It rather involved being on the other side of this airtight hatchway: Reading the user's saved passwords

devblogs.microsoft.com/oldnewthing/20230206-00

February 6, 2023

Raymond Chen

A security vulnerability report claimed that a particular program did not store its saved passwords in the Windows Credential Manager securely:

> Microsoft's Contoso program does not encrypt user passwords before saving them in the Windows Credential Manager. As a result, any program that runs with the user's credentials can read the passwords. The Contoso program should save encrypted passwords in the Credential Manager.

There was another report that put the blame on Credential Manager:

> Credential Manager should require administrator privileges to read a user's saved credentials.

The perceived issue here is that one program can read the passwords saved by another program, and that other program should encrypt its paswords so that they can be decrypted only by that program. Rogue programs enumerating the contents of the user's saved credentials would be stymied because they don't know how to decrypt the data.

But if you have gained the ability to execute code in the context of the victim, then you've already won. You can do anything the victim can do!

It's like saying, "I am a bad guy from a 1960's spy movie. I have successfully hypnotized the victim into obeying all of my commands, while nevertheless behaving perfectly normally. I tell the victim to go to the bank, withdraw all their money, and bring it to me. This is a security flaw in the bank. It should not allow hypnotized people to withdraw money!"

The bank did its job, which is to confirm the identity of the person withdrawing the money. The person at the counter did nothing to draw suspicion, and all the paperwork checked out, so they got their money.

The system sees that there is a process running as the user, and that process is asking for the password that the user had saved earlier. Now, certainly, users are permitted to access the passwords that they had saved (that being the point of saving the password), and the

Credential Manager is correct in returning those saved passwords to that user. The information is not being disclosed to other users: Users can access their own saved passwords, but they cannot access the saved passwords of other users.

Encrypting the data before putting it in the password cache sounds like it would stop an attacker, but it doesn't. Since the original program must be able to decrypt the data, the attacker can analyze the original program and re-run the decryption function. In the attacker is being lazy, they can just *run the original program* and set a breakpoint immediately after it decrypts the password.

You might think that you could protect yourself from hypnosis-induced bank withdrawal by telling the bank, "Before giving me any money, make sure to ask me this secret security question." But that doesn't help, because even though you've been hypnotized, you still know the answer to the secret security question. The hypnotized-you goes to the bank, the bank teller asks the security question, you answer it, get your money, and give it to the 1960's evil bad guy.

And of course requiring administrator privileges to retrieve your own saved passwords would be a non-starter. Every time you want to use a saved password, you have to call your administrator to get it for you?

Allowing users to retrieve their own saved passwords is perfectly normal behavior. No security boundary is crossed, and the information is disclosed only to the user to put the information there in the first place.

If you let someone run arbitrary programs under your identity, you have handed over control of everything tied to your identity, and that includes your saved passwords.

Bonus bogus vulnerability: Suppose you create a shortcut to the command line `runas /savecred /user:.\administrator someprogram.exe`, and you run the shortcut, and an administrator comes over and types their password when prompted. A security vulnerability report claimed that there is an elevation of privilege vulnerability because an attacker can edit the shortcut to read `runas /savecred /user:.\administrator cmd.exe` and gain an administrative command prompt.

That's true, but that's also the whole point of `/savecred`.

The `/savecred` option means "Use the saved credentials from the Credential Manager, or prompt for credentials (and save them in the Credential Manager)." They are not saved in the shortcut at all. Not that saving them in the shortcut is even an option, because you can run `runas` from a command prompt, and in that case there's no shortcut.

If you call over an administrator and get them to type their password into your system, then you found yourself a gullible administrator. In this case, what you tricked them into doing is adding their password to your password cache. It's like calling an administrator to your Web browser and asking them to type their password into a Web site. That password is going to be saved in to *your* Web browser password cache, and you can now extract it from that cache and reuse it for anything you like.

The point of `/savecred` is to let you save your own credentials so you don't have the hassle of typing them over and over into places you use often. If you can trick an administrator into putting their password into your saved credential cache, then more power to you (and shame on the administrator).

Don't be a gullible administrator.