Randomly-generated passwords still have to be legal strings

devblogs.microsoft.com/oldnewthing/20160316-00

March 16, 2016

Raymond Chen

If you need to generate a password for programmatic use, then you don't have to worry about generating characters that are difficult or impossible to type on a keyboard. Go ahead and mix Cyrillic with Vietnamese and throw in some Linear B while you're at it. There is no keyboard that can type all of these characters, but it doesn't matter because nobody will be typing it.

However, you should make sure that your password is a legal string.

> We generate our password from a cryptographically secure random number generator. Basically, we take 256 random bits and treat them as sixteen 16-bit values. (If one of the 16-bit values is zero, then we ask for 16 more bits.)
>
> We found that sometimes (no predictable pattern), we have interoperability problems between systems. The password produced by one system is not recognized by the other.

After much investigation, the problem was traced back to the fact that taking a bunch of non-null 16-bit values and declaring them to be a Unicode (UTF-16LE) string does not always result in a valid Unicode string.

UTF-16 has the concept of *surrogate pairs*, which encode characters outside the BMP as a pair of 16-bit values. The first entry in the pair is a *high surrogate* in the range `0xD800` – `0xDBFF` , and the second is a *low surrogate* in the range `0xDC00` – `0xDFFF` . Together, they encode a character in a supplementary plane.

If your randomly-generated string contains a value in the range `0xD800` – `0xDFFF` , then unless you are very lucky, it will not be part of a valid surrogate pair. The string is therefore not well-formed, and various parts of the system might decide to reject them with `ERROR_INVALID_PARAMETER` , or they might "fix" the problem by changing the illegal values to `U+FFFD` , the Unicode Replacement Character, which is used for unknown or unrepresentable character. For example, if the protocol specifies that the password is transmitted in UTF-8, then the presence of an unpaired surrogate causes the conversion from UTF-16 to UTF-8 to fail, and consequently, the password fails to replicate to the other machine.

If you want to generate a random password, make sure your algorithm produces legal character sequences. A simple solution is to generate the desired amount of entropy, then hex-encode it. Yes, it isn't very space-efficient, but it gets the job done. (Assuming you don't have to meet password complexity rules.)

Raymond Chen

Follow