I called AdjustTokenPrivileges, but I was still told that a necessary privilege was not held

devblogs.microsoft.com/oldnewthing/20190531-00

May 31, 2019

Raymond Chen

A customer had a service running as Local System and wanted to change some token information. The information that they wanted to change required `SeTcbPrivilege`, so they adjusted their token privileges to enable that privilege, but the call still failed with `ERROR_PRIVILEGE_NOT_HELD`: "A required privilege is not held by the client."

Here's sketch of their code. All function calls succeed except the last one.

```
HANDLE processToken;
OpenProcessToken(GetCurrentProcess(), TOKEN_ALL_ACCESS,
    &processToken);

HANDLE newToken;
DuplicateTokenEx(processToken, TOKEN_ALL_ACCESS, nullptr,
    SECURITY_MAX_IMPERSONATION_LEVEL, TokenPrimary, &newToken);

TOKEN_PRIVILEGES privileges;
privileges.PrivilegeCount = 1;
privileges.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
LookupPrivilegeValue(nullptr, SE_TCB_NAME,
    &privileges.Privileges[0].Luid);

AdjustTokenPrivileges(newToken, FALSE, &privileges, 0,
    nullptr, nullptr);

DWORD sessionId = ...;
SetTokenInformation(newToken, TokenSessionId, &sesionId,
    sizeof(sessionId)); // FAILS!
```

This fails because we adjusted the privileges of the wrong token!

The TCB privilege needs to be enabled on the token that is performing the operation, not the token that is the target of the operation. Because you need privileges to *do things*, not to *have things done to you*.

The security folks explained that the correct order of operations is

1. `ImpersonateSelf()`.
2. `OpenThreadToken()`.
3. `AdjustTokenPrivileges(threadToken)`.
4. Do the thing you wanna do. (In this case, duplicate the token and change the session ID.)
5. Close the thread token.
6. `RevertToSelf()`.

The overall sequence therefore goes like this:

```
void DoSomethingAwesome()
{
 if (ImpersonateSelf(SecurityImpersonation)) {
  HANDLE threadToken;
  if (OpenThreadToken(GetCurrentThread(), TOKEN_ALL_ACCESS,
                      TRUE, &threadToken)) {
   TOKEN_PRIVILEGES privileges;
   privileges.PrivilegeCount = 1;
   privileges.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
   if (LookupPrivilegeValue(nullptr, SE_TCB_NAME,
                            &privileges.Privileges[0].Luid)) {
    if (AdjustTokenPrivileges(newToken, FALSE, &privileges, 0,
                              nullptr, nullptr)) {
     // Now do the thing you wanna do.
     HANDLE newToken;
     if (DuplicateTokenEx(threadToken, TOKEN_ALL_ACCESS, nullptr,
                          SECURITY_MAX_IMPERSONATION_LEVEL,
                          TokenPrimary, &newToken)) {
      DWORD sessionId = ...;
      if (SetTokenInformation(newToken, TokenSessionId,
                              &sesionId, sizeof(sessionId))) {
       // Hooray
      }
      CloseHandle(newToken);
     }
    }
   }
   CloseHandle(threadToken);
  }
  RevertToSelf();
 }
}
```

Of course, in real life, you probably would use RAII types to ensure that handles get closed and to remember to `RevertToSelf()` after a successful `ImpersonateSelf()`.

Raymond Chen

Follow