Closed

Issue #30 (https://gitlab.com/libidn/libidn2/issues/30) opened a week ago by **Zbigniew Jędrzejewski-Szmek (/keszybz)**

underscores get stripped

I'm using `idn2_lookup_u8(..., IDN2_NFC_INPUT | IDN2_NONTRANSITIONAL)`. Initial underscores in labels get stripped. It this expected? Example:

"_443._tcp.fedoraproject.org" → "443.tcp.fedoraproject.org"

I'm using libidn2-2.0.2-1.fc26.x86_64.

1 Related Merge Request

!51 **TR46: Disable STD3 ASCII rules by default (/libidn/libidn2/merge_requests/51)** Closed

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

The TR46 non-transitional preprocessing removes these characters and also several others. RFC 5890 basically defines a 'label' (the parts separated by dots in a domain name) consisting only of ASCII letter, digits and hyphens. So yes, this is expected behavior with IDN2_NONTRANSITIONAL.

IDN2_TRANSITIONAL would leave those characters in place. This is definitely more backward compatible to IDNA 2003 and obsolete (by IDNA 2008) domain names.

BTW, you can leave IDN2_NFC_INPUT away. It is implicitly used by IDN2_NONTRANSITIONAL and IDN2_TRANSITIONAL.

For more details see answer 2 at https://stackoverflow.com/questions/2180465/can-domain-name-subdomains-have-an-underscore-in-it (https://stackoverflow.com/questions/2180465/can-domain-name-subdomains-have-an-underscore-in-it).

Edited a week ago by Tim Rühsen (/rockdaboot)

Tim Rühsen @rockdaboot (/rockdaboot) closed a week ago

(/keszybz)
Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) commented a week ago

Hm, can you describe where exactly in the RFC this behaviour is described? https://tools.ietf.org/html/rfc5891#section-5.4 (https://tools.ietf.org/html/rfc5891#section-5.4) gives a list of specific disallowed characters (which "_" is not on afaics), and then says "The string that has now been validated for lookup is converted to ACE form by applying the Punycode algorithm to the string and then adding the ACE prefix ("xn--")." Nowhere is stripping of characters mentioned.

(/nmav)
Nikos Mavrogiannopoulos @nmav (/nmav) commented a week ago

Owner

Are labels which contain underscore the only concern there? There could be a flag which skips these labels for processing, allowing behavior similar to libidn where one could pass resource records similarly to hostnames. A quick and dirty proof of concept is attached.

0001-skip-underscore-labels.patch (https://gitlab.com/libidn/libidn2/uploads/a786edd868e67d57f30b09364489eefd/0001-skip-underscore-labels.patch)

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

We had a similar issue with 'whois'. It throws CIDRs into toASCII and it came back without /. Like

```
$ idn2 192.168.1.0/24
192.168.1.024
```

So this is general problem... but I think TR46 proposes a flag for that. Not sure if it is functional (IDN2_ALLOW_UNASSIGNED).

(/nmav)
Nikos Mavrogiannopoulos @nmav (/nmav) commented a week ago

Owner

> Hm, can you describe where exactly in the RFC this behaviour is described?

The RFCs don't specifically say drop these characters in processing, that's libidn2 behavior. The RFCs define labels as something containing only specific ascii chars.

> So this is general problem... but I think TR46 proposes a flag for that. Not sure if it is functional (IDN2_ALLOW_UNASSIGNED).

We may want to use this flag then for that. I experimented passing verbatim characters not in a map when this flag is present but got an error later in processing.

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

I try again... it is TR46 that filters the character out. From the IdnaMappingTable.text:

```
005B..0060    ; disallowed_STD3_valid                   # 1.1  LEFT SQUARE BRACKET..GRAVE ACCENT
```

From the spec:

```
 4.1.1 UseSTD3ASCIIRules

If UseSTD3ASCIIRules=false, then the validity tests for ASCII characters are not provided by the table

There are a very small number of non-ASCII characters with the data file status disallowed_STD3_valid:

U+2260 ( ≠ ) NOT EQUAL TO
U+226E ( ≮ ) NOT LESS-THAN
U+226F ( ≯ ) NOT GREATER-THAN

Those characters are disallowed with UseSTD3ASCIIRules=true because the set of characters in their cano
```

I think, we don't have the STD3ASCII flag implemented yet, have we ?

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

We have these flags (TR46_FLG_DISALLOWED_STD3_VALID and TR46_FLG_DISALLOWED_STD3_MAPPED) already in the characte `map`, but just don't provide a flag for the API.

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

@keszybz (/keszybz) Allowed characters are first defined in RFC952:

```
A "name" (Net, Host, Gateway, or Domain name) is a text string up
   to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus
   sign (-), and period (.).  Note that periods are only allowed when
   they serve to delimit components of "domain style names". (See
   RFC-921, "Domain Name System Implementation Schedule", for
   background).  No blank or space characters are permitted as part of a
   name. No distinction is made between upper and lower case.  The first
   character must be an alpha character.  The last character must not be
   a minus sign or period.
```

RFC1123 also allowed a digit as first character.

