Exploring the problem: Create a file that only one process can write to

devblogs.microsoft.com/oldnewthing/20171013-00

October 13, 2017

Raymond Chen

A customer liaison explained that they had a customer who wanted to create a file that only one process can write to. The customer has a program that writes important information to a file, and they want to prevent the user from modifying that file. The program is running under the credentials of the logged-in user, so they cannot deny write access to the file, because that would prevent the program itself from being able to write to it. They considered locking the file by denying sharing, but that would be effective only while the program is running.

This is a difficult position right off the bat, because permissions belong to users, not to processes. Since the program is running under the user's credentials, the user has full control over the process and can access to the sensitive file by stealing the file handle out of the process. Any solution would therefore require the involvement of more than one set of user credentials.

One solution is to have a service. The sensitive file is accessible only to the account under which the service runs. The application contacts the service, and it is the service which writes the data to the file.

Mind you, this is still vulnerable: The user can attack the program and manipulate the parameters that it passes to the service. The service cannot trust the data received from the program because the program could be passing false data.

The customer liaison explained that the customer wants to prevent the end users from tampering with the program's log file. The program is monitoring employee activities, and the customer has found that in many cases, employees report issues with the program, and when they debug the problem, they discover that the log files have been tampered with. To prevent tampering, they want the file to be writable only by the program generating the log.

I noted that the fact that the employee is tampering with the log file should be a significant data point in building a case against them. After all, if the user doesn't want the information to be logged, they can reformat the hard drive that contains the log file.

The customer liaison thanked us for our feedback and reported that the customer decided to use a separate account for accessing the log file. To avoid the complexity of a Windows service, the customer is simply having the program temporarily impersonate the special account, write the data to the log file, and then stop impersonating. The password for the special account is stored in an encrypted configuration file, which is how they are currently storing the password to their database.

Okay, now you've created a system that would never pass a security review.

You think you're so smart by encrypting the password, but that doesn't add any security because the program itself must be able to decrypt it. An attacker can simply set a breakpoint in the program right after the code that decrypts the password, and now they have the password in the clear. With this password, they can not only manipulate their log files, they can also manipulate the log files of *other users*. Your original problem was a data tampering security vulnerability, but by giving the user the password to the special account, you added spoofing (the user can impersonate the special user and do *anything* that special user can do), information disclosure (obtaining access to log files for other users), and denial of service (locking the log file and preventing anybody else from accessing it).

And in fact, they have this insecure system already in production, since they admitted that they are already using this technique to record the password to their database.

The customer liaison thanked us for pointing this out and will advise the custom of these additional issues.

Another approach that doesn't involve a service is to use the system event log. The program can write entries to a custom event log, and you get to take advantage of existing infrastructure to collect logs across your organization.

We never did find out what the customer ended up doing. But we hope it wasn't the thing about putting the password in an encrypted file (and giving everybody the decryption key).

Raymond Chen

Follow