What is the default security descriptor?

devblogs.microsoft.com/oldnewthing/20040312-00

March 12, 2004

Raymond Chen

All these functions have an optional LPSECURITY_ATTRIBUTES parameter, for which everybody just passes NULL, thereby obtaining the default security descriptor. But what **is** the default security descriptor?

Of course, the place to start is MSDN, in the section titled Security Descriptors for New Objects.

It says that the default DACL comes from inheritable ACEs (if the object belongs to a hierarchy, like the filesystem or the registry); otherwise, the default DACL comes from the primary or impersonation token of the creator.

But what is the default primary token?

Gosh, I don't know either. So let's write a program to find out.

```
#include <windows.h>
#include <sddl.h> // ConvertSecurityDescriptorToStringSecurityDescriptor
int WINAPI
WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
 HANDLE Token;
 if (OpenProcessToken(GetCurrentProcess(), TOKEN_QUERY, &Token)) {
 DWORD RequiredSize = 0;
 GetTokenInformation(Token, TokenDefaultDacl, NULL, 0, &RequiredSize);
 TOKEN_DEFAULT_DACL* DefaultDacl =
     reinterpret_cast<TOKEN_DEFAULT_DACL*>(LocalAlloc(LPTR, RequiredSize));
 if (DefaultDacl) {
  SECURITY_DESCRIPTOR Sd;
  LPTSTR StringSd;
  if (GetTokenInformation(Token, TokenDefaultDacl, DefaultDacl,
                          RequiredSize, &RequiredSize) &&
      InitializeSecurityDescriptor(&Sd, SECURITY_DESCRIPTOR_REVISION) &&
      SetSecurityDescriptorDacl(&Sd, TRUE,
          DefaultDacl->DefaultDacl, FALSE) &&
      ConvertSecurityDescriptorToStringSecurityDescriptor(&Sd,
          SDDL_REVISION_1, DACL_SECURITY_INFORMATION, &StringSd, NULL)) {
   MessageBox(NULL, StringSd, TEXT("Result"), MB_OK);
   LocalFree(StringSd);
  }
  LocalFree(DefaultDacl);
 }
 CloseHandle(Token);
 }
 return 0;
}
```

Okay, I admit it, the whole purpose of this entry is just so I can call the function ConvertSecurityDescriptorToStringSecurityDescriptor, quite possibly the longest function name in the Win32 API. And just for fun, I used the NT variable naming convention instead of Hungarian.

If you run this program you'll get something like this:

```
D:(A;;GA;;;S-1-5-21-1935655697-839522115-854245398-1003)(A;;GA;;;SY)
```

Pull out our handy reference to the Security Descriptor String Format to decode this.

- "D:" – This introduces the DACL.
- "(A;;GA;;;S-...)" – "Allow" "Generic All" access to "S-...", which happens to be me. Every user by default has full access to their own process.
- "(A;;GA;;;SY)" – "Allow" "Generic All" access to "Local System".

Next time, I'll teach you how to decode that S-... thing.

Raymond Chen

Follow