AFAIK, this is still true. IDNA transforms international strings/domains into this old naming scheme (doing some processing and then using the punycode_encode algorithm). I wish we could simply use UTF-8 instead.

(/nmav)
Nikos Mavrogiannopoulos @nmav (/nmav) commented a week ago

Owner

Another patch which takes advantage of Tim's advice above, though most likely it shouldn't use the unassigned flag but another one. I give up for now.

patch.txt (https://gitlab.com/libidn/libidn2/uploads/3b639dc186af10d299bf4fec0eec8273/patch.txt)

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

@nmav (/nmav) Thanks for the patch. Let me check it against the TR46 spec in the next 1-2 days.

(/keszybz)
Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) commented a week ago

I'm not sure what the right solution is, so let me describe the problem better: underscores are used in DNS names for example to specify service fields (_tcp, _http, ..., e.g. RFC 6698). The underscore is used because it is not allowed in host names (RFC 1123, §2.1) (as you wrote above while I was typing this...) but allowed in DNS labels. Such labels are automatically constructed by combining a user-specified domain and the prefix (e.g. _443._tcp. to resolve TLS certificates for HTTPS). In particular, this might be done for a domain like fa8.de.

What we did so far was to take the address and pass it through IDNA encoding, and resolve that. With libidn, we had _443._tcp.fa8.de encoded as .443._tcp.fass.de. With libidn2 and IDN2_NONTRANSITIONAL I get 443.tcp.xn--fa-hia.de, which cannot work. With libidn2 and IDN2_TRANSITIONAL I get _443._tcp.fass.de. But I really need _443._tcp.xn--fa-hia.de, i.e. the new rules but with underscores preserved.

I have very strong doubts about anything which is not round-trippable, but I need to look at this some more. I'll give your patch a test.

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

> But I really need _443._tcp.xn--fa-hia.de

Default for IDNA2008/TR46 processing is UseSTD3ASCIIRules=true. What you need is UseSTD3ASCIIRules=false, which we didn't implement yet (maybe Nikos's patch above does it). With that you have to check your domain string for validity yourself because you circumvent some of the internal tests.

What you could do right now is to pass only the last part from your string to the idn2_ function. You know already that the first part is fine and needs no processing (_443._tcp.). IDNA processing is always label-by-label, so it's fine to split the input string that way.

(/keszybz)
Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) commented a week ago

I'll reopen this, at least because a patch is being discussed...

Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) reopened a week ago

(/keszybz)
Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) commented a week ago

> What you could do right now is to pass only the last part from your string to the idn2_ function.

This would be problematic. Right now the client constructs a name and send a query to a daemon to have it resolved, as utf-8. And the daemon takes care of idn processing (for DNS) or not (e.g. for LLMNR). So doing that would require both the client to be much smarter, and extra communication about the meaning of specific labels... I'd rather not go there.

> Another patch which takes advantage of Tim's advice above

Yep. Patch from #30 (comment 34723449) (https://gitlab.com/libidn/libidn2/issues/30#note_34723449) seems to work fine.

```
$ systemd-resolve _443._tcp.fa8.de
_443._tcp.fa8.de: 72.52.4.119
                  (_443._tcp.xn--fa-hia.de)

-- Information acquired via protocol DNS in 1.6ms.
-- Data is authenticated: no
```

Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) mentioned in commit `unofficial-mirrors/systemd@7f7ab228` (/unofficial-mirrors/systemd/commit/7f7ab22892a14ad152d2367b23eeb7df80913ff5) a week ago

(/rockdaboot)
Tim Rühsen @rockdaboot (/rockdaboot) commented a week ago

Owner

Fixed up @nmav (/nmav)'s patch, added `--usestd3asciirules` to idn2, changing default behavior to *not* use STD3 ascii rules. These rules can be enabled with the `IDN2_USE_STD3_ASCII_RULES` flag.

Unicode's TR46 document wants STD3 be enabled by default... so I am not sure if we should work against it. The plus is that with patch !51 (closed) (/libidn/libidn2/merge_requests/51) we follow old libidn/IDNA2003 behavior.

(/nmav)
Nikos Mavrogiannopoulos @nmav (/nmav) commented 5 days ago

Owner

Resolved by `a5cbc16e` (/libidn/libidn2/commit/a5cbc16efd02adb78d2d082b21c3ac4d3fa88d2e)

Nikos Mavrogiannopoulos @nmav (/nmav) closed 5 days ago

(/nmav)
Nikos Mavrogiannopoulos @nmav (/nmav) commented 4 days ago

Assignee
No assignee

Milestone
None

Time tracking
No estimate or time spent

Due date
No due date

Labels
None

Weight
None

3 participants
(/keszybz) (/nmav)
(/rockdaboot)

Reference: libidn/libidn2#30

Owner

@keszybz (/keszybz) would a release with this fix only be sufficient to move systemd to libidn2?

(/keszybz)
Zbigniew Jędrzejewski-Szmek @keszybz (/keszybz) commented 4 days ago
I think so. I haven't merged the corresponding patch to systemd yet, but it's very simple.

Please register (/users/sign_in?redirect_to_referer=yes) or sign in (/users/sign_in?redirect_to_referer=yes) to comment