How do I perform shell file operations while avoiding shell copy hooks?

devblogs.microsoft.com/oldnewthing/20120330-00

March 30, 2012

Raymond Chen

Okay, the subject line of the article gives away the answer to the puzzle, but here's the puzzle anyway: A customer reported a problem with the `SHFileOperation` function:

> Consider the following program:
>
> ```
> #include <windows.h>
> #include <shellapi.h>
> int main()
> {
> SHFILEOPSTRUCT fileStruct = {};
>     fileStruct.wFunc = FO_RENAME;
>     fileStruct.pFrom = L"C:\\a\0";
>     fileStruct.pTo   = L"C:\\b\0";
>     fileStruct.fFlags= FOF_NO_UI;
>     ::SHFileOperation(&fileStruct);
>     return 0;
> }
> ```
>
> If "a" is a file, then everything works fine, but if it's a directory, then Application Verifier raises the following error:
>
> > Heap violation detected
> > Memory access operation in the context of a freed block: reuse-after-delete or double-delete
>
> Can you help explain what we're doing wrong? So far as we can tell, all our parameters are correct.

This is one of those "It doesn't work on my machine" issues, because the provided sample program runs fine on a freshly-installed copy of Windows. We asked the customer to send us a crash dump file, and from that crash dump the source of the problem was obvious:

```
eax=00000001 ebx=00000000 ecx=73d34c58 edx=00270001 esi=09fa2ff8 edi=00000000
eip=10001131 esp=0026dea8 ebp=0026df24 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010200
Contoso+0x1131:
10001131 8b4604          mov     eax,dword ptr [esi+4] ds:002b:09fa2ffc=????????
0:000> k
  *** Stack trace for last set context - .thread/.cxr resets it
ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
0026df24 75a3554a Contoso+0x1131
0026df64 75a1b07c ole32!ActivationPropertiesIn::DelegateCreateInstance+0x108
0026dfb8 75a1aff1 ole32!CApartmentActivator::CreateInstance+0x112
0026dfd8 75a1ae16 ole32!CProcessActivator::CCICallback+0x6d
0026dff8 75a1adc7 ole32!CProcessActivator::AttemptActivation+0x2c
0026e034 75a1b0df ole32!CProcessActivator::ActivateByContext+0x4f
0026e05c 75a3554a ole32!CProcessActivator::CreateInstance+0x49
0026e09c 75a352ce ole32!ActivationPropertiesIn::DelegateCreateInstance+0x108
0026e2fc 75a3554a ole32!CClientContextActivator::CreateInstance+0xb0
0026e33c 75a35472 ole32!ActivationPropertiesIn::DelegateCreateInstance+0x108
0026eb18 75a45916 ole32!ICoCreateInstanceEx+0x404
0026eb78 75a45877 ole32!CComActivator::DoCreateInstance+0xd9
0026eb9c 75a45830 ole32!CoCreateInstanceEx+0x38
0026ebcc 75c61fe0 ole32!CoCreateInstance+0x37
0026ee64 75c61354 shell32!_SHCoCreateInstance+0x1ac
0026ee88 75c1b904 shell32!SHExtCoCreateInstance+0x1e
0026eec8 75becbcf shell32!SHExtCoCreateInstanceString+0x43
0026f10c 75beca76 shell32!CreateCopyHooks+0xe1
0026f344 75bec9da shell32!CallCopyHooks+0x4b
0026f370 75bec95b shell32!CallFileCopyHooks+0x29
0026f5bc 75bec8a4 shell32!CFileOperation::CopyHooks+0x119
0026fa3c 75c37955 shell32!CCopyWorkItem::_UpFrontConfirmations+0xb7
0026fc6c 75c378b0 shell32!CCopyWorkItem::ProcessWorkItem+0x83
0026fca0 75c37fda shell32!CRecursiveFolderOperation::Do+0x1d5
0026fce4 75c39a19 shell32!CFileOperation::_EnumRootDo+0x14e
0026fd4c 75c397b9 shell32!CFileOperation::PrepareAndDoOperations+0x27f
0026fd74 75c396c7 shell32!SHFileOperationWithAdditionalFlags+0xe9
```

The crash is in some third party component named Contoso, which is running because it is being `CoCreate`'d. The call came from `CreateCopyHooks`, and it doesn't require very much in the way of psychic powers to conclude that the shell is creating the Contoso object because it registered as a copy hook.

This also explains why the problem occurs only on the customer's machine: The customer installed the Contoso shell extension and we didn't.

Okay, so the problem is that the Contoso shell extension has a use-after-free memory corruption bug. (Some Web searching revealed that a lot of people had encountered problems with the Contoso shell extension.)

The `FOFX_NOCOPYHOOKS` flag comes in handy here. Setting this extended flag disables copy hooks for your file operation. Extended flags cannot be passed to the classic `SHFileOperation` function because the `SHFILEOPSTRUCT` structure uses a 16-bit `WORD` for the `fFlags` member, but the `FOFX_NOCOPYHOOKS` flag has the numerical value `0x00800000` which doesn't fit in a 16-bit integer. (The "X" at the end of the prefix is another clue.) The way to set extended flags is to use the `IFileOperation` interface.

```
// Just for fun, I'll use ATL templates instead of raw C++.
HRESULT RenameAtoB()
{
 HRESULT hr;
 CCoInitialize init;
 hr = init;
 if (FAILED(hr)) return hr;
 CComPtr<IFileOperation> spfo;
 hr = spfo.CoCreateInstance(CLSID_FileOperation);
 if (FAILED(hr)) return hr;
 hr = spfo->SetOperationFlags(FOFX_NOCOPYHOOKS);
 if (FAILED(hr)) return hr;
 CComPtr<IShellItem> spsi;
 hr = SHCreateItemFromParsingName(L"C:\\a", NULL,
                          IID_PPV_ARGS(&spsi));
 if (FAILED(hr)) return hr;
 hr = spfo->RenameItem(spsi, L"b", NULL);
 if (FAILED(hr)) return hr;
 hr = spfo->PerformOperations();
 if (FAILED(hr)) return hr;
 return S_OK;
}
```

Raymond Chen

Follow