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

devblogs.microsoft.com/oldnewthing/20211126-00

November 26, 2021

Raymond Chen

A customer had a service 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."

They followed the cookbook from the last time we tried to address this problem. All calls succeed except the last one.

```
ImpersonateSelf(SecurityImpersonation);

HANDLE threadToken;
OpenThreadToken(GetCurrentThread(), TOKEN_ALL_ACCESS,
                TRUE, &threadToken);

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

TOKEN_PRIVILEGES changedPrivileges;
AdjustTokenPrivileges(threadToken, FALSE, &privileges,
                      sizeof(changedPrivileges),
                      &changedPrivileges,
                      nullptr);

if (changedPrivileges.PrivilegeCount == 0)
{
  // No net changes to privileges occurred,
  // so we must have had TCB already.
  Log("Already had TCB privilege");
}

// Now do the thing that requires TCB privilege.

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

DWORD sessionId = ...;
SetTokenInformation(newToken, TokenSessionId,
                    &sessionId, sizeof(sessionId));
// ^^ this fails
```

What's going on?

There is a horrible non-obvious quirk of the `AdjustTokenPrivileges` function that is tripping us up: The function returns success *even though it may have failed to do what you asked*.

The function "succeeded" in the sense that it successfully attempted to adjust the privileges you requested, and it successfully reported the result of the adjustment attempt. But that doesn't mean that the attempt actually accomplished what you asked it to do.

> If the function succeeds, the return value is nonzero. To determine whether the function adjusted all of the specified privileges, call **GetLastError**, which returns one of the following values when the function succeeds:
>
> | Return code | Description |
> | --- | --- |
> | **ERROR_SUCCESS** | The function adjusted all specified privileges. |
> | **ERROR_NOT_ALL_ASSIGNED** | The token does not have one or more of the privileges specified in the *NewState* parameter. The function may succeed with this error value even if no privileges were adjusted. The *PreviousState* parameter indicates the privileges that were adjusted. |

Emphasis mine.

Therefore, we must also check whether all of the requested privileges were actually adjusted.

```
TOKEN_PRIVILEGES changedPrivileges;
if (!AdjustTokenPrivileges(threadToken, FALSE, &privileges,
                           sizeof(changedPrivileges),
                           &changedPrivileges,
                           nullptr)) {
  clean up and fail;
}

if (GetLastError() == ERROR_NOT_ALL_ASSIGNED)
{
  clean up and fail;
}
```

This case is simple because we are adjusting only one privilege. If we were adjusting more than one privilege, then we would have to call `AdjustTokenPrivileges` a second time with the `changedPrivileges` to restore the token's privileges to their original state. If you forget to do this, you end up returning with a token that has extra privileges enabled, which could be a security issue.

```
if (GetLastError() == ERROR_NOT_ALL_ASSIGNED)
{
  // Need to clean up the partially-adjusted token.
  AdjustTokenPrivileges(threadToken, FALSE, &changedPrivileges,
                        sizeof(changedPrivileges), nullptr, nullptr);
  clean up and fail;
}
```

Raymond Chen

Follow