What are these SIDs of the form S-1-15-2-xxx?

devblogs.microsoft.com/oldnewthing/20220502-00

May 2, 2022

Raymond Chen

If you are the curious sort of person who looks at security descriptors, you may find SIDs of the form S-1-15-2-xxx. What are these things?

SIDs of the form S-1-15-2-xxx are app container SIDs. These SIDs are present in the token of apps running in an app container, and they encode the app container identity.

According to the rules for Mandatory Integrity Control, objects default to allowing write access only to medium integrity level (IL) or higher. App containers run at low IL, so they by default don't have write access to such objects.

An object can add access control entries (ACEs) to its access control list (ACL) to grant access to low IL. There are a few security identifiers (SIDs) you may see when an object extends access to low IL.

S-1-15-2-1: All Application Packages (SDDL abbreviation: "AC")

Packaged apps, also known as Universal Windows apps. This covers all classic Windows Store apps.

I've seen some web pages where people speculate that "All Application Packages" is some sort of malware. It's not. It's just a security group.

S-1-15-2-2: All Restricted Application Packages

I'm not sure what this means or how it differs from All Application Packages.

S-1-15-2-x1-x2-x3-x4-x5-x6-x7: Specific app

This is a SID for one specific app. The seven numbers that come after the S-1-15-2 are 32-bit decimal numbers that collectively represent the first 28 bytes of the SHA256 hash of the app package family name.

You can ask the system to calculate this SID for you by calling `DeriveAppContainerSid-FromAppContainerName`:

```
PSID sid;
if (SUCCEEDED(DeriveAppContainerSidFromAppContainerName(
        L"Contoso.Deluxe_yda3mdg2t4ngp", &sid)))
{
    // you have the sid in "sid"
}
```

You use this SID to grant access to a resource for one specific app. Of course, if you want to allow multiple apps, you can add multiple ACEs, one for each SID you want to allow.

Since the value comes from a cryptographic hash, there is no known way to reverse the SID back to the original package family name.[1] If you find one of these and are curious which app it applies to, you could enumerate all the apps that are installed on the system and feed each one into `DeriveAppContainerSidFromAppContainerName` looking for a match. But even then, you may not find it, because the SID may refer to an app that isn't currently installed: It may have been applied in anticipation of installing an app, or it may have been applied at a time when the app was present, but you have since uninstalled it.

There's another family of SIDs related to app containers that you may also run across. We'll look at them next time.

[1] If you find a way to do that, you'll probably use your new discovery to do much more lucrative things than reversing package family names.

Raymond Chen

Follow