Confusing gotcha: PSECURITY_DESCRIPTOR is not a pointer to a SECURITY_DESCRIPTOR

devblogs.microsoft.com/oldnewthing/20151223-00

December 23, 2015

Raymond Chen

There is a structure called a `SECURITY_DESCRIPTOR`. It describes the layout of an absolute security descriptor.

There is also a structure called a `SECURITY_DESCRIPTOR_RELATIVE`. It describes the layout of a relative security descriptor.

And then there is a type called `PSECURITY_DESCRIPTOR`. You might think based on its name that it is a pointer to a `SECURITY_DESCRIPTOR`. But it's not. It is defined as

```
typedef PVOID PSECURITY_DESCRIPTOR;
// equivalent to
// typedef void *PSECURITY_DESCRIPTOR;
```

Most code that accept security descriptors don't care whether the security descriptor is absolute or relative. They just pass the security descriptor through to functions like `Access-Check`. And the name for a generic "pointer to some type of security descriptor, maybe relative, maybe absolute" is `PSECURITY_DESCRIPTOR`.

You rarely notice this switcheroo because code that deals with security descriptors typically use helper functions to do the heavy lifting. You notice this problem if you try to use something like `std::unique_ptr` to manage the lifetime of a security descriptor.

```
template<typename T>
struct LocalAlloc_delete
{
 LocalAlloc_delete() { }
 void operator()(T* p) throw() { LocalFree(p); }
};

template<typename T>
using unique_localptr = std::unique_ptr<T, LocalAlloc_delete<T>>;

void some_function()
{
 PSECURITY_DESCRIPTOR result;
 ConvertStringSecurityDescriptorToSecurityDescriptorW(
  L"O:AOG:DAD:(A;;RPWPCCDCLCSWRCWDWOGA;;;S-1-0-0)",
  SDDL_REVISION_1, &result, nullptr);

 // compiler error here
 unique_localptr<SECURITY_DESCRIPTOR> sd(result);

 .. do stuff with sd ...
}
```

The compiler complains because `result` is a `PSECURITY_DESCRIPTOR`, but it expects a `SECURITY_DESCRIPTOR*`.

I can't think of a clean way out of this. Here are some ugly ways out:

```
// ugly option 1 - cast it away
unique_localptr<SECURITY_DESCRIPTOR>
    sd(reinterpret_cast<SECURITY_DESCRIPTOR*>(result));

// ugly option 2 - special knowledge about PSECURITY_DESCRIPTOR
unique_localptr<void> sd(result);

// ugly option 3 - general, but an awful lot of typing
 unique_localptr<
    std::remove_pointer<PSECURITY_DESCRIPTOR>::type> sd(result);
```

In retrospect, the structure for an absolute security descriptor should have been named `SECURITY_DESCRIPTOR_ABSOLUTE`. My guess is that the name is historical: Initially, the only kind of security descriptor was absolute. Later, relative security descriptors were invented, and the easiest way to retrofit them into the existing interfaces was to make `PSECURITY_DESCRIPTOR` the generic security descriptor pointer.

Raymond Chen

Follow