If I'm going to store a SID in a file, should I store the string form or the binary form?

devblogs.microsoft.com/oldnewthing/20160122-00

January 22, 2016

Raymond Chen

A customer needed to save some user SIDs into a file and wanted guidance on whether to use string format or binary format.

Here are some pros and cons.

String format: To save the SID, use the `ConvertSidToStringSid` function, then save the string to the file. To load the SID, read the string from the file, then call `ConvertStringSid-ToSid`. The conversion may fail if the string is corrupted (either accidentally or maliciously).

- Pro: Hard to get wrong. The heavy lifting is done by the two helper functions. In particular a corrupted string SID will be detected by `ConvertStringSidToSid`.
- Pro: Strings are well-known quantities. You probably already have code to load and save strings. They are also easy to see in memory dumps or in file viewers.
- Con: Takes up more space. The string format is larger than the binary format.
- Con: The conversion cost can add up if you have to do it a lot.
- Worst-case size is large: S-255-281474976710655-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295-4294967295 = 188 characters, if you also include the null terminator.

Binary format: To save the SID, use the `GetLengthSid` function, then save the raw bytes to the file. To load the SID, read the bytes from the file, then validate the SID to ensure it was not corrupted (either accidentally or maliciously).

- Cons: Validating a SID is tricky. (Details below.) If you mess up, you may have a security vulnerability.
- Cons: Binary format is harder to spot in a memory dump or in a file viewer.
- Pro: Takes up less space.
- Worst-case size is `SECURITY_MAX_SID_SIZE` = 68 bytes, so you might even just allocate a fixed buffer for the SID and avoid the variable-length problem.

The tricky part is validating that a chunk of memory is a valid SID.

You might think that the `IsValidSid` function would do that for you, but it can't because the function doesn't have a `cbSize` parameter, so it cannot validate that the purported SID fits inside the buffer. The `IsValidSid` function does logical validation, not physical validation. (It assumes that the memory is formatted like a SID, and it's checking whether the formatting is legal.)

Therefore, you have to do the length validation yourself, and then let `IsValidSid` do the semantic validation only after you have verified that the length is correct.

```
BOOL IsValidUntrustedSid(PSID psid, size_t cbSize)
{
    // First make sure the SID is at least the minimum size.
    // This ensures that we can read the revision and subauthority
    // count.
    if (cbSize < SECURITY_SID_SIZE(0)) return FALSE;

    // Now that we know the header is readable, we can calculate
    // the length the SID claims to be and make sure it is actually
    // that length.
    if (cbSize != GetLengthSid(psid)) return FALSE;

    // Now that we know the entire memory block is the right size,
    // we can use IsValidSid.
    return IsValidSid(psid);
}
```

Using strings is more convenient, and as long as the conversion isn't a bottleneck, and the disk space is not an issue, it would probably be a more convenient choice for a persistence format.

Note that the `ConvertStringSidToSid` function parses abbreviations for well-known SIDs. For example, you can pass `BA` and out will come the Builtin Administrators group. If you want to block that, you can first check that the string being converted begins with `S-`.

On the other hand, the security people tell me that defending against shorthand SIDs like `BA` isn't all that interesting. Since the attacker controls the string, they could just use the raw format `S-1-5-32-544` instead. Some shorthand SIDs expand to include the domain SID. For example `EA` expands to `S-1-5-21-X-519`, where `X` is the domain RID. Even if you blocked the shorthand SID, the attacker could still pass the full string `S-1-5-21-X-519`. (From a security-theoretical point of view, the SID for the domain is not considered sensitive data. You should assume that attackers already know your domain SID.)

But wait, we got all distracted with answering the question and forgot to solve the problem.

In general, it is rare to save just the SID all by itself. Usually a SID is part of a security descriptor, so you should be saving the entire security descriptor. (We saw this some time ago when we discussed how the SID history is used when a user's SID changes.)

Raymond Chen

Follow