python / cpython Public

Code Issues 5k+ Pull requests 2.2k Actions Projects Security and q

Commit b8b4b71

4 people authored on Feb 19, 2025 Partially verified

```
[3.10] gh-105704: Disallow square brackets ([ and ]) in domain names for parsed
URLs (GH-129418) (#129529)

(cherry picked from commit d89a5f6)

Co-authored-by: Seth Michael Larson <seth@python.org>
Co-authored-by: Peter Bierma <zintensitydev@gmail.com>
Co-authored-by: Łukasz Langa <lukasz@langa.pl>
```

3.10 (#129529) · v3.10.20 ••• v3.10.17

1 parent 8175648 commit b8b4b71

3 files changed +58 -3 lines changed

Top

Filter files…

- Lib
  - test
    - test_urlparse.py
  - urllib
    - parse.py
- Misc/NEWS.d/next/Security
  - 2025-01-28-14-08-03.gh-issue-105704.EnhHxu.rst

3 files changed +58 -3 lines changed

Search within code

Lib/test/test_urlparse.py

```
@@ -1149,16 +1149,51 @@ def test_invalid_bracketed_hosts(self):
```

```
1149  1149          self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'Scheme://user@[0439:23af::2309::fae7:1234]/Path?Query')
1150  1150          self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'Scheme://user@[0439:23af:2309::fae7:1234:2342:438e:192.0.2.146]/Path?Query')
1151  1151          self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'Scheme://user@]v6a.ip[/Path')
      1152  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[v6a.ip]')
      1153  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[v6a.ip].suffix')
      1154  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[v6a.ip]/')
      1155  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[v6a.ip].suffix/')
      1156  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[v6a.ip]?')
      1157  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[v6a.ip].suffix?')
      1158  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[::1]')
      1159  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[::1].suffix')
      1160  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[::1]/')
      1161  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[::1].suffix/')
      1162  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[::1]?')
      1163  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[::1].suffix?')
      1164  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[::1]:a')
      1165  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[::1].suffix:a')
      1166  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[::1]:a1')
      1167  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://[::1].suffix:a1')
      1168  +       self.assertRaises(ValueError, urllib.parse.urlsplit,
                    'scheme://prefix.[::1]:1a')
```

```
        1169  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://[::1].suffix:1a')
        1170  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://prefix.[::1]:')
        1171  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://[::1].suffix:/')
        1172  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://prefix.[::1]:?')
        1173  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://user@prefix.[v6a.ip]')
        1174  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://user@[v6a.ip].suffix')
        1175  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://[v6a.ip')
        1176  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://v6a.ip]')
        1177  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://]v6a.ip[')
        1178  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://]v6a.ip')
        1179  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://v6a.ip[')
        1180  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://prefix.[v6a.ip')
        1181  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://v6a.ip].suffix')
        1182  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://prefix]v6a.ip[suffix')
        1183  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://prefix]v6a.ip')
        1184  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                  'scheme://v6a.ip[suffix')
1152    1185
1153    1186      def test_splitting_bracketed_hosts(self):
1154          -       p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]/path?query')
        1187  +       p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]:1234/path?query')
1155    1188          self.assertEqual(p1.hostname, 'v6a.ip')
1156    1189          self.assertEqual(p1.username, 'user')
1157    1190          self.assertEqual(p1.path, '/path')
        1191  +       self.assertEqual(p1.port, 1234)
```

```
1158  1192            p2 =
                      urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7%test]/path?query')
1159  1193            self.assertEqual(p2.hostname, '0439:23af:2309::fae7%test')
1160  1194            self.assertEqual(p2.username, 'user')
1161  1195            self.assertEqual(p2.path, '/path')
      1196  +         self.assertIs(p2.port, None)
1162  1197            p3 =
                      urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7:1234:192.0.2.146%test]/path?query')
1163  1198            self.assertEqual(p3.hostname,
                      '0439:23af:2309::fae7:1234:192.0.2.146%test')
1164  1199            self.assertEqual(p3.username, 'user')
```

Lib/urllib/parse.py

```
                  @@ -442,6 +442,23 @@ def _checknetloc(netloc):
442   442               raise ValueError("netloc '" + netloc + "' contains invalid " +
443   443                                "characters under NFKC normalization")
444   444
      445 + def _check_bracketed_netloc(netloc):
      446 +     # Note that this function must mirror the splitting
      447 +     # done in NetlocResultMixins._hostinfo().
      448 +     hostname_and_port = netloc.rpartition('@')[2]
      449 +     before_bracket, have_open_br, bracketed = hostname_and_port.partition('[')
      450 +     if have_open_br:
      451 +         # No data is allowed before a bracket.
      452 +         if before_bracket:
      453 +             raise ValueError("Invalid IPv6 URL")
      454 +         hostname, _, port = bracketed.partition(']')
      455 +         # No data is allowed after the bracket but before the port delimiter.
      456 +         if port and not port.startswith(":"):
      457 +             raise ValueError("Invalid IPv6 URL")
      458 +     else:
      459 +         hostname, _, port = hostname_and_port.partition(':')
      460 +     _check_bracketed_host(hostname)
      461 +
445   462   # Valid bracketed hosts are defined in
446   463   # https://www.rfc-editor.org/rfc/rfc3986#page-49 and
                https://url.spec.whatwg.org/
447   464   def _check_bracketed_host(hostname):
```

```
@@ -505,8 +522,7 @@ def urlsplit(url, scheme='', allow_fragments=True):
                    (']' in netloc and '[' not in netloc)):
                raise ValueError("Invalid IPv6 URL")
            if '[' in netloc and ']' in netloc:
-               bracketed_host = netloc.partition('[')[2].partition(']')[0]
-               _check_bracketed_host(bracketed_host)
+               _check_bracketed_netloc(netloc)
        if allow_fragments and '#' in url:
            url, fragment = url.split('#', 1)
        if '?' in url:
```

...5-01-28-14-08-03.gh-issue-105704.EnhHxu.rst

```
@@ -0,0 +1,4 @@
+ When using :func:`urllib.parse.urlsplit` and :func:`urllib.parse.urlparse` host
+ parsing would not reject domain names containing square brackets (``[`` and
+ ``]``). Square brackets are only valid for IPv6 and IPvFuture hosts according to
+ `RFC 3986 Section 3.2.2 <https://www.rfc-editor.org/rfc/rfc3986#section-3.2.2>`__.
```

Comments 0