SIDs are really just another a fancy way of creating unique IDs in a decentralized way

devblogs.microsoft.com/oldnewthing/20230613-00

June 13, 2023

Raymond Chen

A customer stumbled across a SID in an access control entry: `S-1-15-3-1024-1365790099-2797813016-1714917928-519942599-2377126242-1094757716-3949770552-3596009590`. They found the documentation on SIDs, which told them that given a SID of the form `S-R-I-S...`, the R is the revision ID, the I is the identifier authority value, and the S is a list of subauthorities. Applying this to the mystery SID told them that

- R = 1, this is a revision 1 SID.
- I = 15, the authority is 15.
- S = 3-1024-…, subauthorities.

The customer wanted to know what subauthorities this access control entry is granting access to. They couldn't find this SID in the list of well-known SIDs.

Access control entries grant access to SIDs. They don't grant access to subauthorities. The subauthority is the entity that issues the SID, but that doesn't mean that the SID grants access to the subauthority.

Suppose you find a list of "Phone numbers which are allowed to call into this conference call." Now, in many jurisdictions, there is some structure to telephone numbers that give you information about what kind of number it is and who issued it. You take one of those phone numbers, subject it to analysis, and conclude that the phone number was issued by Contoso Telecommunications. Does that mean that Contoso Telecommunications has access to your conference calls? No. All it tells you is that the phone that is allowed to call into your conference call was issued by Contoso Telecommunications. The power to join the conference call belongs to the phone number, not to the phone number's issuer.

Similarly, when you grant access to a SID, you are granting access to anybody who possesses that SID in their token, but that doesn't mean that that the subauthority which issued that SID is granted access. The subauthority is merely the entity that produced the SID. Once they generate it, they don't have any more power over that SID than anybody else.

But really, all this authority/subauthority nonsense is beside the point. The use of authorities and subauthorities is an implementation detail of SIDs.

The point of SIDs is that they are a unique ID to identify an entity that can be granted (or denied) access.

It so happens that authorities and subauthorities are a mechanism for how the system ensures that no two SIDs are the same. But that's not really relevant to what SIDs are used for, which is to identify entities which can be granted access. What authorities and subauthorities do is allow entities to carve up the SID space and say, "Okay, I'm going to let you create new SIDs in this little corner of the SID namespace. That corner is all yours. Create SIDs there any way you like."

Let's go back to the SID that this customer is trying to understand.

We see that the SID begins S-1-15-3-1024-…, and if you look in `winnt.h`, you can see that authority 15 is assigned to `SECURITY_APP_PACKAGE_AUTHORITY` subauthority 3 is assigned to `SECURITY_CAPABILITY_BASE_RID`, and sub-subauthority 1024 is assigned to `SECURITY_CAPABILITY_APP_RID`. If you have a good memory, you may recall that I discussed this corner of the SID space in the past. These SIDs are app capability SIDs, and the numbers that come after the 1024 are the decimal representation of eight 32-bit values that together form the SHA256 hash of the capability name.

Bonus chatter: One commenter noted that it would be great if Process Explorer could resolve these hash-based capability SIDs into the original name. Since these are SHA256 hashes, you cannot reasonably expect Process Explorer (or anybody else) to be able to recover the original name from an arbitrary hash. If you could do that, you would use that power to do much more lucrative things than recovering app capability names! At best, Process Explorer could have a hard-coded list of app capability SIDs and their corresponding capability names. But that list is never going to be comprehensive because anybody can make up a new one by calling `DeriveCapabilitySidsFromName`.