When you submit a security vulnerability report, we go the extra mile and try to fix your typos

devblogs.microsoft.com/oldnewthing/20170324-00

March 24, 2017

Raymond Chen

A security vulnerability report arrived that went like this:

> 1. Create the folder `C:\Folder` and grant full control to authenticated users.
> 2. Create the subfolder `C:\Folder\bar`.
> 3. Create the files `C:\Folder\bar\foo` and `C:\Folder\foo`.
> 4. Deny all permissions to everyone for those two files.
> 5. Set the owner of the two files to a domain administrator or some other user.
>
> Even though all access has been revoked to the two files, the following program deletes `C:\Folder\foo`:
>
> ```
> #include <windows.h>
>
> int main(int, char**)
> {
>  DeleteFileW(L"C:\\Folder\\Bar\foo");
>  DeleteFileW(L"C:\\Folder\\foo");
>  return 0;
> }
> ```
>
> From reading this page, it seems that file deletion is related to the control on the file itself and not on the parent folder. And anyway, both have the same rights.

There's nothing wrong here. Even though all access to the file has been revoked, all authenticated users still have `DELETE_CHILD` permission on the parent, and as we noted some time ago, that is sufficient access to delete any file in the directory. This is also noted in the documentation for `DeleteFile`:

> To delete or rename a file, you must have either delete permission on the file, or delete child permission in the parent directory.

The page cited by the finder lists the various permissions available to files and folders. It even says

> If a user has full control over a folder, the user can delete files in the folder regardless of the permission on the files.

That statement covers the exact scenario described here: The permissions on the parent folder grant full control, which includes `DELETE_CHILD`. Mind you, that statement could be strengthened by weakening the hypothesis and strengthening the conclusion:

> If a user has permission to delete children of a folder, the user can delete files and subfolders in the folder regardless of the permission on the files or subfolders.

Table 13-3 reiterates that *Full Control* on a folder grants permission to delete files and subfolders.

So everything is behaving as normal, and the security team replied, "Upon investigation we have determined that this is not a valid vulnerability as you are an authenticated user, or providing authenticated access to a potential attackers to reproduce this report." In other words, "You gave authenticated users full control, so it's not a vulnerability that any authenticated user has full control."

What made this interesting to me is that the finder posted to Stack Overflow wondering why this wasn't a bug. And then the finder eventually discovered a typographical error in their sample program: The first `DeleteFile` call passes the string `L"C:\\Folder\\Bar\foo"` instead of `L"C:\\Folder\\Bar\\foo"`. "I'm not sure why that was not Microsoft's answer."

Okay, let's look back at the situation. The finder appeared to be concerned about two things but articulated only one of them in the report.

1. The user is able to delete the file `C:\Folder\foo`.
2. The user is not able to delete the file `C:\Folder\bar\foo` even though its permissions are identical to `C:\Folder\foo`.

The finder raised only the first question, and that's the question the security team answered.

That said, the security team tries to be thorough and in this case assumed that the finder created the `C:\Folder\bar\foo` file for a reason, namely in order to delete it. And if they fix the obvious typo in the program, the file does get deleted. So it seems that the deletion is following the security model in all cases. The finder never said, "But the `C:\Folder\bar\foo` file is not deleted," so the security team assumed that the finder was expecting the file to be deleted, and it was.

The security team didn't mention the typo because they assumed that it was just a transcription error. In general, the security team gives the finder all benefit of the doubt and assume that they are dealing with an experienced programmer who understands the security model. Any misunderstandings are assumed to be due to communication problems or poor explanations. These sorts of things happen a lot with legitimate security issues because it is

common to receive vulnerability reports from people for whom English is not their native language. You don't want to reject an issue just because you can't understand it. And you definitely don't want to reject an issue just because there was a typo in the report.

In a sense, the security team failed this particular finder because they assumed *too much* from the finder.[1]

Sorry.

But you don't want to take the default position that the finder simply doesn't understand how the system works, because that biases you toward rejecting issues just because you can't understand them when they are initially presented to you.

[1] It's like asking a question at a physics symposium: The speaker is going to assume that you're a physicist (or at least well-versed enough in physics to understand the proceedings at a physics symposium), and they will fix the obvious errors in your question, assuming that you misspoke or were nervous. But if you're not a physicist, then those automatic corrections might end up confusing you even more, because they are now answering a question different from the one you asked.

Raymond Chen

Follow