilege until after it had already created the file handle.

This requirement that the privilege be active at the point the handle is created becomes less surprising when you realize that the general policy for kernel object security is that security is checked at the time handles are created, rather than when they are used.

Raymond Chen

Follow